# Learning Relational Bayesian Classifiers from RDF Data

Harris T. Lin *, Neeraj Koul **, and Vasant Honavar

Department of Computer Science
Iowa State University
Ames, IA 50011 USA
{htlin,neeraj,honavar}@iastate.edu

**Abstract.** The increasing availability of large RDF datasets offers an exciting opportunity to use such data to build predictive models using machine learning algorithms. However, the massive size and distributed nature of RDF data calls for approaches to learning from RDF data in a setting where the data can be accessed only through a query interface, e.g., the SPARQL endpoint of the RDF store. In applications where the data are subject to frequent updates, there is a need for algorithms that allow the predictive model to be incrementally updated in response to changes in the data. Furthermore, in some applications, the attributes that are relevant for specific prediction tasks are not known a priori and hence need to be discovered by the algorithm. We present an approach to learning Relational Bayesian Classifiers (RBCs) from RDF data that addresses such scenarios. Specifically, we show how to build RBCs from RDF data using statistical queries through the SPARQL endpoint of the RDF store. We compare the communication complexity of our algorithm with one that requires direct centralized access to the data and hence has to retrieve the entire RDF dataset from the remote location for processing. We establish the conditions under which the RBC models can be incrementally updated in response to addition or deletion of RDF data. We show how our approach can be extended to the setting where the attributes that are relevant for prediction are not known a priori, by selectively crawling the RDF data for attributes of interest. We provide open source implementation and evaluate the proposed approach on several large RDF datasets.

## 1 Introduction

The *Semantic Web* as envisioned by Berners-Lee, Hendler, and others [14, 3] aims to describe the semantics of Web content in a form that can be processed by computers [5, 2]. A key step in realizing this vision is to cast knowledge and data on the Web in a form that is conducive to processing by computers [15]. Resource Description Framework (RDF) ([23] for a primer) offers a formal language for

* Primary author
** Primary author, contributed equally

describing structured information on the Web. RDF represents data 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 (also called resources). RDF stores or triple stores and associated query languages such as SPARQL [26] offer the means to store and query large amounts of RDF data. Over the past decade, RDF has emerged as a basic representation format for the Semantic Web [15]. Cyganiak [8] estimated in 2010 that there are 207 RDF datasets containing over 28 billion triples published in the Linked Open Data cloud.

The increasing availability of large RDF datasets on the web offers unprecedented opportunities for extracting useful knowledge or predictive models from RDF data, and using the resulting models to guide decisions in a broad range of application domains. Hence, it is natural to consider the use of machine learning approaches, and in particular, statistical relational learning algorithms [11], to extract knowledge from RDF data [17, 4, 28]. However, existing approaches to learning predictive models from RDF data have significant shortcomings that limit their applicability in practice. Specifically, existing approaches rely on the learning algorithm having direct access to RDF data. However, in many settings, it may not be feasible to transfer data a massive RDF dataset from a remote location for local processing by the learning algorithm. Even in settings where it is feasible to provide the learning algorithm direct access to a local copy of an RDF dataset, algorithms that assume in-memory access to data cannot cope with RDF datasets that are too large to fit in memory. Hence, there is an urgent need for approaches to learning from RDF data in a setting where the data can be accessed only through a query interface, e.g., the SPARQL endpoint for the RDF store. In applications where the data are subject to frequent updates, there is a need for algorithms that allow the predictive model to be incrementally updated in response to changes in the data. Furthermore, in some applications, the attributes that are relevant for specific prediction tasks are not known a priori and hence need to be discovered by the algorithm. We present an approach to learning Relational Bayesian Classifiers from RDF data that addresses such scenarios.

Our approach to learning Relational Bayesian Classifiers (RBCs) from RDF data adopts the general framework introduced by Caragea et al. [6] for transforming a broad class of standard learning algorithms that assume in memory access to a dataset into algorithms that interact with the data source(s) only through statistical queries or procedures that can be executed on the remote data sources. This involves decomposing the learning algorithm into two parts: (i) a component that poses the relevant statistical queries to a data source to acquire the information needed by the learner; and (ii) a component that uses the resulting statistics to update or refine a partial model (and if necessary, further invoke the statistical query component). This approach has been previously used to learn a variety of classifiers from relational databases [20] using SQL queries and from biomolecular sequence data [19]. It has recently become feasible to use a similar approach to learning RBCs from RDF data due to the incorporation

of support for aggregate queries in SPARQL. (SPARQL 1.1 supports aggregate queries whereas SPARQL 1.0 does not).

We show how to learn RBCs from RDF data using only aggregate queries through the SPARQL endpoint of the RDF store. This approach does not require in-memory access to RDF data to be processed by the learning algorithm, and hence can scale up to very large data sets. Because the predictive model is built using aggregate queries against a SPARQL endpoint, it can be used to learn RBCs from large remote RDF stores without having to transfer the data to a local RDF store for processing (in general, the cost of retrieving the statistics needed for learning is much lower than the cost of retrieving the entire dataset). Under certain conditions which we identify in the paper, we show how the RBC models can be incrementally updated in response to changes (addition or deletion of triples) from the RDF store. We further show how our approach can be extended to the setting where the attributes that are relevant for prediction are not known a priori, by selectively crawling the RDF data for attributes of interest. We have implemented the proposed approach into INDUS [21], an open source suite of learning algorithms, that learn from massive data sets only using statistical queries. We describe results of experiments on several large RDF datasets that demonstrate the feasibility of the proposed approach to learning RBCs from RDF stores.

The rest of the paper is organized as follows: Section 2 introduces a precise formulation of the problem of learning RBCs from RDF data. Section 3 describes how to build RBCs from RDF data using only aggregate queries. Section 4 identifies the conditions under which it is possible to incrementally update an RBC learned model from an RDF store in response to updates to the underlying RDF store. Section 5 presents an analysis of the communication complexity of learning RBCs from RDF stores. Section 6 describes how to extend to the setting where the attributes that are relevant for prediction are not known a priori, by selectively crawling the RDF data for attributes of interest. Section 7 describes results of experiments with several RDF datasets that demonstrate the feasibility proposed approach. Finally Sec. 8 concludes with a summary and a brief discussion of related work.

## 2 Problem Formulation

In this section we formulate the problem of learning predictive models from RDF data. Assume there are pairwise disjoint infinite sets $I$, $B$, $L$ and $V$ (IRIs, Blank nodes, Literals and Variables respectively). A triple $(s, p, o) \in (I \cup B) \times I \times (I \cup B \cup L)$ is called an RDF triple. In this triple, $s$ is the subject, $p$ the predicate, and $o$ the object. An RDF graph is a set of RDF triples.

As a running example for the following definitions, we consider the RDF schema for the movie domain as shown in Fig. 1. We wish to predict whether a movie receives more than $2M in its opening week.

**Definition 1 (Target Class)** *Given an RDF graph $\mathcal{G}$, a* target class *is a distinguished IRI of type* rdfs:Class *in $\mathcal{G}$. For example, Movie.*

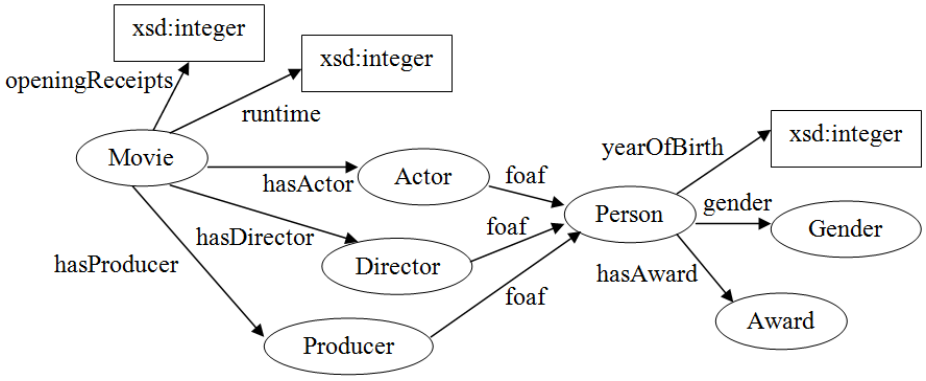


**Fig. 1.** RDF schema for the movie domain

**Definition 2 (Instances)** *Given an RDF graph $\mathcal{G}$ and a target class $\mathcal{T}$, the instances of $\mathcal{T}$, denoted $\mathcal{T}(\mathcal{G})$ is the set $\{x : (x, \text{rdf:type}, \mathcal{T}) \in \mathcal{G}\}$.*

**Definition 3 (Attribute)** *Given an RDF graph $\mathcal{G}$ and a target class $\mathcal{T}$, an* attribute *$A$ (of a target class $\mathcal{T}$) is a tuple of IRIs $(p_1, \ldots, p_n)$ such that the domain of $p_1$ is $\mathcal{T}$, the range of $p_i$ is the domain of $p_{i+1}$, and the range of $p_n$ is a literal. For example, (hasActor, foaf, yearOfBirth). We also refer the range of the attribute $A$ as the range of $p_n$.*

**Definition 4 (Attribute Graph)** *Given an instance $x$ of the target class $\mathcal{T}$ in the RDF graph $\mathcal{G}$ and an attribute $A = (p_1, \ldots, p_n)$, the attribute graph of the instance $x$, denoted by $A(x)$, is the union of the sets of triples that match the Basic Graph Pattern [26]*

$$((x, p_1, ?v_1)\ AND\ (?v_1, p_2, ?v_2)\ AND\ \ldots\ AND\ (?v_{n-1}, p_n, ?v_n)) \qquad (1)$$

*where $v_i \in V$ are variables.*

*Given an additional literal value $a$, we also define a filtered attributed graph, denoted $A(x, a)$, which includes the filter constraint FILTER$(?v_n = a)$ in the graph pattern (1). Further, if $\mathcal{A}$ is a tuple of attributes $(A_1, \ldots, A_n)$, then we define $\mathcal{A}(x)$ to be $(A_1(x), \ldots, A_n(x))$*

**Definition 5 (Target Attribute)** *Given an RDF graph $\mathcal{G}$ and a target class $\mathcal{T}$, a* target attribute *is a distinguished attribute denoted by $C$. For example, (openingReceipts).*

$C(x)$ is intended to describe the *class label* of the instance $x$, hence we assume that each instance has exactly one class label, i.e., $|C(x)| = 1$ for every $x \in \mathcal{T}(\mathcal{G})$. Given a target attribute $C = (p_1, \ldots, p_n)$, we define $v(C, x)$ to be the value of $?v_n$ matched by the graph pattern (1).

**Definition 6 (Class Label)** *Given a* target attribute $C = (p_1, \ldots, p_n)$*, the set of* class labels *is the the range of* $p_n$*. For brevity we denote this set by* $\mathcal{C}$*.*

**Definition 7 (RDF Dataset)** *An* RDF dataset $D$ *is a tuple* $(\mathcal{G}, \mathcal{T}, \mathcal{A}, C)$ *where* $\mathcal{G}$ *is an RDF graph,* $\mathcal{T}$ *a target class in* $\mathcal{G}$*,* $\mathcal{A}$ *a tuple of attributes, and* $C$ *is a target attribute. We also denote the tuple* $(\mathcal{T}, \mathcal{A}, C)$ *as* $Desc(D)$ *corresponding to the descriptor of the dataset.*

**Definition 8 (Induced Attribute Graph Dataset)** *Given an RDF dataset* $D = (\mathcal{G}, \mathcal{T}, \mathcal{A}, C)$*, its* induced attribute graph dataset*, denoted* $I(D)$*, is defined as* $\{(\mathcal{A}(x), v(C, x)) : \ x \in \mathcal{T}(\mathcal{G})\}$*.*

We now formalize the the problem of learning from RDF data.

*Problem 1.* Given an RDF dataset $D = (\mathcal{G}, \mathcal{T}, \mathcal{A}, C)$ and its induced attribute graph dataset $I(D)$, a hypothesis class $H$, and a performance criterion $P$, the learning algorithm $L$ outputs a classifier $h \in H$ that optimizes $P$. The input to the classifier $h$ is $\mathcal{A}(x)$ where $x$ is an instance of a target class $\mathcal{T}$, and the output $h(x) \in \mathcal{C}$ is a class label.

## 3 Learning from RDF data

We reduce the problem of learning from RDF data to the problem of learning from multiset attribute data which is defined below. This reduction allows for application of algorithms for learning from multiset attribute data (e.g. Relational Bayesian Classifier [25]) to this setting. Given an RDF dataset $D = (\mathcal{G}, \mathcal{T}, \mathcal{A}, C)$ and its induced attribute graph dataset $I(D)$, consider an attribute $A$ and the attribute graph $A(x)$ of an instance $x \in \mathcal{T}(\mathcal{G})$. The attribute graph $A(x)$ can be viewed as a directed acyclic graph (DAG) rooted in $x$, and here we are interested in only the leaves of this DAG. The following definition captures this notion.

**Definition 9 (Leaf)** *Given an attribute* $A_i$*, we define the leaf function* $\mathcal{L}(A_i(x))$ *that returns the multiset of leaves of* $A_i(x)$*, such that each leaf* $a \in A_i(x)$ *is replaced with* $n$ *copies of* $a$ *where* $n$ *is the number of unique paths from* $x$ *to* $a$*. For brevity we write* $\mathcal{L}(A_i(x))$ *as* $\mathcal{L}_i(x)$ *and* $\mathcal{L}(A_i(x, a))$ *as* $\mathcal{L}_i(x, a)$*.*

*Also, we overload the leaf function on a tuple of attributes* $\mathcal{A} = (A_1, \ldots, A_n)$ *by* $\mathcal{L}(\mathcal{A}(x)) = (\mathcal{L}_1(x), \ldots, \mathcal{L}_n(x))$*.*

Using the leaf function, we reduce $I(D)$ into a multiset attributed dataset $\mathcal{M}(D) = \{(\mathcal{L}(\mathcal{A}(x)), v(C, x)) : \ x \in \mathcal{T}(\mathcal{G})\}$. To learn from $\mathcal{M}(D)$ we focus our attention on Relational Bayesian Classifiers (RBC) motivated from modeling relational data [25]. RBC assumes that attribute multisets are independent given the class, and the most probable class of an instance is given by:

$$h_{RBC}(x) = \underset{c \in \mathcal{C}}{\operatorname{argmax}}\, p(c) \prod_i p(\mathcal{L}_i(x) : \ c) \tag{2}$$

Several methods to estimate the probabilities $p(\mathcal{L}_i(x) : \ c)$ are described in [25]:

– Aggregation: $\hat{p}_{\text{agg}}(\mathcal{L}_i(x) :\ c) = \hat{p}(\text{agg}(\mathcal{L}_i(x)) :\ c)$, where agg is an aggregation function such as min, max, average for continuous attributes; and mode for discrete attributes.
– Independent Value: $\hat{p}_{\text{ind}}(\mathcal{L}_i(x) :\ c) = \prod_{a \in \mathcal{L}_i(x)} \hat{p}(a :\ c)$, which assumes each value in the multiset is independently drawn from the same distribution (attribute value independence).
– Average Probability: $\hat{p}_{\text{avg}}(\mathcal{L}_i(x) :\ c) = \frac{\sum_{a \in \mathcal{L}_i(x)} \hat{p}(a:\ c)}{|\mathcal{L}_i(x)|}$, which also assumes attribute value independence as in *Independent Value*, however during inference the probabilities are averaged instead of multiplied.

For estimating the parameters in (2), we assume that the learner does not have access to the RDF graph $\mathcal{G}$ but instead only has knowledge $\mathcal{T}, \mathcal{A}$, and $C$. In addition, we assume that the RDF store answers statistical queries over the RDF graph $\mathcal{G}$ which in our setting correspond to aggregate SPARQL queries submitted to a SPARQL endpoint. Given a descriptor $Desc(D) = (\mathcal{T}, \mathcal{A}, C)$ where $C = (c_1, \ldots, c_m)$ we assume that the RDF store supports the following type of primitive queries:

(Q1) $S(\mathcal{G}, \mathcal{T}) = |\mathcal{T}(\mathcal{G})|$, the number of instances of target type $\mathcal{T}$ in $\mathcal{G}$. This corresponds to the SPARQL query:

```
SELECT COUNT(*) WHERE { ?x rdf:type <T> . }
```

(Q2) $S(\mathcal{G}, \mathcal{T}, C = c) = |\{x \in \mathcal{T}(\mathcal{G}) :\ v(C, x) = c\}|$, the number of instances of target type $\mathcal{T}$ in which the target attribute takes the class label $c$. This corresponds to the SPARQL query:

```
SELECT COUNT(*) WHERE {
  ?x rdf:type <T> .
  ?x <c1> ?c1 .  ...  ?cm-1 <cm> c .
}
```

(Q3) $S(\mathcal{G}, \mathcal{T}, C = c, A_i) = \sum_{x \in \mathcal{T}(\mathcal{G}) \text{ and } v(C,x)=c} |\mathcal{L}_i(x)|$. Assuming the attribute $A_i = (p_1, \ldots, p_j)$ this corresponds to the SPARQL query:

```
SELECT COUNT(*) WHERE {
  ?x rdf:type <T> .
  ?x <c1> ?c1 .  ...  ?cm-1 <cm> c .
  ?x <p1> ?v1 .  ...  ?vj-1 <pj> ?vj .
}
```

(Q4) $S(\mathcal{G}, \mathcal{T}, C = c, A_i = a) = \sum_{x \in \mathcal{T}(\mathcal{G}) \text{ and } v(C,x)=c} |\mathcal{L}_i(x, a)|$. Assuming the attribute $A_i = (p_1, \ldots, p_j)$ this corresponds to the SPARQL query:

```
SELECT COUNT(*) WHERE {
  ?x rdf:type <C> .
  ?x <c1> ?c1 .  ...  ?cm-1 <cm> c .
  ?x <p1> ?v1 .  ...  ?vj-1 <pj> a .
}
```

(Q5) $S(\mathcal{G}, \mathcal{T}, C = c, A_i, \text{agg}, [v_l, v_h])$. Given a range $[v_l, v_h]$ this corresponds to the SPARQL query:

```
SELECT COUNT(*) WHERE {
  { SELECT (agg(?vj) AS ?aggvalue) WHERE {
      ?x rdf:type <T> .
      ?x <c1> ?c1 .  ...  ?cm-1 <cm> c .
      OPTIONAL { ?x <p1> ?v1 .  ...  ?vj-1 <pj> ?vj . }
    } GROUP BY ?x
  } FILTER(?aggvalue >= vl && ?aggvalue <= vh)
}
```

We now proceed to describe how an RBC can be built using the supported SPARQL queries without requiring access to the underlying dataset. The RBC estimates the following probabilities from training data:

1. $\hat{p}(c)$
2. $\hat{p}(\text{agg}(A_i) :\ c)$ for each attribute $A_i$ where aggregation is used to estimate probabilities. For simplicity, we discretize the aggregated values and predetermine the bins prior to learning. Hence, we estimate $\hat{p}(\text{agg}(A_i) \in [v_l, v_h] :\ c)$ for each bin $[v_l, v_h]$
3. $\hat{p}(a :\ c)$ where $a$ is in the range of $A_i$, for each attribute $A_i$ where independent value or average probability is used to estimate the probabilities.

The above three probabilities can be estimated (using Laplace correction for smoothing) as follows:

1. $\hat{p}(c) = \frac{S(\mathcal{G},\mathcal{T},c)+1}{S(\mathcal{G},\mathcal{T})+m}$ where $m$ is the number of class labels
2. $\hat{p}(\text{agg}(A_i) \in [v_l, v_h] :\ c) = \frac{S(\mathcal{G},\mathcal{T},C=c,A_i,\text{agg},[v_l,v_h])+1}{S(\mathcal{G},\mathcal{T},c)+m}$ where $m$ is the number of bins (ranges)
3. $\hat{p}(a :\ c) = \frac{S(\mathcal{G},\mathcal{T},C=c,A_i=a)+1}{S(\mathcal{G},\mathcal{T},C=c,A_i)+m}$ where $a$ is in the range of $A_i$ and $m$ is the size of range of $A_i$

Hence, it is possible to learn RBCs from an RDF graph by interacting with the RDF store only through SPARQL queries. This approach does not require access to the underlying dataset and in most practical settings requires much less bandwidth as compared to transferring the data to a local store for processing (see Sec. 5).

## 4 Updatable Models

In many settings, the RDF store undergoes frequent updates i.e., addition or deletion of sets of RDF triples. In such settings, it is necessary to update the predictive model to reflect the changes in the RDF store used to build the model. While in principle, the algorithm introduced in Sec. 3 can be re-executed each time there is an update to the RDF store, it is of interest to explore more efficient solutions for incrementally updating the RBC model by updating only the relevant statistics.

Given a dataset $D$ and a learning algorithm $L$, let $L(D)$ be a predictive model built from the dataset $D$. Let $\theta$ be a primitive query required over the dataset $D$ to build $L(D)$.

**Definition 10 (Updatable Model [19])** *Given datasets $D_1$ and $D_2$ such that $D_1 \subseteq D_2$, we say that a primitive query $\theta$ is updatable iff we can specify functions $f$ and $g$ such that:*

1. $\theta(D_2) = f(\theta(D_2 - D_1), \theta(D_1))$
2. $\theta(D_1) = g(\theta(D_2), \theta(D_2 - D_1))$

*We say that the predictive model constructed using $L$ is updatable iff all primitive queries required over the dataset $D$ to build $L(D)$ are updatable.*

The following propositions show that the primitive query $(Q1)$ of the RBC model is updatable, whereas the rest of the queries are not updatable. Hence, in general, the RBC model is not updatable.

**Proposition 1.** *The primitive query $S(\mathcal{G}, \mathcal{T})$ is updatable.*

*Proof.* This query counts the number of instances of target type $\mathcal{T}$ in $\mathcal{G}$, which is the cardinality of $\{x : (x, \text{rdf:type}, \mathcal{T}) \in \mathcal{G}\}$. Since $\mathcal{G}_1 \subseteq \mathcal{G}_2$ we have $S(\mathcal{G}_2, \mathcal{T}) = S(\mathcal{G}_2 - \mathcal{G}_1, \mathcal{T}) + S(\mathcal{G}_1, \mathcal{T})$, and also $S(\mathcal{G}_1, \mathcal{T}) = S(\mathcal{G}_2, \mathcal{T}) - S(\mathcal{G}_2 - \mathcal{G}_1, \mathcal{T})$.

**Proposition 2.** *The primitive query $S(\mathcal{G}, \mathcal{T}, C = c, A = a)$ is not updatable.*

*Proof.* We prove by showing a counter example. Let the target class be $\mathcal{T}$, the target attribute be $C = (c_1)$, an attribute $A = (p_1, \ldots, p_i, \ldots, p_n)$, and suppose we have the following RDF graphs: $S_1 = \{(x, \text{rdf:type}, \mathcal{T}), (x, c_1, c)\}$, $S_2 = \{(x, p_1, o_1), \ldots, (o_{i-1}, p_i, o_i)\}$, and $S_3 = \{(o_i, p_{i+1}, o_{i+1}), \ldots, (o_{n-1}, p_n, a)\}$. Suppose the graph before update is $\mathcal{G}_1 = S_1 \cup S_2$ and after an insertion of $S_3$ the graph becomes $\mathcal{G}_2 = S_1 \cup S_2 \cup S_3$. For brevity let $\theta(\mathcal{G}) = S(\mathcal{G}, \mathcal{T}, C = c, A = a)$. We will show that there exists no functions $f$ for the query $\theta(\mathcal{G}_2)$, which counts the total number of leaves of $A(x, a)$ such that $x$ has the class label $c$. In $\mathcal{G}_2$ the attribute graph $A(x)$ is $S_2 \cup S_3$, and hence $\theta(\mathcal{G}_2) = 1$. However, $A(x)$ is partitioned among $S_2$ and $S_3$, so $\theta(\mathcal{G}_1) = 0$ and $\theta(\mathcal{G}_2 - \mathcal{G}_1) = 0$, therefore in this case $f(\theta(\mathcal{G}_2 - \mathcal{G}_1), \theta(\mathcal{G}_1)) = f(0, 0) = 1 = \theta(\mathcal{G}_2)$. Now consider another case where initially the graph is $\mathcal{G}_3 = S_1$ and after insertion of $S_3$ the graph becomes $\mathcal{G}_4 = S_1 \cup S_3$. In this case we have $\theta(\mathcal{G}_4) = 0$, $\theta(\mathcal{G}_3) = 0$, and $\theta(\mathcal{G}_4 - \mathcal{G}_3) = 0$, and so $f(0, 0) = 0$. Since a function can not map an input to more than one output, this shows that there exists no function $f$ to maintain the query result.

Similarly, can show that the primitive queries $S(\mathcal{G}, \mathcal{T}, C = c)$, $S(\mathcal{G}, \mathcal{T}, C = c, A)$, and $S(\mathcal{G}, \mathcal{T}, C = c, A, \text{agg}, [v_l, v_h])$ are not updatable.

**Corollary 1.** *RBC model is not updatable.*

The proof of Prop. 2 shows that when an attribute graph is partitioned across multiple updates, there exists no function to update the required counts. This raises the question as to whether we can ensure updatability by requiring that each update involves only complete attribute graphs. However, this requirement is not sufficient for the query to be updatable. To see why, consider $\mathcal{G}_1 = \{(x, \text{rdf:type}, \mathcal{T}), (x, c_1, c), (x, p_1, o), (o, p_2, a)\}$ and $\mathcal{G}_2 - \mathcal{G}_1 =$

$\{(y, \text{rdf:type}, \mathcal{T}), (y, c_1, c), (y, p_1, o), (o, p_2, b)\}$, then $f(0,1) = 2$. The extra count from $\theta(\mathcal{G}_2)$ is due to $o$ being shared between two datasets despite the fact that each attribute graph is complete. This motivates the restriction of not allowing the update to *reuse* certain subjects or objects. We formalize this notion as follows.

**Definition 11 (Clean Update)** *Assume $\mathcal{G}_1 \subseteq \mathcal{G}_2$, and let $V(\mathcal{G}) = \{s : (s,p,o) \in \mathcal{G}\} \cup \{o : (s,p,o) \in \mathcal{G}\}$ denote the set of all subjects and objects of an RDF graph $\mathcal{G}$. An update (from $\mathcal{G}_1$ to $\mathcal{G}_2$ by insertion, or from $\mathcal{G}_2$ to $\mathcal{G}_1$ by deletion) is said to be clean if $[\forall (s,p,o) \in \mathcal{G}_2][s \notin V(\mathcal{G}_1) \cap V(\mathcal{G}_2 - \mathcal{G}_1)]$. That is, triples in $\mathcal{G}_2 - \mathcal{G}_1$ share objects with only the leaves of attribute graphs in $\mathcal{G}_1$.*

**Proposition 3.** *RBC models are updatable if every update is clean.*

*Proof.* Let $D_1$ and $D_2$ be two RDF datasets such that $D_1 \subseteq D_2$. We first consider the primitive query $\theta(\mathcal{G}) = S(\mathcal{G}, \mathcal{T}, C = c, A = a)$. Since every update is clean, the attribute graphs $A(x)$ for all attributes in $A$, and all instances $x \in \mathcal{T}(\mathcal{G}_1)$ and $x \in \mathcal{T}(\mathcal{G}_2 - \mathcal{G}_1)$ remain the same after insertion (or deletion). Hence, $\mathcal{M}(D_2) = \mathcal{M}(D_1) \cup \mathcal{M}(D_2 - D_1)$ and similarly $\mathcal{M}(D_1) = \mathcal{M}(D_2) - \mathcal{M}(D_2 - D_1)$ for the multiset attributed dataset reductions. It follows that $\theta(\mathcal{G}_2) = \theta(\mathcal{G}_1) + \theta(\mathcal{G}_2 - \mathcal{G}_1)$ and $\theta(\mathcal{G}_1) = \theta(\mathcal{G}_2) - \theta(\mathcal{G}_2 - \mathcal{G}_1)$. Similar argument also holds true for the other queries used for learning a RBC.

Thus, RBC model can be updated incrementally in a restricted setting where every update is clean in the sense defined above. When clean updates are not available, RBC models can still be incrementally updated if we are willing to sacrifice some accuracy; and rebuild the model periodically by querying the entire RDF store, with the frequency of rebuild chosen based on the desired tradeoff between computational efficiency and model accuracy. Regardless of whether the RBC model is updatable or not, answering of aggregate queries from RDF stores answering can be optimized using an aggregate view maintenance algorithm [16]. Since we assume that the data descriptor does not change as frequently as the data, the aggregate queries needed by the RBC model can be set up and maintained as views on the RDF store.

## 5 Communication Complexity

In this section, we analyze the communication complexity, i.e., the amount of data transfer needed to build an RBC model. We compare the communication complexity of building an RBC model from RDF data in the following two scenarios: (i) posing statistical queries needed for learning the model against a remote RDF store which is the approach proposed in this paper; and (ii) retrieving the entire RDF dataset from a remote RDF store for local processing.

Given an RDF dataset $D = (\mathcal{G}, \mathcal{T}, \mathcal{A}, C)$ where $\mathcal{A} = (A_1, \ldots, A_n)$. Suppose the RDF store holds the RDF graph $\mathcal{G}$, and let $|\mathcal{G}|$ denotes the size of this graph. The communication complexity in scenario (ii) is simply $O(|\mathcal{G}|)$. We now analyze

the communication complexity in scenario (i). Let $l_C$ denotes the length of tuple $C$, let $r_C$ denotes the size of range of $C$, and let $l_A$ denotes the maximum length of an attribute tuple. Also let $r_A^1$ denotes the maximum number of bins of those attributes estimated by aggregation, let $r_A^2$ denotes the maximum size of range of the remaining attributes, and we define $r_A$ to be $\max(r_A^1, r_A^2)$.

The size of query expressed in SPARQL, is $O(1)$ for $(Q1)$, $O(l_C)$ for $(Q2)$, and $O(l_C + l_A)$ for $(Q3)$, $(Q4)$, and $(Q5)$. Further, to estimate the probabilities to build an RBC, the following number of calls for each query described in Sec. 3 are required:

(Q1) one.
(Q2) $r_C$, once for each class label.
(Q3) $r_C \cdot n$, once for each class label and each attribute.
(Q4) $O(r_C \cdot n \cdot r_A)$, once for each class label, each attribute, and each value of the attribute.
(Q5) $O(r_C \cdot n \cdot r_A)$, same as $(Q4)$.

Therefore, the total complexity is $O(1) + O(l_C r_C) + O((l_C + l_A) r_C n) + O((l_C + l_A) r_C \cdot n \cdot r_A) + O((l_C + l_A) r_C \cdot n \cdot r_A) = O((l_C + l_A) r_C \cdot n \cdot r_A)$. In Sec. 7.1 we provide results of experiments which show that $O((l_C + l_A) r_C \cdot n \cdot r_A)$ is usually less than $O(|\mathcal{G}|)$ in practice.

## 6 Selective Attribute Crawling

In previous sections we have considered the problem of learning RBCs given an RDF dataset $D = (\mathcal{G}, \mathcal{T}, \mathcal{A}, C)$ in the setting where the learner has direct access to $\mathcal{T}, \mathcal{A}$, and $C$, but not $\mathcal{G}$. Here we consider a more general problem where the learner does not have a priori knowledge of $\mathcal{A}$. This requires the learner to interact with the RDF store containing $\mathcal{G}$ in order to determine $\mathcal{A}$ (e.g. by crawling and selecting attributes) that best optimizes a predetermined performance criterion $P$. Since the number of attributes in an RDF store can be arbitrarily large we specify an additional constraint $Z$ to guarantee termination (e.g. number of attributes crawled, number of queries posed, time spent, etc.).

*Problem 2.* Given an RDF dataset without attributes, $D = (\mathcal{G}, \mathcal{T}, C)$, a hypothesis class $H$, a performance criterion $P$, and constraint $Z$, the learning algorithm $L$ outputs the following while respecting $Z$: (i) The selected tuple of attributes $\mathcal{A}$, and (ii) a classifier $h \in H$ that optimizes $P$.

For simplicity, we focus the setting where the constraint $Z$ specifies the maximum the number of attributes crawled. We consider the problem of identifying $\mathcal{A}$ of cardinality at most $Z$ so as to optimize $P$. This problem is a variant of the well-studied feature subset selection problem [22, 12], albeit in a setting where the set of features is a priori unknown. Identifying attributes one at a time to optimize $P$ can be seen as a search over a tree rooted at $\mathcal{T}$, where the edges are IRIs of properties and the nodes are the domain/range of properties, and an

attribute corresponds to a path from the root to an RDF literal (a leaf in this tree). To complete the specification of the search problem, we need to specify operations for expanding a node to generate its successors and define the scoring function for evaluating nodes. Expanding a node consists of querying (i) the set of distinct properties outgoing from a node, (ii) the range of each property, and (iii) the type of each range (e.g. numeric, string, non-literal), each of which can be expressed as SPARQL queries. We define the score of a node based on the degree of correlation of the node with the target attribute $C$. Specifically, for each attribute (represented by a leaf), we compute mutual information [7] between it and the the target attribute $C$. The score of an internal node is defined (recursively) as a function of its descendants, e.g. average of the scores of its children.

Formally, the score of an attribute $A$ is:

$$Score(A) = \sum_{C=c, A=a} p(A=a, C=c) \log_2 \frac{p(A=a, C=c)}{p(A=a)p(C=c)} \tag{3}$$

These probabilities can also be estimated based on the queries described in Sec. 3. Given this framework, a variety of alternative search strategies can be considered, along with several alternative scoring functions.

## 7 Experiments

We conduct three experiments each with a different goal. The first measures the communication complexity using the LinkedMDB [13] dataset. The second experiment combines the US Census dataset with a government dataset to evaluate the accuracy of models using different attribute crawling strategies. Finally we demonstrate learning of RBC from another government dataset through a live SPARQL endpoint.

### 7.1 Communication Complexity Experiment

The goal of this experiment is to measure the communication complexity under two different approaches described in Sec. 5.

**Dataset and Experiment Setup** The IMDB dataset is a standard benchmark that has been used to evaluate probabilistic relational models including RBCs [25]. The task is to predict whether a movie receives more than $2M in its opening week. We used LinkedMDB [13], which is an RDF store extracted from IMDB, with links to other datasets on the Linked Open Data cloud [8]. We used links to Freebase[1] which includes $foaf$ property to the $Person$ class and three properties of class $Person$. Fig. 1 shows the RDF schema of the extracted dataset. Since LinkedMDB does not have openingReceipts, we add

[1] `http://www.freebase.com`

them by crawling the IMDB website[2]; also for the Freebase data, we parse the yearOfBirth property for each Person from the dateOfBirth property. We extract 20 movies which are released after 2006 such that each movie has at least one actor, one director, and one producer. The target class is $Movie$ and the target attribute is $(openingReceipts)$. We consider a total of 10 attributes: $(runtime)$, and $(h, foaf, a)$ where $h \in \{hasActor, hasDirector, hasProducer\}$ and $a \in \{yearOfBirth, gender, hasAward\}$.

To show the growth of data transfer, we prepared 20 subsets of the dataset by corresponding to 1 to 20 movies. A movie instance consists of the URI of the movie and all reachable linked data for it. For communication complexity of learning RBC from RDF stores using statistical queries, we used the proposed approach to build an RBC for each subset and logged the SPARQL queries sent, saved the log in a plain text format, and measured the size of the logs. We compared the results with the communication complexity of learning RBC by first retrieving the data from a remote store for local processing as measured by the size of the corresponding dataset in RDF/XML format on disk.

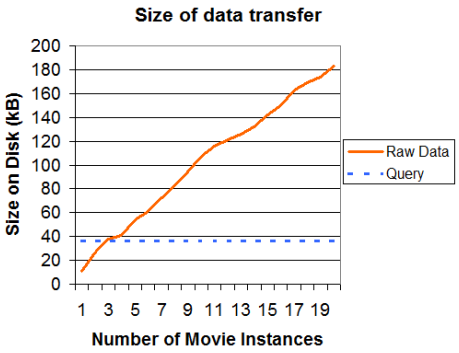


**Fig. 2.** Comparison of size of data transfer for Experiment 7.1

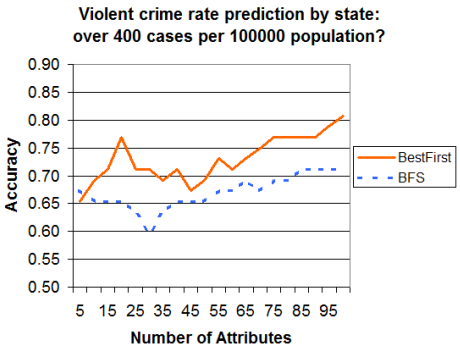


**Fig. 3.** Comparison of two crawling strategies for Experiment 7.2

**Results** Figure 2 shows that the size of the raw data exceeds that of the query when there are more than three movie instances in the dataset. We also considered the case where the RDF store compresses the raw data before transfer, and in this case the size of the compressed raw data exceeds that of the query when there are more than 90 movie instances.

[2] `http://www.imdb.com`

### 7.2 Selective Attribute Crawling Experiment

The goal of this experiment is to evaluate the accuracy of RBC models built using different attribute crawling strategies. Recall that in this setting the learner is only given a SPARQL endpoint of the RDF store, the target class, and the target attribute.

**Dataset and Experiment Setup** In this experiment we use datasets from Data.gov and US census 2000. The target class is 52 US states and we wish to predict whether a state's violent crime rate is over 400 per 100,000 population, which is from dataset 311 of the Data-gov project [9]. We link this with the US Census 2000 dataset for the corresponding states. This dataset was converted to over 1 billion RDF triples by [27]. Part of its RDF schema is shown in [27]. It uses a property as a way to sub-divide the population, and a number at a leaf represents the population that satisfies the conditions (properties) on the path from root. In our experiment we normalize by dividing every number of a state by the state's total population. We vary the maximum number of attributes to be crawled. We set the constraint to be the number of attributes the learner is allowed to crawl. We apply two different attribute crawling strategies (described below) separately and build RBC models using the crawled attributes. To measure the accuracy of the built models, we randomly partition 52 states into 13 groups (of 4 states each) and perform cross validation. That is, for each group, the 4 states in the group are held out and used for prediction, and the remaining are used for training the model. The overall accuracy is the total number of correct predictions divided by the number of states (52).

We experiment with two crawling strategies: BreadthFirst (BFS) and BestFirst. BFS chooses the node with the least depth to expand and BestFirst chooses the node that has the highest score as defined in Sec. 6.

**Results** As shown in Fig. 3, BestFirst outperforms BFS with the exception of the case where the number of attributes is 5. We examined the crawled attributes for BestFirst from for choices of $Z$ from 20 to 45, and found that the strategy focused on expanding the households property. This is because attribute selection is guided by mutual information between a candidate attribute and the target class. The sub-divisions of this property may provide very minimal additional information compared to the first one crawled in this group, and hence they may not contribute to the predictive accuracy. One way to circumvent this problem is to use a scoring function to that measures the amount of *information gain* resulting from a candidate attribute given all the attributes that have already been chosen. Another approach is to penalize the attributes based on the depth of search. A third approach is to use the marginal improvement in the accuracy of the RBC classifier resulting from inclusion of the attribute to decide whether to retain it. Other alternatives worth exploring include different search strategies such as Iterative Deepening Search (IDS) [18].

### 7.3 Live Demonstration

The goal of this experiment is to demonstrate learning of RBC from a government dataset through a live SPARQL endpoint[3] hosted on Rensselaer Polytechnic Institute [10]. This endpoint supports aggregate and nested queries proposed in SPARQL 1.1.

**Dataset and Experiment Setup** We used a Health Information National Trends Survey (HINTS) [24] from NCI which has been converted into RDF as part of the Data-gov project [9]. The survey represents a cross-sectional study of health media use and cancer-related knowledge among adults in the United States, and it has been used by [1] to study associations of covariates with different smoking statuses. There are 12080 participants across two years (2003 and 2005), represented by 623544 total number of RDF triples, and the raw RDF data (as TTL dump) has a size 35.9MB on disk. The task in our setting is to predict the smoking status (never, former, or current) of a participant from 16 other attributes such as race, sex, household income, and education. The dataset is propositional in nature although represented in RDF format; that is, every attribute has exactly one value in terms of the reduced multiset attribute data, hence the task reduces to learning of a conventional Naive Bayes classifier. Nevertheless, the experiment demonstrates learning of RBC from large and remote RDF store by querying its SPARQL endpoint.

**Results** A total of 159 queries were posed to the live SPARQL endpoint, and the model was learned in approximately 30 secs, using 2.8 GHz processor with 4 GB memory, and the network download and upload speed is approximately 3 Mbps.

## 8 Summary and Related Work

**Summary** The emergence of RDF as a basic data representation format for Semantic Web has led to increasing availability of all kinds of data in RDF. Transforming this data into knowledge calls for approaches to learning predictive models from massive RDF stores in settings where (i) the learning algorithm can interact with the data store only through a SPARQL endpoint; (ii) the model needs to be updated in response to updates to the underlying RDF store; and (iii) the attributes that can be used to build the predictive models are not known a priori and hence need to be identified by crawling the RDF store. We have introduced an approach to learning predictive models from RDF stores in such settings using Relational Bayesian Classifiers (RBCs) as an example. We have implemented our solutions in an open source system available as part of the INDUS toolkit for learning predictive models from massive data sets [21] and demonstrated the its feasibility using experiments with several RDF datasets.

[3] `http://logd.tw.rpi.edu/sparql`

**Related Work** The work on SPARQL-ML [17] extends SPARQL with data mining support to build classifiers, including statistical relational models such as RBC from RDF data. Other works on learning predictive models from RDF data include [4] and [28]. In [4] kernel machines are defined over RDF data where features are constructed by ILP-based dynamic propositionalization. In [28] RDF triples are represented as entries in a Boolean matrix, and matrix completion methods are used to train the model and predict unknown triples off-line. However, all the approaches assume that the learner has direct access to RDF data. In contrast, our approach does not require the learning algorithm to have direct access to RDF data, and relies only on the ability of the RDF store to answer aggregate SPARQL queries.

## 9 Acknowledgments

The authors would like to thank our anonymous reviewers who have provided valuable comments. This work is supported in part by a grant (IIS 0711356) from the National Science Foundation and in part by the Iowa State University Center for Computational Intelligence, Learning, and Discovery. The work of Vasant Honavar was supported by the National Science Foundation, 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 National Science Foundation.

## References

1. L. K. Ackerson and K. Viswanath. Communication inequalities, social determinants, and intermittent smoking in the 2003 health information national trends survey. *Prev Chronic Dis*, 6(2), 2009.
2. G. Antoniou and F. van Harmelen. *A Semantic Web Primer*. MIT Press, Cambridge, MA, 2. edition, 2008.
3. T. Berners-Lee, J. Hendler, and O. Lassila. The semantic web. Scientific American, 2001.
4. V. Bicer, T. Tran, and A. Gossen. Relational kernel machines for learning from graph-structured RDF data. In *Proceedings of the 8th Extended Semantic Web Conference*, 2011.
5. K. Breitmann, M. Casanova, and W. Truszkowski. *Semantic Web: Concepts, Technologies and Applications*. Springer-Verlag, 2007.
6. 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 A. Hoffmann, H. Motoda, and T. Scheffer, editors, *Discovery Science*, volume 3735 of *Lecture Notes in Computer Science*, pages 13–44. Springer Berlin / Heidelberg, 2005.
7. T. M. Cover and J. A. Thomas. *Elements of information theory*. Wiley-Interscience, New York, NY, USA, 1991.
8. R. Cyganiak and A. Jentzsch. Linking open data cloud diagram. Online. http://lod-cloud.net/, Accessed 2011.

9. L. Ding, D. DiFranzo, A. Graves, J. R. Michaelis, X. Li, D. L. McGuinness, and J. Hendler. Data-gov wiki: Towards linking government data. In *AAAI Spring Symposium on Linked Data Meets Artificial Intelligence*, 2010.
10. L. Ding, D. DiFranzo, A. Graves, J. R. Michaelis, X. Li, D. L. McGuinness, and J. A. Hendler. TWC data-gov corpus: incrementally generating linked government data from data. gov. In *Proceedings of the 19th international conference on World Wide Web*, pages 1383–1386, 2010.
11. L. Getoor and B. Taskar. *Introduction to Statistical Relational Learning*. The MIT Press, 2007.
12. I. Guyon and A. Elisseeff. An introduction to variable and feature selection. *J. Mach. Learn. Res.*, 3:1157–1182, March 2003.
13. O. Hassanzadeh and M. Consens. Linked movie data base. In *WWW 2009 LDOW Workshop*, 2009.
14. J. Hendler. Science and the semantic web. *Science*, 299:520–521, 2003.
15. P. Hitzler, M. Krötzsch, and S. Rudolph. *Foundations of Semantic Web Technologies*. Chapman & Hall/CRC, 2009.
16. E. Hung, Y. Deng, and V. S. Subrahmanian. RDF aggregate queries and views. In *21st International Conference on Data Engineering*, page 717–728, 2005.
17. C. Kiefer, A. Bernstein, and A. Locher. Adding data mining support to SPARQL via statistical relational learning methods. In *Proceedings of the 5th European semantic web conference*, page 478–492, 2008.
18. R. E. Korf. Depth-first iterative-deepening: an optimal admissible tree search. *Artif. Intell.*, 27:97–109, September 1985.
19. N. Koul, N. Bui, and V. Honavar. Scalable, updatable predictive models for sequence data. In *BIBM*, pages 681–685, 2010.
20. N. Koul, C. Caragea, V. Honavar, V. Bahirwani, and D. Caragea. Learning classifiers from large databases using statistical queries. In *Web Intelligence*, pages 923–926, 2008.
21. N. Koul and H. T. Lin. Indus learning framework. Google Code - http://code.google.com/p/induslearningframework/, 2011.
22. H. Liu and H. Motoda. *Feature Extraction, Construction and Selection: A Data Mining Perspective*. Kluwer Academic Publishers, Norwell, MA, USA, 1998.
23. F. Manola and E. Miller, editors. *RDF Primer*. W3C Recommendation. World Wide Web Consortium, February 2004.
24. D. Nelson, G. Kreps, B. Hesse, R. Croyle, G. Willis, N. Arora, B. Rimer, K. V. Viswanath, N. Weinstein, and S. Alden. The health information national trends survey (HINTS): Development, design, and dissemination. *Journal of Health Communication: International Perspectives*, 9(5):443–460, 2004.
25. J. Neville, D. Jensen, and B. Gallagher. Simple estimators for relational bayesian classifiers. In *Proceedings of the Third IEEE International Conference on Data Mining*, pages 609–612, 2003.
26. E. Prud'ommeaux and A. Seaborne. SPARQL query language for RDF. Online. http://www.w3.org/TR/2008/REC-rdf-sparql-query-20080115/, Accessed 2011.
27. J. Tauberer. The 2000 U.S. census: 1 billion RDF triples. Online. http://www.rdfabout.com/demo/census/, Accessed 2011.
28. V. Tresp, Y. Huang, M. Bundschus, and A. Rettinger. Materializing and querying learned knowledge. In *Proceedings of the ESWC 2009 IRMLeS Workshop*, 2009.