Learning Classifiers from Chains of Multiple Interlinked RDF Data Stores

Harris T. Lin and Vasant Honavar
Department of Computer Science
Iowa State University
Ames, IA 50011 USA
{*htlin,honavar*}@*iastate.edu*

***Abstract*—The emergence of many interlinked, physically distributed, and autonomously maintained RDF stores offers unprecedented opportunities for predictive modeling and knowledge discovery from such data. However existing machine learning approaches are limited in their applicability because it is neither desirable nor feasible to gather all of the data in a centralized location for analysis due to access, memory, bandwidth, computational restrictions, and sometimes privacy and confidentiality constraints. Against this background, we consider the problem of learning predictive models from *multiple interlinked* RDF stores. Specifically we: (i) introduce statistical query based formulations of several representative algorithms for learning classifiers from RDF data; (ii) introduce a distributed learning framework to learn classifiers from multiple interlinked RDF stores that form a chain; (iii) identify three special cases of RDF data fragmentation and describe effective strategies for learning predictive models in each case; (iv) consider a novel application of a matrix reconstruction technique from the field of Computerized Tomography [1] to approximate the statistics needed by the learning algorithm from projections using count queries, thus dramatically reducing the amount of information transmitted from the remote data sources to the learner; and (v) report results of experiments with a real-world social network data set (Last.fm), which demonstrate the feasibility of the proposed approach.**

***Keywords*-classifier; supervised learning; distributed learning; RDF; SPARQL; linked data**

I. Introduction

The growing adoption of a set of best practices, collectively referred to as Linked Data, for publishing structured data on the Web [2], has made it possible to link and share many disparate, previously isolated, distributed, autonomously generated and managed data across virtually every domain of human endeavor. The community-driven Linked Open Data (LOD) effort allows structured data to be represented using Resource Description Framework (RDF, [3]) in the form of subject-predicate-object triples (also called RDF triples), which describe a directed graph where the directed labeled edges encode binary relations between labeled nodes. RDF stores and associated query languages such as SPARQL [4] offer the means to store and query large amounts of RDF data. LOD also enables integration of previously isolated distributed data such as data stored in multi-relational databases [5]. At present, LOD include a few hundred linked data sets that together contain in excess of a few trillion RDF triples [6]. These cover a broad range of domains including government, life sciences, geography, social media, and commerce. The emergence of LOD offers unprecedented opportunities for using disparate data sources in predictive modeling and decision making in such domains.

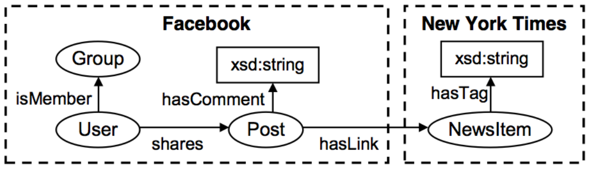

Figure 1. A motivating scenario of two RDF stores that are linked to form a *chain* of RDF stores: Facebook users share posts about news items published in New York Times.

We motivate the problem of learning predictive models from multiple interlinked RDF stores using the scenario shown in Fig. 1. In this case, one might want to use data from Facebook and New York Times to predict the interest of a user in belonging to a Facebook group, based on the distribution of tags associated with the New York Times news stories that the user has shared with her social network on Facebook. This is an instance of the *node prediction problem* [7]. In general, building such predictive models entails using information from multiple interlinked, physically distributed, autonomously maintained RDF stores. In such a setting, it is neither desirable nor feasible to gather all of the data in a centralized location for analysis, because of access, memory, bandwidth, and computational restrictions. In other settings, access to data may be limited due to privacy and confidentiality constraints [8], [9]. This calls for techniques for learning predictive models (e.g. classifiers) from multiple interlinked RDF stores that support only *indirect* access to data (e.g. via a query interface such as SPARQL). Barring Lin et al. [10] who proposed an approach to learning relational Bayesian classifiers [11] from a *single* remote RDF store using statistical queries against its SPARQL endpoint, to the best of our knowledge, there has been very little work on this problem.

Against this background, we consider the problem of learning predictive models from *multiple interlinked* RDF stores. Specifically we: (i) introduce statistical query based formulations of several representative algorithms for learning classifiers from RDF data; (ii) introduce a distributed learning framework to learn classifiers from multiple interlinked RDF stores that form a chain; (iii) identify three special cases of RDF data fragmentation and describe effective strategies for learning predictive models in each case; (iv) consider a novel application of a matrix reconstruction technique from the field of Computerized Tomography [1] to approximate the statistics needed by the learning algorithm from projections using count queries, thus dramatically reducing the amount of information transmitted from the remote data sources to the learner; and (v) report results of experiments with a real-world social network data set (Last.fm), which demonstrate the feasibility of the proposed approach.

The paper is organized as follows: Sec. II begins with learning classifiers from a single remote RDF store. Sec. III extends learning to multiple interlinked RDF stores. Sec. IV describes the experiments and the results. Finally Sec. V concludes with a summary, related work, and future work.

II. Learning Classifiers from RDF Data

A. RDF Learner Defined

Recall that an RDF triple is $(s, p, o) \in (I \cup B) \times I \times (I \cup B \cup L)$ where s is the subject, p the predicate, and o the object of the triple and I, B, and L are pairwise disjoint infinite sets of URIs, Blank nodes, and Literals respectively. An RDF graph is a set of RDF triples. Given an RDF graph $\mathcal{G}$, and a *target class* $\mathcal{T}$ which is a distinguished URI of type rdfs:Class in $\mathcal{G}$, we denote the set of instances of the target class as $\mathcal{T}(\mathcal{G}) = \{x : (x, \text{rdf:type}, \mathcal{T}) \in \mathcal{G}\}$. An *attribute* A (of a target class $\mathcal{T}$) is a tuple of predicates $(p_1, \ldots, p_J)$ such that the domain of p_1 is $\mathcal{T}$, the range of p_j is the domain of p_{j+1}, and the range of p_J is a literal. Given an instance x_i of the target class $\mathcal{T}$ and an attribute $A_k = (p_1^k, \ldots, p_J^k)$, we define B_k^i to be the bag (multi-set) of literals matched by the variable $?\,v_J$ in the Basic Graph Pattern [4] $((x, p_1^k, ?\,v_1)$ AND $(?\,v_1, p_2^k, ?\,v_2) \ldots (?\,v_{J-1}, p_J^k, ?\,v_J))$ where $v_j \in V$ are variables. For convenience we denote the range of p_J^k by Δ_k, and let $|\Delta_k| = s_k$. A *target attribute* is a distinguished attribute denoted by A_c, which describes the *class label* of an instance, hence we assume that each instance has exactly one class label, i.e., $\left|B_c^i\right| = 1$ for every $x_i \in \mathcal{T}(\mathcal{G})$; for brevity the class label is denoted by c_i and the set of all possible values of c_i is denoted by $\mathcal{C}$. An *RDF data set* D is a tuple $(\mathcal{G}, \mathcal{T}, \mathcal{A}, A_c)$ where $\mathcal{G}$ is an RDF graph, $\mathcal{T}$ a target class in $\mathcal{G}$, $\mathcal{A} = (A_1, \ldots, A_K)$ a tuple of attributes, and A_c is a target attribute. Given an RDF data set $D = (\mathcal{G}, \mathcal{T}, \mathcal{A}, A_c)$, its induced *multiset attributed data set* [10] is defined as $\mathcal{M}(D) = \{((B_1^i, \ldots, B_K^i), c_i) : x_i \in \mathcal{T}(\mathcal{G})\}$.

Definition 1. The input to an RDF node classifier h is $(B_1^i, \ldots, B_K^i)$ where x_i is an instance of a target class $\mathcal{T}$, and the output $h(x_i) \in \mathcal{C}$ is a class label.

An RDF Learner L [10] is an algorithm that given an RDF data set $D = (\mathcal{G}, \mathcal{T}, \mathcal{A}, A_c)$, its induced multiset attributed data set $\mathcal{M}(D)$, a hypothesis class H, and a performance criterion P, outputs a classifier $h \in H$ that optimizes P.

B. Representative Classes of RDF Learners

We consider two basic approaches to learn from RDF data: (i) those that rely on *aggregation* to encode nodes to be classified as tuples of attribute values, i.e., instances that can be handled by traditional supervised machine learning algorithms; and (ii) those that are based on *generative models* of data.

1) Aggregation: Here we represent each bag of attributes in $\mathcal{M}(D)$ by a single value, by applying a suitable aggregation function, e.g., min, max, $average$ for continuous values and $mode$ for discrete values. Hence we reduce the data set into a traditional attribute-value data set where each instance is represented by a finite number of attributes, each of which takes a single value from the set of possible values for the corresponding attribute.

We also consider more sophisticated aggregation schemes proposed by Perlich and Provost [12]. WLOG, consider binary class labels ($\mathcal{C} = \{+, -\}$), and an attribute A_k with discrete values (i.e. Δ_k is a finite set). Suppose that B_k^i is the bag of values for the k^{th} attribute of an instance x_i. We define $V_k^i = \left(v_{k1}^i, \ldots, v_{ks_k}^i\right)$ to be a vector of counts of values in B_k^i where v_{kt}^i is the number of occurrences of the t^{th} value $d_{kt} \in \Delta_k$. Next we define a class-conditional reference vector for $c \in \mathcal{C}$ as $V_k^{(c)} = \sum_i \delta_{c,c_i} V_k^i$ where δ is the Kronecker delta. A number of aggregation schemes can be defined using various measures of distance between V_k^i and the reference vectors [12]. Here we outline a representative model called Class-Conditional Vector Distances (CCVD). Let $DIST$ be a set of M distance functions between two vectors such as cosine or Euclidean, then we compute a $|\mathcal{C}|M$-sized vector, e.g., $(dist_m(V_k^{(+)}, V_k^i),\ dist_m(V_k^{(-)}, V_k^i))_{m=1}^M$. We concatenate these vectors from each of the K bags of attributes of x_i to obtain a single attribute vector of length MK.

Regardless of which aggregation scheme described above, by applying an aggregation scheme to each of the instances in $\mathcal{M}(D)$, we can effectively reduce the problem of learning from an RDF data set to the well-studied problem of supervised learning in the traditional setting where each instance to be classified is represented by a tuple of attribute values.

2) Generative Models: We consider a joint distribution $p(B_1, \ldots, B_K, c)$. For simplicity, under the naive Bayes (NB) assumption that bags of attributes are conditionally independent given the class label c the most probable class label is given by:

$$\begin{aligned} h_{NB}(x) &\triangleq \arg\max_{c\in\mathcal{C}} p(c \mid B_1,\ldots,B_K) \\ &= \arg\max_{c\in\mathcal{C}} p(c)\prod_{k=1}^{K} p(B_k \mid c)\,. \end{aligned}$$

We can now consider a variety of models for $p(B_k \mid c)$ including those based on Bernoulli or multinomial event models [13], Dirichlet distribution [14], [15] or Dirichlet-multinomial (Polya) distribution [16], [15]. We denote these models by NB(Ber), NB(Mul), NB(Dir), and NB(Pol) respectively, and outline each of them below.

Let $b_{kt} \in \{1,0\}$ denote the presence or absence of $d_{kt} \in \Delta_k$ in an attribute bag B_k and, similarly, let v_{kt} denote the number of occurrences of d_{kt}. A class-conditional bag probability, $p(B_k \mid c)$, can be modeled by event models such as Bernoulli (1) or multinomial (2):

$$p(B_k \mid c;\boldsymbol{\theta}) \triangleq \prod_{t=1}^{s_k} \theta_{ckt}^{b_{kt}} (1-\theta_{ckt})^{1-b_{kt}} \tag{1}$$

$$p(B_k \mid c;\boldsymbol{\theta}) \triangleq p(|B_k|)\,\frac{\left(\sum_{t=1}^{s_k} v_{kt}\right)!}{\prod_{t=1}^{s_k} v_{kt}!}\prod_{t=1}^{s_k}\theta_{ckt}^{v_{kt}} \tag{2}$$

where $\theta_{ckt} = p(d_{kt} \mid c)$.

Next, the Dirichlet distribution (3) allows us to treat B_k as a sample from a distribution which, in turn, is drawn from another distribution as follows

$$\begin{aligned} p(B_k \mid c;\boldsymbol{\alpha}) &\triangleq p\left(\bar{V}_k \mid c\right) \triangleq \mathcal{D}(\boldsymbol{\alpha}_{ck}) \\ &= \frac{\Gamma(\sum_{t=1}^{s_k}\alpha_{ckt})}{\prod_{t=1}^{s_k}\Gamma(\alpha_{ckt})}\prod_{t=1}^{s_k}\bar{v}_{kt}^{\alpha_{ckt}-1} \end{aligned} \tag{3}$$

where $\boldsymbol{\alpha}_{ck} = (\alpha_{ck1},\ldots,\alpha_{cks_k})$ is a vector parameter of Dirichlet distribution for class $c \in \mathcal{C}$ and $\bar{V}_k = (\bar{v}_{k1}\cdots\bar{v}_{ks_k})$ is the *normalized* vector of counts of values in B_k with $\bar{v}_{kt} = v_{kt}/\sum_t v_{kt}$. Finally, we describe the Dirichlet-multinomial (Polya) distribution (4) that compounds a Dirichlet with a multinomial:

$$\begin{aligned} p(B_k \mid c;\boldsymbol{\alpha}) &\triangleq p(V_k \mid c) \\ &\triangleq \int p(V_k;\boldsymbol{\theta}_{ck})\,p(\boldsymbol{\theta}_{ck};\boldsymbol{\alpha}_{ck})\,d\boldsymbol{\theta}_{ck} \\ &= \frac{\Gamma\left(\sum_t \alpha_{ckt}\right)}{\Gamma\left(\sum_t v_{kt}+\alpha_{ckt}\right)}\prod_{t=1}^{s_k}\frac{\Gamma(v_{kt}+\alpha_{ckt})}{\Gamma(\alpha_{ckt})} \end{aligned} \tag{4}$$

where $\boldsymbol{\theta}_{ck} = (\theta_{ck1},\ldots,\theta_{cks_k})$ is a vector of multinomial parameters.

For the above four models, their parameters, which is a set of parameters for each class and for each attribute, can be estimated by maximum likelihood employing the Laplace correction.

C. Sufficient Statistics

We describe the sufficient statistics to estimate the parameters (via maximum likelihood) for each attribute A_k and for each of the models in Sec. II-B, and provide the corresponding SPARQL queries to obtain these statistics.

- Aggregation function: $agg(B_k^i)$ and the class label for each instance x_i where agg is some aggregation function. If naive Bayes is learned on the reduced data set then the following is sufficient: number of instances with the class label c and $d = agg(B_k^i)$ for every combination of c and d. The former can be expressed by an aggregation query and the later is equivalent to $S(\mathcal{G},\mathcal{T},C=c,A_k,agg,d)$ in [10].
- CCVD and NB(Pol): for each $c \in \mathcal{C}$, V_k^i for each instance x_i such that $c_i = c$. Its SPARQL query can be expressed by:

```
SELECT ?x ?vj COUNT(?vj) WHERE {
  ?x rdf:type <T> .
  ?x <classLabel> c .
  ?x <p1> ?v1 .  ... ?vj-1 <pj> ?vj .
} GROUP BY ?x ?vj
```

- NB(Ber) and NB(Mul): $V_k^{(c)}$ for each $c \in \mathcal{C}$. Its SPARQL query can be expressed by:

```
SELECT ?vj COUNT(?vj) WHERE {
  ?x rdf:type <T> .
  ?x <classLabel> c .
  ?x <p1> ?v1 .  ... ?vj-1 <pj> ?vj .
} GROUP BY ?vj
```

- NB(Dir): for each $c \in \mathcal{C}$, $log(\bar{v}_{kt}) = \sum_i log(v_{kt}^i) - log(\sum_t v_{kt}^i)$ for the subset of instances x_i such that $c_i = c$ (see [15])[1]. An alternative statistic though not minimal is V_k^i for each instance x_i such that $c_i = c$.

D. Approximating V_k^i

In Sec. II-C we observe that V_k^i (conditioned on some class $c \in \mathcal{C}$) is required for a number of models (CCVD, NB(Dir), and NB(Pol)). However, as shown by the experiment in Sec. IV-D, obtaining V_k^i for each x_i is quite expensive. Instead, it is much cheaper to obtain approximate summaries of each V_k^i. For example, define $V_{kt}^* = \sum_i v_{kt}^i$ and let $V_k^* = \left(v_{k1}^*,\ldots,v_{ks_k}^*\right)$; similarly define $V_{k*}^i = \sum_t v_{kt}^i$ and let $V_{k*} = \left(v_{k*}^1,\ldots,v_{k*}^I\right)$ where I is the total number of instances. In other words, if V_k^i for all $i \in [1,I]$ are represented as an I-by-s_k matrix where v_{kt}^i is the value of row-i and column-t, then V_k^* and V_{k*} are its column and row projections respectively. We say that V_k^* is the projection towards the *leaf* of the attribute chain A_k and hence we refer V_k^* as the *leaf projection*[2]; similarly we refer V_{k*} as the *root projection*. Here we propose to approximate V_k^i from the leaf and root projections in order to save the size of communication.

The problem of reconstructing a matrix from its projections is closely related to the problem of reconstructing a

[1]This requires log function which is currently not supported by SPARQL.
[2]Equivalently, $V_k^{(c)}$ is the leaf projection for those x_i where $c_i = c$.

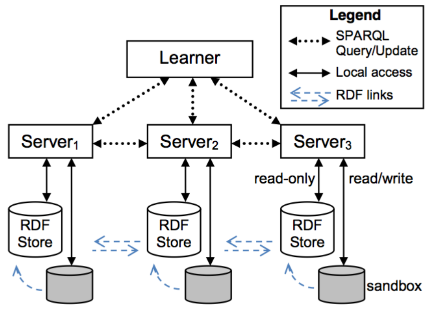

Figure 2. Distributed learning framework from multiple interlinked RDF stores. In practice there can be interactions (queries and RDF links) between any two data sources, in the figure only the adjacent interactions are drawn for simplicity.

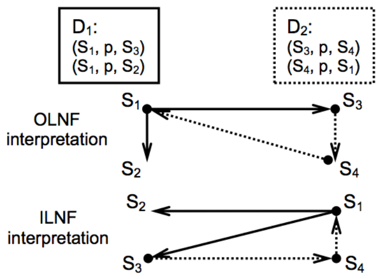

Figure 3. Two fragmented data sources D_1 and D_2 showing an example of both OLNF and ILNF, because both subject resources $S_1 \cup \{S_3, S_4\}$ and object resources $\{S_2, S_3\} \cup \{S_1, S_4\}$ for each fragment are disjoint. However it is not LFNF because S_1 and S_3 are shared.

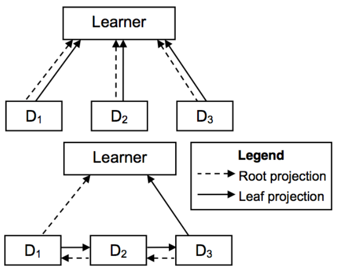

Figure 4. Computation of root projection and leaf projection under different data fragmentations: (i) LFNF (top); and (ii) OLNF and ILNF (bottom).

3D representation of an object from images of its slices, that has been widely studied in the field of Computerized Tomography (CT) (see [1] for a review). The problem of reconstructing V_k^i from its root and leaf projections is a special case of the matrix reconstruction problem. Hence, we can adapt existing approaches from CT, and one of the simplest such methods is Algebraic Reconstruction Technique (ART, [17]). ART is an iterative algorithm for solving a system of linear equations where each equation encodes the projection angle and its projected value from a matrix. Here we describe the update equation in our simplified case of column and row projections. Let x_t^i be the element of row-i and column-t of an I-by-T matrix, representing our reconstructed matrix; and let X^* and X_* be its column and row projections respectively. Let V^* and V_* be the true column and row projections respectively (i.e., those computed from the original matrix). Then the update equation is $x_t^i := x_t^i + \lambda \left(\frac{V_*^i - X_*^i}{T} + \frac{V_t^* - X_t^*}{I} \right)$ where λ is a relaxation parameter between 0 and 1.

III. Learning Classifiers from Multiple Interlinked RDF Data Stores

We now turn to the problem of learning predictive models from *multiple*, interlinked data sources. Consider the scenario shown in Fig. 2. We assume that each data source corresponds to an RDF store that can be queried through an access interface (e.g. SPARQL query server), and (optionally) a sandbox that is set up with write access for each user (i.e. learner). We can use SPARQL 1.1 update queries [4] to store intermediate results of queries in the sandbox for use by the learner. Also we can use SPARQL 1.1 federated queries [4] to retrieve query results from other remote servers as needed and store them in the sandbox for use by the learner. Thus, an RDF data set D is fragmented across sites $[1, N]$ into data set fragments $D_1, \ldots, D_N$ such that $\bigcup_{n=1}^N D_n = D$; we further assume that the learner may be subject to access constraints $Z_1, \ldots, Z_N$ associated with $D_1, \ldots, D_N$ such that $\bigcup_{n=1}^N Z_n = Z$. An access constraint may restrict the class of queries that can be answered by a data source, e.g. due to privacy considerations or the query answering capabilities of the data source. The task of learning from multiple interlinked RDF stores can be stated as follows.

Definition 2. A Distributed RDF Learner L_d is an algorithm that given the fragments $D_1, \ldots, D_N$ of a training data set D distributed across the sites $[1, N]$ through a set of access interfaces $A_1, \ldots, A_N$ with access constraints $Z = \bigcup_{n=1}^N Z_n$, a hypothesis class H, and a performance criterion P, outputs a classifier $h \in H$ that optimizes P using only the interactions against D that are allowed by Z.

It is useful to consider three generic ways in which an RDF data set can be fragmented across multiple interlinked RDF stores.

A. *Characterizing RDF Data Fragmentation*

The simplest case of RDF data set fragmentation corresponds to the setting where there are no links between individual data stores. However, in general, the data stores may contain triples (edges) that link two or more data stores. E.g., a triple (i, c, j) could be in D_1 while (j, c, k) could be in D_2. We refer to the set of all resources that play the role of either the *subject* or the *object* of an RDF triple in a data set as the *resources* of a data set; we use the term *subject resources* to refer to the set of all resources that appear *only* as the subject of an RDF triple in the data set; we use the term *object resources* to refer to the set of all resources that appear *only* as the object of an RDF triple in the data set. We can now identify three special cases of data fragmentation across multiple interlinked RDF stores (Fig. 3):

1) The *link-free normal form* (LFNF) where different fragments do not share any resources.
2) The *out-link normal form* (OLNF) where different fragments do not share any *subject* resources.
3) The *in-link normal form* (ILNF) where different fragments do not share any *object* resources.

The scenario in Fig. 1 is an example of OLNF. Note that formally an RDF store holds a set of triples (edges), and in

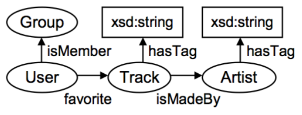

Figure 5. RDF Schema of Last.fm data set.

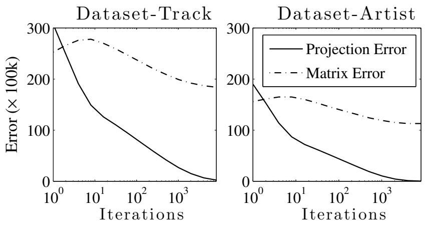

Figure 6. Projection and matrix errors at various stages of ART approximation.

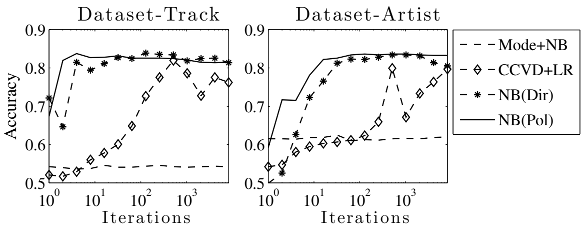

Figure 7. Classification performance using data reconstructed at various stages of ART approximation.

general the resources (nodes) are not necessarily owned by any data store; thus, it is possible that a set of data sources can simultaneously conform to both OLNF and ILNF as in Fig. 3. However in practice, the domain name of a resource often indicates its ownership; hence if a set of data sources satisfy both OLNF and ILNF we can use the domain name of the resources to determine which normal form is more appropriate to use. We further note that the three normal forms described above are not exhaustive, i.e., an RDF data set can, in general, be fragmented in ways that do not conform to any of the three normal forms considered here.

We observe that obtaining the statistics needed for learning classifiers from multiple RDF stores when the data fragmentation corresponds to LFNF reduces to combining the results of the statistical queries from the individual sources, e.g., the root and leaf projections (see Fig. 4, top). Because we can decompose the statistical queries in this fashion, the communication complexity of learning classifiers from multiple RDF data sources in LFNF is equivalent to that of learning classifiers from a single RDF store obtained by taking the union of the RDF triples from the respective sources.

B. Learning Classifiers under OLNF and ILNF

WLOG we focus on only OLNF. We consider a chain of interlinked RDF stores (e.g. Fig. 1). Specifically, we address the problem of obtaining the sufficient statistics in Sec. II-C without having to gather the data from multiple RDF stores into a central location. First we describe how to obtain the leaf and root projections in such a setting. Consider an attribute A_k, and let J_n be the number of resources shared between D_n and D_{n+1},. We define a J_n-by-J_{n+1} matrix M_n where the value at row-r_n and column-c_n is the total number of paths that connect the shared resources indexed by r_n and c_n respectively. We set J_0 to be I (number of instances), and set J_N to be s_k (number of possible values for attribute A_k). Now, we have $V_k^* = \mathbf{1}^T M_1 \cdots M_N$ and $V_{k*} = M_1 \cdots M_N \mathbf{1}$. We note that it is more efficient to multiply the matrices from the left to right for V_k^*, and from right to left for V_{k*} (see Fig. 4, bottom). Thus, the leaf projection $V_k^* = \left(v_{k1}^*, \ldots, v_{ks_k}^*\right)$ is computed starting at D_1 by transferring $\mathbf{1}^T M_1$ to D_2, and so on ending up with V_k^* at D_N which is then transferred to the learner. The root projection $V_{k*} = \left(v_{k*}^1, \ldots, v_{k*}^I\right)$ is computed in a similar fashion starting at D_N and working towards D_1. It is easy to see that the communication costs associated with the computation of the leaf and root projections respectively are given by $s_k + \sum_{n=1}^{N-1} J_n$ and $I + \sum_{n=1}^{N-1} J_n$.

In the case of V_k^i required by CCVD, NB(Dir), and NB(Pol), we first gather the leaf projection V_k^* and the root projection V_{k*} as described above, and use ART to reconstruct the corresponding V_k^i (see Sec. II-D), which is used to construct the predictive model. In the case of aggregation, we use the approximated V_k^i to compute the aggregation function $agg(B_k^i)$.

We note that NB(Ber) and NB(Mul) classifiers can be learned from the leaf projections (for each class $c \in \mathcal{C}$) alone, which guarantees that the classifiers learned from an OLNF fragmented RDF data set are identical to their centralized counterparts (that are learned from the data set obtained by combining the fragments).

IV. Experiments and Results

A. Data Sets

We used a real world data set crawled from a social music network Last.fm[3] using its API (its schema is shown in Fig. 5). We selected two disjoint groups that contain approximately equal number of users (2098/2081), and include those tracks and artists whose number of occurrences are greater than or equal to 45 and 100, respectively. Likewise, we eliminated all the track's tags and artist's tags that occurred fewer than 350 and 120 times. All collections of tags are preprocessed by removing stop words and stemming, using Apache Lucene. The resulting data set is converted to RDF format which includes 8340 tracks attributed to one or more of the 3753 artists. From this data, we extracted two subsets: (i) Dataset-Track, which includes only the tags associated with the tracks; and (ii) Dataset-Artist, which includes only the tags associated with the artists. In both cases, the task is to predict the group of the user. We simulate the OLNF setting by suitably fragmenting the datasets. For example in the case of Dataset-Track we store the triples of $isMember$ and $favorite$ in D_1 and the triples of $hasTag$ in D_2 such that $Track$ resources are shared between D_1 and D_2.

Table I
RESULTS FOR EXPERIMENT IV-B THAT REPORT ACCURACY (%) AND STANDARD DEVIATION (IN PARENTHESES) FROM 10-FOLD CROSS VALIDATION. STARRED (*) INDICATES THE OLNF MODEL IS PROVABLY EXACT WITH RESPECT TO ITS CENTRALIZED COUNTERPART. BOLDED INDICATES BEST RESULTS FOR EACH COLUMN BASED ON PAIRED t-TEST ON 10-FOLD CROSS VALIDATION WITH ALPHA = 0.05.

Model	Dataset-Track		Dataset-Artist	
	Centralized	OLNF	Centralized	OLNF
Mode+NB	71.4(3.2)	53.2(1.2)	70.8(2.5)	59.8(1.4)
CCVD+LR	**81.1(2.3)**	75.7(3.5)	**81.7(1.9)**	68.9(6.3)
NB(Ber)	71.3(2.5)	71.3(2.5)*	69.5(1.8)	69.5(1.8)*
NB(Mul)	**82.0(2.4)**	**82.0(2.4)***	**81.7(2.5)**	**81.7(2.5)***
NB(Dir)	**81.4(2.7)**	78.0(3.3)	79.9(1.9)	74.1(4.2)
NB(Pol)	**82.2(2.1)**	**81.6(2.4)**	**82.2(2.3)**	**81.8(2.5)**

B. Learning Classifiers from OLNF RDF Data Fragments

The first set of experiments was designed to compare the performance of the proposed approaches to learning classifiers from an RDF data set that is fragmented (in OLNF) across multiple RDF stores with their centralized counterparts that have access to the entire data set in a single location. We trained two aggregation models and four generative models described in Sec. II-B: the *mode* aggregation coupled with a naive Bayes classifier (Mode+NB), the CCVD aggregation coupled with a logistic regression classifier (CCVD+LR), and the four naive Bayes generative models NB(Ber), NB(Mul), NB(Dir), and NB(Pol). Note that NB(Ber) and NB(Mul) need only leaf projections and therefore their models under OLNF is provably exact with respect to its centralized counterparts. The rest of the classifiers in the OLNF setting rely on the ART approximations of V_k^i and hence their performance is a function of the quality of the approximation. In this experiment, the termination threshold (difference between the true projection and its ART reconstruction) is set to 5% of the size of the matrix, and λ is set to 0.25.

The results in Table I show that: (i) not surprisingly, the performance of Mode+NB, CCVD+LR, NB(Dir), and NB(Pol) that rely on ART approximation of the needed statistics in the OLNF setting is always no better than that of their centralized counterparts which do not have to rely on the ART approximation and can instead use the statistics obtained directly from the entire data set; (ii) NB(Pol), despite its reliance on the ART approximation in the OLNF setting, shows performance that is competitive with its centralized counterpart although the latter has the advantage of using statistics obtained directly from the entire data set; and (iii) NB(Mul) surprisingly, is quite competitive with NB(Pol) in both centralized and OLNF settings despite using less information than NB(Pol).

C. Sensitivity of ART

The previous experiment used a fixed termination threshold for the iterative ART approximation procedure. Because the performance of the classifiers that rely on ART approximations of V_k^i is a function of the quality of the approximation which in turn depends on the the number of ART iterations, we designed an experiment to explore this dependence. In this set of experiments, we used 80% of the data for training, and 20% of the data for testing. First we measure the error of the reconstruction as estimated by (i) projection error which is the sum of absolute differences of each element between the true and reconstructed leaf and root projections; and (ii) matrix error which is the sum of absolute difference of each element between the true and reconstructed matrices. We also measure the classification accuracies of the trained models in the case of Mode+NB, CCVD+LR, NB(Dir), and NB(Pol) which make use of the ART approximation.

The results summarized in Fig. 6 show that the projection error approaches zero after a sufficiently large number of iterations; however, the matrix error remains relatively high even after 1000 iterations. In the case of classification accuracies, we note three clear trends shown in Fig. 7: (i) Mode+NB using the ART approximation does not quite approach Mode+NB that uses statistics obtained directly from the data regardless of the number of ART iterations; (ii) the performance of NB(Dir) lags that of NB(Pol) during the first few iterations of ART but both achieve comparable performance with increasing number of ART iterations; and (iii) CCVD+LR starts off with the worst performance but shows steady improvement with increasing number of ART iterations. However, theoretical underpinnings of these observations remain to be investigated.

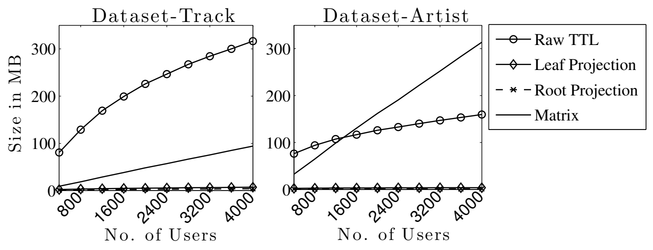

Figure 8. Communication complexities over the size of data sets (measured by number of Users).

D. Communication Complexity

The third experiment was designed to measure the communication cost of obtaining the projections required by the ART approximation. Since the size of query is negligible compared to the query results in our setting, we measure only the size of query results transferred, as the size of the underlying data set is varied. We used Dataset-Track and Dataset-Artist described in Sec. IV-A considering subsets of users ranging from 400 to 4000 in steps of 400, retaining in each case only the resources (tracks, artists, tags) that are connected to the subset of users. We recorded the size of raw RDF data in TTL format, the size of leaf projection,

[3]http://www.last.fm/

the size of root projection, and finally the size of matrix (stored as an adjacency list).

The results of this experiment summarized in Fig. 8 show that, not surprisingly, the size of raw data as well as matrices are significantly larger than the leaf and root projections, demonstrating the advantages of ART approximation in learning classifiers from large, OLNF-fragmented RDF data sets over alternative approaches that transmit the data or the matrix (as opposed to only the leaf and root projections) from the data source(s) to the learner. In the case of Dataset-Artist, we observe that the size of matrix even exceeds that of raw data, and this can be explained by the fact that a majority of artists are shared among (indirectly connected to) different users which blows up the size of the matrix, whereas in the RDF representation the artist resources and their tags only appear once in the data set.

V. Summary and Discussion

A. Summary

The emergence of many interlinked, physically distributed, and autonomously maintained RDF stores such as the LOD cloud offers unprecedented opportunities for predictive modeling and knowledge discovery from such data. However existing machine learning approaches are limited in their applicability because it is neither desirable nor feasible to gather all of the data in a centralized location for analysis due to access, memory, bandwidth, computational restrictions, and sometimes privacy or confidentiality constraints. Against this background we propose to learn classifiers from multiple interlinked RDF stores via their SPARQL query interfaces. Specifically we have: (i) introduced statistical query based formulations of several representative algorithms for learning classifiers from RDF data; (ii) introduced a distributed learning framework to learn classifiers from multiple interlinked RDF stores; (iii) identified three special cases of RDF data fragmentation and describe effective strategies for learning in each case; (iv) considered a novel application of a matrix reconstruction technique from the field of Computerized Tomography [1] to approximate the statistics needed by the learning algorithm from projections using count queries, thus dramatically reducing the amount of information transmitted from the remote data sources to the learner; and (v) reported results of experiments with a real-world social network data set, which demonstrate the feasibility of the proposed approach.

B. Related Work

Most of the existing work on learning predictive models from RDF data (e.g. [18], [19]) assume that the learner has direct access to data. Lin et al. [10] proposed an approach to learning relational Bayesian classifiers [11] from a remote RDF store in a setting where the learner can only query the RDF data store through a restricted class of statistical queries. This paper extends the work in [10] to the setting of multiple interlinked RDF stores using a larger class of predictive models including Mode+NB, CCVD+LR, NB(Dir) and NB(Pol) where, for practical reasons, we have to approximate the relevant statistics. Our approach takes advantage of SPARQL 1.1 update queries [4] and federated queries [4], which extends the remote access framework first introduced in [20] to multiple RDF stores. As opposed to federated query processing approaches for RDF data ([21], [22], [23]), which focus on the problem of answering queries formulated in a general purpose query language from multiple RDF data sources, our focus in this paper is on answering restricted classes of statistical queries needed for learning classifiers from RDF data. Restricting the classes of queries to those that useful in the learning predictive models from RDF data allows us to take advantage of optimizations such as the efficient accumulation of projections (Sec. III-B).

The work of [10] was inspired by a general learning framework proposed by Caragea et al. [24] for learning classifiers from distributed *tabular* data [25], [24]. However, to the best of our knowledge the approaches described in this paper are among the first of their kind for learning classifiers from an RDF data set that is fragmented across multiple interlinked RDF stores.

C. Future Work

ART, the method for reconstructing an approximation of a matrix from its projections is among the simplest such technique originally developed in the field of CT. Other promising matrix reconstruction methods worth exploring in our setting include filtered backprojection [1] and quadratic optimization [1]. The kinds of RDF data fragmentation considered in this paper are relatively simple, albeit interesting special cases. It would be interesting to consider learning classifiers in a setting where an RDF data set is fragmented across multiple RDF stores that are connected through more complex linkage patterns including in particular, trees and DAGs as opposed to the linear chains considered in this study. Lastly it is of interest to consider richer classes of predictive models and the corresponding learning problems, including those that model dependency between attributes (e.g. adaptations of statistical relational learning [26]), or feature construction strategies for linked data [27].

Acknowledgements

This work is supported in part by a grant (IIS 0711356) from the National Science Foundation (NSF) and in part by the Iowa State University Center for Computational Intelligence, Learning, and Discovery. The work of Vasant Honavar was supported by the NSF, while working at the Foundation. Any opinion, finding, and conclusions contained in this article are those of the authors and do not necessarily reflect the views of the NSF.

References

[1] G. T. Herman, *Fundamentals of Computerized Tomography: Image Reconstruction from Projections*, 2nd ed. Springer Publishing Company, Incorporated, 2009.

[2] T. Heath and C. Bizer, "Linked data: Evolving the web into a global data space," *Synthesis Lectures on the Semantic Web: Theory and Technology*, vol. 1, no. 1, pp. 1–136, 2011.

[3] F. Manola and E. Miller, Eds., *RDF Primer*, ser. W3C Recommendation. World Wide Web Consortium, February 2004. [Online]. Available: http://www.w3.org/TR/rdf-primer/

[4] W3C SPARQL Working Group, "SPARQL 1.1 overview." [Online]. Available: http://www.w3.org/TR/sparql11-overview/

[5] M. Hert, G. Reif, and H. C. Gall, "A comparison of RDB-to-RDF mapping languages," in *Proceedings of the 7th International Conference on Semantic Systems*, ser. I-Semantics '11. New York, NY, USA: ACM, 2011, pp. 25–32.

[6] R. Cyganiak and A. Jentzsch, "Linking open data cloud diagram," Online. http://lod-cloud.net/, September 2011.

[7] S. Bhagat, G. Cormode, and S. Muthukrishnan, "Node classification in social networks," in *Social Network Data Analytics*, C. C. Aggarwal, Ed. Springer US, 2011, pp. 115–148.

[8] C. C. Aggarwal and P. S. Yu, *Privacy-Preserving Data Mining: Models and Algorithms*, 1st ed. Springer Publishing Company, Incorporated, 2008.

[9] X. Wu, X. Ying, K. Liu, and L. Chen, "A survey of privacy-preservation of graphs and social networks," in *Managing and Mining Graph Data*, ser. The Kluwer International Series on Advances in Database Systems, C. C. Aggarwal, H. Wang, and A. K. Elmagarmid, Eds. Springer US, 2010, vol. 40, pp. 421–453.

[10] H. T. Lin, N. Koul, and V. Honavar, "Learning relational bayesian classifiers from RDF data," in *Proceedings of the 10th international conference on The semantic web*, vol. 1, 2011, pp. 389–404.

[11] J. Neville, D. Jensen, and B. Gallagher, "Simple estimators for relational bayesian classifiers," in *Proceedings of the Third IEEE International Conference on Data Mining*, 2003, pp. 609–612.

[12] C. Perlich and F. Provost, "Distribution-based aggregation for relational learning with identifier attributes," *Machine Learning*, vol. 62, no. 1-2, pp. 65–105, Feb. 2006.

[13] A. McCallum and K. Nigam, "A comparison of event models for naive bayes text classification," in *AAAI-98 Workshop on Learning for Text Categorization*. AAAI Press, 1998, pp. 41–48.

[14] T. Ferguson, "A bayesian analysis of some nonparametric problems," *The annals of statistics*, pp. 209–230, 1973.

[15] T. P. Minka, "Estimating a dirichlet distribution," Tech. Rep., 2012.

[16] R. E. Madsen, D. Kauchak, and C. Elkan, "Modeling word burstiness using the dirichlet distribution," in *Proceedings of the 22nd International Conference on Machine Learning*, ser. ICML '05. New York, NY, USA: ACM, 2005, pp. 545–552.

[17] R. Gordon, R. Bender, and G. T. Herman, "Algebraic reconstruction techniques (ART) for three-dimensional electron microscopy and x-ray photography," *Journal of Theoretical Biology*, vol. 29, no. 3, pp. 471–481, 1970.

[18] M. Nickel, V. Tresp, and H.-P. Kriegel, "Factorizing YAGO: scalable machine learning for linked data," in *Proceedings of the 21st international conference on World Wide Web*, ser. WWW '12. New York, NY, USA: ACM, 2012, pp. 271–280.

[19] U. Lösch, S. Bloehdorn, and A. Rettinger, "Graph kernels for RDF data," in *The Semantic Web: Research and Applications*, ser. Lecture Notes in Computer Science, E. Simperl, P. Cimiano, A. Polleres, O. Corcho, and V. Presutti, Eds. Springer Berlin Heidelberg, 2012, vol. 7295, pp. 134–148.

[20] L. Qi, H. T. Lin, and V. Honavar, "Clustering remote RDF data using SPARQL update queries," in *In Proceedings of the ICDE Workshop on Graph Data Management: Techniques and Applications*, 2013.

[21] C. Buil-Aranda, M. Arenas, O. Corcho, and A. Polleres, "Federating queries in SPARQL 1.1: Syntax, semantics and evaluation," *Web Semantics: Science, Services and Agents on the World Wide Web*, vol. 18, no. 1, pp. 1–17, 2013.

[22] A. Schwarte, P. Haase, K. Hose, R. Schenkel, and M. Schmidt, "Fedx: optimization techniques for federated query processing on linked data," in *Proceedings of the 10th international conference on The semantic web*, vol. 1, 2011, pp. 601–616.

[23] M. Karnstedt, K.-U. Sattler, and M. Hauswirth, "Scalable distributed indexing and query processing over linked data," *Web Semantics: Science, Services and Agents on the World Wide Web*, vol. 10, no. 0, pp. 3–32, 2012.

[24] D. Caragea, J. Zhang, J. Bao, J. Pathak, and V. Honavar, "Algorithms and software for collaborative discovery from autonomous, semantically heterogeneous, distributed information sources," in *Discovery Science*, ser. Lecture Notes in Computer Science, A. Hoffmann, H. Motoda, and T. Scheffer, Eds. Springer Berlin / Heidelberg, 2005, vol. 3735, pp. 13–44.

[25] D. Caragea, A. Silvescu, and V. Honavar, "A framework for learning from distributed data using sufficient statistics and its application to learning decision trees," *International Journal of Hybrid Intelligent Systems*, vol. 1, no. 1-2, pp. 80–89, 2004.

[26] L. Getoor and B. Taskar, *Introduction to Statistical Relational Learning*. The MIT Press, 2007.

[27] R. A. Rossi, L. K. McDowell, D. W. Aha, and J. Neville, "Transforming graph data for statistical relational learning," *J. Artif. Int. Res.*, vol. 45, no. 1, pp. 363–441, Sep. 2012.