Extending Functional Dependency to Detect Abnormal Data in RDF Graphs

Yang Yu and Jeff Heflin

Department of Computer Science and Engineering. Lehigh University
19 Memorial Drive West, Bethlehem, PA 18015
{yay208,heflin}@cse.lehigh.edu

Abstract. Data quality issues arise in the Semantic Web because data is created by diverse people and/or automated tools. In particular, erroneous triples may occur due to factual errors in the original data source, the acquisition tools employed, misuse of ontologies, or errors in ontology alignment. We propose that the degree to which a triple deviates from similar triples can be an important heuristic for identifying errors. Inspired by functional dependency, which has shown promise in database data quality research, we introduce *value-clustered graph functional dependency* to detect abnormal data in RDF graphs. To better deal with Semantic Web data, this extends the concept of functional dependency on several aspects. First, there is the issue of scale, since we must consider the whole data schema instead of being restricted to one database relation. Second, it deals with multi-valued properties without explicit value correlations as specified as tuples in databases. Third, it uses clustering to consider classes of values. Focusing on these characteristics, we propose a number of heuristics and algorithms to efficiently discover the extended dependencies and use them to detect abnormal data. Experiments have shown that the system is efficient on multiple data sets and also detects many quality problems in real world data.

Keywords: value-clustered graph functional dependency, abnormal data in RDF graphs

1 Introduction

Data quality (DQ) research has been intensively applied to traditional forms of data, e.g. databases and web pages. The data are deemed of high quality if they correctly represent the real-world construct to which they refer. In the last decade, data dependencies, e.g. functional dependency (FD) [1] and conditional functional dependency (CFD) [2, 3], have been used in promising DQ research efforts on databases. Data quality is also critically important for Semantic Web data. A large amount of heterogeneous data is converted into RDF/OWL format by a variety of tools and then made available as Linked Data[1]. During the creation or conversion of this data, numerous data quality problems can arise.

[1] http://linkeddata.org

Some works [4–6] began to focus on the quality of Semantic Web data, but such research is still in its very early stages. No previous work has utilized the fact that RDF data can be viewed as a graph database, therefore we can benefit from traditional database approaches, but we must make special considerations for RDF's unique features. Since the Semantic Web represents many points of view, there is no objective measure of correctness for all Semantic Web data. Therefore, we focus on the detection of abnormal triples, i.e., triples that violate certain data dependencies. This in turn is used as a heuristic of a potential data quality problem. We recognize that not all abnormal data is incorrect (in fact, in some scenarios the abnormal data may be the most interesting data) and thus leave it up to the application to determine how to use the heuristic.

A typical data dependency in databases is functional dependency [7]. Given a relation R, a set of attributes X in R is said to functionally determine another attribute Y, also in R, (written $X \rightarrow Y$), if and only if each X value is associated with precisely one Y value. An example FD *zipCode* $\rightarrow$ *state* means, for any tuple, the value of *zipCode* determines the value of *state*.

RDF data also has various dependencies. But RDF data has a very different organization and FD cannot be directly applied because RDF data is not organized into relations with a fixed set of attributes. We propose *value-clustered graph functional dependency* (VGFD) based on the following thoughts. First, FD is formally defined over one entire relation. However RDF data can be seen as extremely decomposed tables where each table is a set of triples for a single property. Thus we must look for dependencies that cross these extremely decomposed tables and extend the concept of dependency from a single database relation to a whole data set. Second, the correlation between values is trivially determined in a database of relational tuples. But in RDF data, it is non-trivial to determine the correlation, especially for multi-valued properties. For example, in DBPedia, the properties *city* and *province* do not have cardinality restrictions, and thus instances can have multiple values for each property. This makes sense, considering that some organizations can have multiple places. Yet finding the correlation between the different values of *city* and *province* becomes non-trivial. Third, traditionally value equality is used to determine FD. However, this is not appropriate for real world, distributed data. Consider (1) for floating point numbers, rounding and measurement errors must be considered. (2) Sometimes dependencies are probabilistic in nature, and one-to-one value correspondences are inappropriate. For example, the days needed for processing an order before shipping for a certain product is usually limited to a small range but not an exact value. (3) Sometimes certain values can be grouped to form a more abstract value.

In sum, our work makes the following contributions.

- we automatically find optimal clusters of values
- we efficiently discover VGFDs over clustered values
- we use the clusters and VGFDs to detect outliers and abnormal data
- we conducts experiments on three data sets that validate the system

The rest of the paper is as follows. Section 2 discusses related work. Section 3 describes how to efficiently discover VGFDs while Section 4 discusses categorizing property values for their use. Sections 5 and 6 give the experimental results and the conclusion.

2 Related Work

Functional dependencies are by far the most common integrity constraints for databases in the real world. They are very important when designing or analyzing relational databases. Most approaches to find FD [8–10] are mainly based on the concept of an agree set [11]. Given a pair of tuples, the agree set is all the attributes for which these tuples have the same values. Since the search for FDs occurs over a given relation and each tuple has at most one value for each attribute, then each tuple can be placed into exactly one cluster where all tuples in the cluster have the same agree set with all other tuples. Agree sets are not very useful when applied to the extensions of RDF properties, which are equivalent to relations with only two attributes (i.e. the subject and object of the triple). Furthermore, many properties in RDF data are multi-valued and so the correlation between values of different properties becomes more complex. Finally, since most RDF properties are designed just for a subset of instances in the data set, an agree set-based approach will partition many instances based on null values is common.

RDF graphs are more like graph database models. The value functional dependency (VFD) [12] defined for the object-oriented data model can have multi-valued properties on the right-hand side, e.g. *title* $\rightarrow$ *authors*. However the dependencies we envision can have multi-valued properties on both sides and our system can determine the correlation between each value in both sets. The path functional dependency (PFD) [13] defined for semantic data models considered multiple attributes on a path, however the PFD did not consider multi-valued attributes. FD_{XML} is the FD's counterpart in XML [14] where its left-hand side is a path starting from the XML document root which essentially is another form of a record in a database. Hartmann et al. [15] generalized the definitions of several previous FDs in XML from a graph matching perspective.

As mentioned previously, the basic equality comparison of values used in FD is limited in many situations. Algebraic constraints [16, 17] in database relations are about the algebraic relation between two columns in a database and are often used for query optimization. The algebraic relation can be $+, -, \times, /$. However these works are limited to numerical attribute values and the mapping function can only be defined using several algebraic operators. The reason is that numerical columns are more often indexed and queried over as selective conditions in databases than strings. In contrast, we try to find a general mapping function between the values of different properties, both numbers and strings. Additionally, for the purpose of query optimization, they focus on pairs of columns with top ranked relational significance, the major parts in each of these pairs and the

data related to dependencies that is often queried over, rather than all possible pairs of properties and all pairs of values existing in the data set.

Data dependencies have recently shown promise for data quality management in databases. Bohannon et al. [1] focuses on repairing inconsistencies based on standard FDs and inclusion dependencies, that is, to edit the instance via minimal value modification such that the updated instance satisfies the constraints. A CFD [2, 3] is more expressive than a FD because it can describe a dependency that only holds for a subset of the tuples in a relation, i.e., those that satisfy some condition. Fan et. al [2] gave a theoretical analysis and algorithms for computing implications and minimal cover of CFDs; Cong et al. [3], similar to Bohannon et al., focused on repairing inconsistent data. The CFD discovery problem has high complexity; it is known to be more complex than the implication problem, which is already coNP-complete [2]. In contrast to them, we are trying to both automatically find fuzzy constraints, i.e. those that hold for most of the data, and report on exceptional data for applications. Our work incorporates advantages from both FD and CFD, i.e. fast execution and the ability to tolerate exceptions.

With respect to data quality on the Semantic Web, Sabou et al. [4] evaluate semantic relations between concepts in ontologies by counting the similar axioms (both explicit and entailed) in online ontologies and their derivation length. For instance data, previous evaluations mainly focused on two types of errors: explicit inconsistency with the syntax of the ontologies and logical inconsistency that can be checked by DL reasoners. However, many Linked Data ontologies do not fully specify the semantics of the terms defined, and OWL cannot specify axioms that only hold part of the time. Our work focuses on detecting abnormal semantic data by automatically discovering probabilistic integrity constraints (IC). Tao et al. [6] proposed an IC semantics for ontology modelers and suggested that it is useful for data validation purposes. But the precursor problem of how to discover these ICs is not addressed. Furber et al. [5] also noticed that using FD could be helpful for data quality management on the Semantic Web, but do not give an automatic algorithm to find such FDs and, more importantly, direct application of FD to RDF data may not capture the unique characteristics of RDF data.

3 Discovering VGFDs

We begin with some definitions.

Definition 1 *An RDF graph is* $G := \langle I, L, R, E \rangle$*, where three sets* I*,* L *and* R *are instance, literal and relation identifiers and the set of directional edges is* $E \subseteq I \times R \times (I \cup L)$*. Let* $\mathcal{G}$ *be the set of all possible graphs and* $G \in \mathcal{G}$*. Let* $R^- = \{r^- | r \in R\}$*.*

Definition 2 *A Path* c *in graph* G *is a tuple* $\langle I_0, r_1, I_1, ..., r_n, I_n \rangle$ *where* $I_i \in I, r_i \in R \cup R^-$*, and* $\forall i, 0 \leqslant i < n$*, if* $r_i \in R$ *then* $(I_i, r_{i+1}, I_{i+1}) \in E$ *or if* $r_{i+1} \in R^-$ *then* $(I_{i+1}, r_{i+1}, I_i) \in E$*;* $\forall j$*, if* $i \neq j$ *then* $I_i \neq I_j$*.*

Paths are acyclic and directional, but can include inverted relations of the form r^-.

Definition 3 *A Composite Property (P_{comp}) r° in graph G is $r_1 \circ r_2 ... r_n$, where $\exists I_0, ..., I_n$ and $\langle I_0, r_1, I_1, ..., r_n, I_n \rangle$ is a Path in G. Let $\mathcal{R}^\circ$ be all possible P_{comp}s. Given $r^\circ \in \mathcal{R}^\circ$, $Inst(r^\circ, G) = \{\langle I_0, r^\circ, I_n \rangle | \langle I_0, r_1, I_1, r_2, I_2, ..., r_n, I_n \rangle$ is a Path in $G\}$. $Length(r^\circ) = n$. $\forall r \in R$, $r \in \mathcal{R}^\circ$ and $Length(r) = 1$.*

Definition 4 *A Conjunctive Property (P_{conj}) r^+ in graph G is a set $\{r_1, r_2, ..., r_n\}$ (written $r_1 + r_2 + ... + r_n$), where $\forall i, r_i \in \mathcal{R}^\circ$ and $\exists I'$ s.t. $\forall 1 \leq i \leq n$, $\langle I', r_i, I_i \rangle \in Inst(r_i, G)$. Let $\mathcal{R}^+$ be all possible P_{conj}s. $Size(r^+) = \sum_{r_i \in r^+} Length(r_i)$.*

A Composite Property is a sequence of edges on a Path. The subject and object of a P_{comp} are the first and last objects on the Paths consisting of this sequence of edges. Every property is a special case of P_{comp}. A Conjunctive Property groups a set of P_{comp}s that have a common subject I'. Note, each original $r \in R$ is also $r \in \mathcal{R}^\circ$ and each $r^\circ \in \mathcal{R}^\circ$ is also $r^\circ \in \mathcal{R}^+$.

Definition 5 *Given $i \in I$ and $r^\circ \in \mathcal{R}^\circ$, $V^\circ(i, r^\circ) = \{i' | \exists \langle i, r^\circ, i' \rangle \in Inst(r^\circ, G)\}$. Given $r^+ \in \mathcal{R}^+$, $V^+(i, r^+)$ is a tuple $\langle V^\circ(i, r_1), ..., V^\circ(i, r_1) \rangle$ where $\forall j, r_j \in \mathcal{R}^+$.*

Given a P_{comp}, value function V° returns the objects connected with a subject through P_{comp}, and given a P_{conj}, the value function V^+ returns the set of objects connected with a subject through P_{conj}.

Definition 6 *Given $i, j \in I$ and $r^\circ \in \mathcal{R}^\circ$, Dependency Equality (DE) between i and j on r° is: $V(i, r^\circ) \doteq V(j, r^\circ) \iff (\forall x \in V^\circ(i, r^\circ) \iff \exists y \in V^\circ(j, r^\circ), C(x) = C(y))$, where $C(x)$ is the dependency cluster of x (discussed in Section 4). With a slight abuse of notation for DE, given $r^+ \in \mathcal{R}^+$, $V^+(i, r^+) \doteq V^+(j, r^+) \iff \forall r_k \in r^+, V^\circ(i, r_k) \doteq V^\circ(j, r_k)$.*

Definition 7 *A value-clustered graph functional dependency (VGFD) s in graph G is $X \to Y$, where $X \in \mathcal{R}^+$, $Y \in \mathcal{R}^\circ$ and $\forall i, j \in I$, if $V^+(i, X) \doteq V^+(j, X)$ then $V^\circ(i, Y) \doteq V^\circ(j, Y)$.*

These definitions state that for all instances, if the values of the left-hand side (LHS) P_{comp} of a given VGFD satisfy Dependency Equality (DE), then there is a DE on the right-hand side (RHS) P_{conj}. Note, due to the union rule of Armstrong's axioms used to infer all the functional dependencies, if $\alpha \to \beta$ and $\alpha \to \gamma$ hold, then $\alpha \to \beta\gamma$ holds. Therefore, it is enough to define the VGFD whose right-hand side (RHS) is each single element of a set of properties. In this work, DE includes basic equality for both object and datatype property values, transitivity of the sameAs relation for instances and clustering for datatype values.

Shown in Algorithm 1, this section introduces the VGFD search (line 8-15) and the next section introduces value clustering (line 2-5) which is used to detect dependencies. To efficiently discover a minimum set of VGFDs which is a cover of the whole set of VGFDs, our approach essentially is computed level-wise. Each level L_i consists of VGFDs with LHS of size i (Fig. 1 gives an example). The computation of VGFDs with smaller sets of LHS properties can

Algorithm 1 $Search\_VGFDs(G, \alpha, \beta, \gamma)$, $G = (I, L, R, E)$ is a graph; α is the confidence threshold for a VGFD; β is the sampling size; γ is the threshold for pre-clustering.

```
1: S ← ∅, C ← ∅
2: for each r ∈ R s.t. r is a datatype property do
3:    groups ← Preclustering(Range(r), γ)
4:    C_r ← Optimal_Kmeans(Range(r), groups)
5:    C ← C ∪ C_r
6: i = 0
7: L_i ← ∅
8: repeat
9:    i = i + 1
10:   L_i ← Generate_Level_with_Static_Pruning(L_{i−1}, E)
11:   for each s ∈ L_i do
12:      if Runtime_Pruning(s, α, β, E, C) = FALSE then
13:         if (M ← Compute_VGFD(s, α, E, C)) ≠ ∅ then
14:            S ← S ∪ (s, M)                    //M is the set of value mappings of s.
15: until L_i = ∅ or i >= DEPTH_LIMIT
16: return S
```

be used when computing children VGFDs that have a superset of those LHS properties. A similar level-wise search was proposed for the Tane algorithm [9], but each node in Tane corresponds to a subset of our nodes whose LHS is based on single properties instead of P_{comp}s. In contrast, our nodes are finer grained which leads to more opportunities for pruning. Our algorithm starts with level 0. On each new level, it first generates possible VGFDs on this level based on the results of previous levels and it also eliminates many potential VGFDs from further consideration based on some easily computed heuristics (Section 3.1). Then, a runtime pruning (Section 3.3) and a detailed computation (Section 3.2) are conducted on each candidate VGFD. The whole process can terminate at a specified level, or after all levels, although the latter is usually unnecessary and unrealistic. The process returns each VGFD and its value mappings which is used for detecting violations.

3.1 Heuristics for Static Pruning

We first define the discriminability for a property as the number of distinct values divided by the sum of the property extension, and when it is compared between properties, it is over the instances they have in common. Then, static pruning heuristics used to eliminate potential VGFDs from further consideration are:

1. insufficient subject overlap between the LHS and the RHS,
2. the LHS or RHS has too high a discriminability,
3. the discriminability of the LHS is less than that of the RHS.

The information for rule 1 can be acquired from an ontology (e.g. using domain and range information) or a simple relational join on data. Here insufficient

overlap means too few common subjects, e.g. 20. For rule 2, if the discriminability is close to one, e.g. 0.95 which means 95%, the property functions like a superkey in a database. Since such keys identify an individual, they are not useful for detecting abnormal data. For rule 3, if there is a mapping between two such properties, some values of the property with smaller discriminability must be mapped to different values on the RHS which would not be a functional mapping. In order to apply these heuristics, we make the additional observations:

1. The discriminability of a P_{comp} (P_{conj} resp.) is generally no greater than (no less than resp.) that of each property involved.
2. A P_{conj} (P_{comp} resp.) cannot be based on two properties that have few common subjects (objects and subjects resp.).
3. All children of a true VGFD on the level-wise search graph are also true VGFDs, but are not minimal.

For example, given a P_{comp} $A \circ B$, its values all come from the values of B and its extension is a subset of the Cartesian product between objects of A and subjects of B, then its discriminability, i.e. the distinct values divided by the usages, should be no greater than that of each component. A similar explanation applies for P_{conj} in observation 1. An extension of the observation 2 is that a P_{conj} cannot be followed by other properties in a property chain, e.g. $(A+B)\circ C$, since its values are tuples (e.g. the values of $A+B$) as opposed to the instances and literals that are the subjects of another property (e.g. subjects of C).

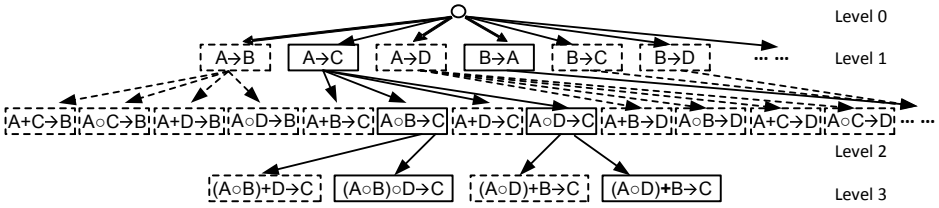

Fig. 1. An example of level-wise discovering process. We suppose that (1) property A and B have few common subjects, (2) the discriminability of B is less than that of C and (3) D has a high discriminability.

Fig.1 is an example showing how these heuristics are useful in the level-wise searching. Each edge is from a VGFD to a VGFD with an LHS that is a superset of the parent LHS and the two VGFDs have the same RHS. Note, our current algorithm does not support the use of composite properties on the RHS. The VGFDs pruned by the above heuristics are in dotted boxes and dotted lines pointing to the children pruned. For this example, we make assumptions typical of real RDF data. For instance, in DBPedia less than 2% of all possible pairs of properties share sufficient common instances. So following our heuristics, four VGFDs on level 1 are pruned: $A \rightarrow B$ is due to heuristic rule 1, $B \rightarrow C$ is due to rule 3 and the other two are due to rule 2. Then the children of $A \rightarrow B$ and

$A \rightarrow D$ are pruned due to the same reason as their parents. $A + B \rightarrow C$ on level 2 and $(A \circ D) + B \rightarrow C$ on level 3 are pruned due to the first assumption plus the observation 2. Finally, $A + D \rightarrow C$ on level 2 and $(A \circ B) + D \rightarrow C$ on level 3 are pruned due to the observation 1 and heuristic rule 2. From this example, we can see simple conditions can reduce the level-wise search space greatly based on these heuristics.

3.2 Handling Multi-Valued Properties

The fundamental difference between VGFD and FD when computing VGFD is that we consider multi-valued properties. When finding FDs in databases, the multi-valued attributes either are not considered (if they are not in the same relation), or the correlation of their values is given by having separate tuples for each value. RDF frequently has multi-valued properties without any explicit correlation of values, e.g. in DBPedia, more than 60% properties are multi-valued. When computing a VGFD, we try to find a functional mapping

Table 1. The left table is the triple list. The right table is mapping count.

deptNo		deptName	
subject	object	subject	object
A	1	A	CS
A	2	A	EE
B	1	B	EE
C	2	C	CS
D	2	D	EE

Candidate Value Mapping	Count
1→ EE	2
2→ EE	2
2→ CS	2
1→ CS	1

from each LHS value to an RHS value such that this mapping maximizes the number of correlations. We consider any two values for a pair of multi-valued properties that share a subject to be a possible mapping. Then we greedily select the LHS value that has the most such mappings and remove all other possible mappings for this LHS value. If multiple RHS values are tied for the largest number of mappings, then we pick the one that appears in the fewest mappings so far. Consider Table 1 which analyzes the dependency *deptNo* $\rightarrow$ *deptName*. The triples are given to the left and each possible value mapping and its maximal possible count are listed in descending order to the right. The maximal count of $1 \rightarrow EE$ is 2, because these two values co-occur for instances A and B once for each. We first greedily choose mapping $1 \rightarrow EE$, because it has the largest count among all mappings for $depNo = 1$. After this selection, the mapping $1 \rightarrow CS$ is removed since $deptNo = 1$ has been mapped. Then for $deptNo = 2$, to maximize the number of distinct values being matched on both sides, we choose $(2, CS)$ since CS has been mapped to by fewer LHS values than EE. Note the basic equality used here is a special case of cluster-based Dependency Equality. For example, if CS and EE are clustered together, then the mappings will be $1 \rightarrow EECS$ and $2 \rightarrow EECS$, where $EECS$ is the cluster.

Our confidence in a VGFD depends on how often the data agree with it, i.e., the total matches divided by the sum of the LHS's extension, e.g. the VGFD above has the confidence of $4/5 = 0.8$. In this work, we set the confidence threshold $\alpha = 0.9$ to ensure that patterns are significant, while allowing for some variation due to noise, input errors, and exceptions.

3.3 Run-time Pruning

In the worst case, the expensive full scan of value pairs must occur $|R^+| \cdot |R^\circ|$ times. So we propose to use *mutual information* (MI) computed over sampled value pairs for estimating the degree of dependency. In Algorithm 2, given a candidate VGFD $s\ X \rightarrow Y$, we start by randomly selecting a percentage β of the instances. In line 2, for each instance i, we randomly pick a pair of values from $V^+(i, X)$ and $V^\circ(i, Y)$. *Distribution*() also applies the clusters C_X and C_Y and returns these pairs in lieu of the actual values. In information theory, a MI I_{XY} of two random variables X and Y is formally defined as $I_{XY} = \sum_i \sum_j p_{i,j} \log (p_{i,j}/p_i p_j)$, where p_i, p_j are the marginal probability distribution functions of X and Y, and $p_{i,j}$ is the joint probability distribution function of X and Y respectively. Intuitively, MI measures how much knowing one of these variables reduces our uncertainty about the other. Furthermore, the *entropy coefficient* (EC), using MI, measures the percentage reduction in uncertainty in predicting the dependent variable based on knowledge of the independent variable. When it is zero, the independent variable is of no help in predicting the dependent variable; and when it is one, there is a full dependency. The EC is directional and $EC(X|Y)$ for predicting the variable X with respect to variable Y is defined as I_{XY}/E_Y, where E_Y is the entropy of variable Y, formally $\sum_j p_j \log 1/p_j$. Because I_{XY} also can be expressed as $E_X + E_Y - E_{XY}$ which has a easier form to compute.

Algorithm 2 *Runtime_Pruning*(s, α, β, E, C), s is a candidate VGFD $X \rightarrow Y$; α is the confidence threshold for a VGFD; β is the sampling size as a percentage; E is a set of triples. C is a set of cluster sets for each property.

1: $I \leftarrow$ *Sampling_Subjects*(s, β, E)//Sampled subjects shared by the LHS and RHS.
2: $\{(X_i, Y_i)\} \leftarrow$ *Distribution*(s, I, E, C) //A list of value pairs where each pair consists of two single sampled values of LHS and RHS for the same subject.
3: $E_X = -\sum_{distinct\ x \in \{X_i\}} \frac{|\{X_i|X_i=x\}|}{|\{X_i\}|} \cdot log \frac{|\{X_i|X_i=x\}|}{|\{X_i\}|}$
4: $E_Y = -\sum_{distinct\ y \in \{Y_i\}} \frac{|\{Y_i|Y_i=y\}|}{|\{Y_i\}|} \cdot log \frac{|\{Y_i|Y_k=i\}|}{|\{Y_i\}|}$
5: $E_{XY} = -\sum_{distinct\ (x,y) \in \{(X_i,Y_i)\}} \frac{|\{(X_i,Y_i)|X_i=x \wedge Y_i=y\}|}{|\{(X_i,Y_i)\}|} \cdot log \frac{|\{(X_i,Y_i)|X_i=x \wedge Y_i=y\}|}{|\{(X_i,Y_i)\}|}$
6: **if** $(E_X + E_Y - E_{XY})/E_X < \alpha - 0.2$ **then**
7: **return** TRUE
8: **return** FALSE

We note that Paradies et al. [18] also used entropy to estimate the dependency between two columns in databases. Since they want to determine attribute

pairs that can be estimated with high certainty, i.e. focusing on precision of the positives, they need a complex statistical estimator. In contrast, our aim is a fast filter that is good enough to remove most negatives, i.e. independent pairs, thus a statistical estimator is not necessary. We can avoid missing positives by setting a low enough threshold. In our experiments, the difference between EC for a 20% sample and EC of full data is less than 0.15 on average and the estimated values typically have higher ECs. For example, it is very rare that a VGFD estimated lower than 0.7 has an actual value above 0.9. Therefore, a threshold of 0.2 less than α (line 6) is a reasonable lower bound for filtering out independent pairs.

4 Clustering Property Values

As introduced in Section 1, we must cluster property values in order to discover dependencies that allow for rounding errors, measurement errors, and distributions of values. For object property values, clustering groups all identifiers that stand for the same real world object by computing the transitive closure of sameAs. The rest of this section discusses clustering the values for each datatype property. This is used to determine Dependency Equality (Definition 6) between two objects.

4.1 Pre-clustering

The pre-clustering process is a light-weight computation that provides two benefits for finer clustering later: the minimum number of clusters and reserves expensive distance calculations for pairs of points within the same pre-cluster. Since the pre-clustering is used for VGFD discovery, there are three thoughts. First, the values to be clustered are from various properties and have very different features. So the clustering process needs to be generic in two aspects: (1) a pair-wise distance metric that is suitable for different types of values and multiple feature dimensions, and (2) suitable for the most common distribution in real world, i.e. the normal distribution. Second, we prefer a comparatively larger number of clusters where elements are really close (if not, they may not be clustered). The reason is that the clusters will be used as class types for detecting dependencies. Larger values of k generate finer-grained class types, which in turn allow us to generate more precise VGFDs, albeit at the risk of bluring boundaries between classes and making it harder to discover some dependencies. This point also makes our approach different from many other pre-clustering approaches, e.g. [19], because their results of pre-clustering can be overlapped and rigid clustering later could merge these groups into fewer clusters.

Based on the above thoughts, specifically, given a list of values, the process first selects a value that is closest to the center (we choose the mean for numeric values and discuss strings in the next paragraph), and then moves it from the list to be the centroid of a new group. Second, for each value on the list, if the distance to this centroid is within the threshold (we use the standard deviation), it will be moved from the list to a new group. Finally, the above process is repeated

if the list is not empty. Thus the process generally finds the cluster around the original center first, and then the clusters further away from the center. This is much better than random selection, because if an outlier is selected, then most instances remain on the list for clustering after this round of computation.

To compute the center and distance of string values, we compute the weight of each token in a string according to its frequency in values for the property. Then we pick the string that has the largest sum of weights divided by the number of tokens in it as the center and the distance between two strings is the sum of weights of the different tokens in them. The intuition is that by taking these strings as a class, the most representative one is the one with the most common words. For example, the property *color* in DBPedia has values "light green", "lime green", etc. Then, the representative of these two strings is the common word "green". For "light green", the distance to "lime green" will be less than that to "light red", since "red" and "green" are more common and have larger weights.

4.2 Optimal k-Means Clustering

There are several popular clustering methods, e.g. k-Means, Greedy Agglomerative Clustering, etc. However most of them need a parameter for the number of resulting clusters. To automatically find the most natural/best clusters, we designed the following unsupervised method of finding optimal clusters.

The approach is inspired by the gap statistic [20] which is used to cluster numeric values with a gradually increasingly number of clusters. The idea is that when we increase k to above the optimum, e.g. adding a cluster center in the middle of an already ideal cluster, the pooled within-cluster sum of squares around the cluster mean decreases more slowly than its expected rate. Thus the gap between the expectation and actual improvement over different k will be in a shape with an inflexion which indicates the best k. Our approach improves upon this idea in three aspects: we start at the number of pre-clusters instead of 1; in each round of k-Means, the initial centroids are selected according to pre-clusters; and the distance computation is only made among points within the same pre-cluster.

Our $Optimal\_kMeans$ algorithm is presented as Algorithm 3. At first, k is set to the number of pre-clusters. At each iteration, we increment k and select k random estimated centroids m_i, each of which starts a new cluster c_i. $Init()$ selects the centroids from the pre-clusters in proportion to their sizes. In each inner loop (line 8-13), every value is labeled as a member of the cluster whose centroid has the shortest distance to this instance among all centroids that are within the same pre-cluster as that value (line 10). Then each centroid is recomputed based on the cheap distance metric until the centroid does not change. After each round of modified k-Means clustering, we compute the difference on $Gap(k)$ and stop the process if it is an inflexion point. Since the clustering is used to detect abnormal data in which string values are expected to be caused by accidental input or data conversion, in this clustering, we use edit distance as the distance metric for string values as opposed to the above pre-clustering.

Algorithm 3 $Optimal\_kMeans(L, groups)$, L is a set of literal values; $groups$ is a set of pre-clustered groups of L.

```
1: k = |groups|
2: Gap(k) = Gap_Statistic(groups)
3: tmpC ← groups
4: repeat
5:    k = k + 1, C ← tmpC, tmpC ← ∅                    //tmpC is the set of k clusters
6:    for each i ≤ k do
7:       Init(m_i), c_i ← c_i ∪ m_i, tmpC ← tmpC ∪ c_i   //m_i is the center of each cluster
8:    repeat
9:       for each x ∈ L do
10:         c_i ← c_i ∪ arg min_{m_i ∈ Group(x)} Distance(x, m_i)
11:      for each i ≤ k do
12:         m_i = Mean(c_i)
13:   until ∀i ≤ k, m_i converges
14:   Gap(k) = Gap_Statistic(tmpC)
15: until Gap(k) < Gap(k − 1)
16: return C
```

5 Experimental Results

For our experiments, we selected the Semantic Web Research Corpus[2] (SWRC), DBPedia and RKB[3] data sets. All of them are widely used subsets of Linked Data that cover different domains. Experiments were conducted on a Sun workstation with 8 Xeon 2.8G cores and 6G memory. We observed that there are few dependencies with an LHS size larger than four and that such dependencies tend to have less plausible meanings. For this reason, we set the maximal size of a VGFD to four in our experiments. Based on clusters and VGFDs, abnormal data has two types: one is far away from other clusters and the other is a violation of VGFDs. Specifically, in this work, a triple is reported as an outlier if its value is the only element of some cluster whose distance to the nearest cluster centroid is above twice of the average distances between all clusters for this property. A triple is reported as abnormal due to violation of VGFDs only when its value conflicts with a value mapping determined by some VGFD and this value mapping is confirmed by other triple usages more than twice.

In our first experiment, we compared the overall performance of the system on three data sets. The sampling size β used in runtime pruning is 20%. In Table 2, we can see that the running time appears to be more heavily influenced by the number of properties than the data set size. Note that RKB has more triples but fewer properties than DBPedia, and thus has more triples per property. This leads to a longer clustering time, but thanks to static and runtime pruning, the total time to find VGFDs is less.

[2] http://data.semanticweb.org/
[3] http://www.rkbexplorer.com/data/

Table 2. System overall performance on SWRC, DBPedia and RKB data sets.

	SWRC	DBPedia	RKB
Number of Triples (M) / Properties	0.07 / 112	10 / 1114	38 / 54
Discovered VGFDs on Level 1	12	228	6
Discovered VGFDs on Level 2	37	304	3
Discovered VGFDs on Level 3	2	126	0
Discovered VGFDs on Level 4	0	53	0
Time for Clustering (s)	18	114	396
Time for Level 1 (s)	11	172	67
Time for Level 2 (s)	20	246	44
Time for Level 3 (s)	4	108	0
Time for Level 4 (s)	1	47	0
Total Time (s) / Discovered VGFDs	54 / 51	687 / 721	507 / 9
Reported Abnormal Triples	75	2868	227

Table 3. Some VGFDs from the three data sets. The first and second group of VGFDs are of size 1 and 2. The third group is a set of VGFDs with clustered values.

VGFD	Description
genus→family	Organisms in the same genus also have the same family.
writer→genre	A work's writer determines the work's genre.
teamOwner→chairman	The teams with the same owner also have the same chairman.
composer→mediaType	The works by the same composer have the same media type.
militaryRank→title	The people of the same military rank also have the same title.
location→nearestCity	The things on the same location have the same nearest city.
topic→primaryTopic	The papers with the same topic have the same primary topic.
manufacturer+oilSystem →compressionRatio	The manufacturer and oil system determine the engine's compression ratio.
publisher ∘ country →language	The publisher's country determines the language of that published work.
article-of-journal+has-volume →has_date	The volume number of a journal where an article is published determines the published date of this article.
faculty→budget	The size of the faculty determines the budget range.
militaryRank→salary	The military rank determines the range of salary.
occupation→salary	The occupation determines the range of salary.
type→upperAge	A school's type determines the range of upper age.

Table 3 gives some VGFDs from the three data sets and their short descriptions. In DBPedia, among 200 samples out of 2868 abnormal triples, 173 of them (86.5%) are confirmed to be true errors in the original data. The correctness of 10 of the remaining triples was difficult to judge. SWRC and RKB have 51% and 62% precision respectively. We believe the lower precision for SWRC is because it has a higher initial data quality and its properties have a much smaller set of possible values than those of DBPedia. We list a number of confirmed erroneous triples in Table 4. For example, the first triple is reported as an outlier after automatic clustering. The second triple violates the VGFD that a school's type determines the cluster of its upper age, because the triple's subject is a certain type of school while its value is not in the cluster of values for the same type of schools.

Next, to check the impact of our pruning algorithms, we performed an ablation study using DBPedia that removes these steps. The left part of Table 5

Table 4. Some confirmed erroneous triples in the three data sets, where r, o, i, p, s are prefixes for http://www.dbpedia.org/resource/, http://www.dbpedia.org/ontology/, http://acm.rkbexplorer.com/id/, http://www.aktors.org/ontology/portal/ and http://data.semanticweb.org/.

1	<r:Shanghai_Jiao_Tong_University, o:university/undergrad, 194323445>
2	<r:Harrow_College, o:School/upperAge, 2009.0>
3	<r:Melbourne_Grammar_School, o:School/ranking, 2006.0>
4	<r:Google_Maps, o:Work/language, r:Coverage_details_of_Google_Maps>
5	<r:Wiktionary, o:Work/language, r:History_and_development>
6	<r:Dembela, o:Place/coordinates, coord|N|W>
7	<r:Hutt_Valley_High_School, o:EducationalInstitution/principal, r:2008>
8	<r:Wake_Island, o:Island/country, r:United_States_Air_Force>
9	<r:Albuquerque_Plaza, o:Building/floorCount, 2221>
10	<r:varedo, o:City/province, r:Province_of_Milan>
11	<i:796511, p:has-date, to-10-01>
12	<i:journals/jair/DarwicheP97, p:has-date, 1996>
13	<s:person/bastian-quilitz, s:ns/swc/ontology#affiliation, research assistant>
14	<s:person/ulf-leser, s:ns/swc/ontology#affiliation, professor>

shows that using static and runtime pruning respectively saves over 62% and 55% of time compared to using neither. Because they utilize different characteristics, using them together saves 85% over neither. When we do not prune, the few additional VGFDs discovered lead to fewer abnormal triples than those discovered with pruning (on average 2.2 per VGFD vs. 3.97 per VGFD). Thus the pruning techniques not only save time but do not affect the abnormality detection much.

Table 5. The left table is showing the impact of our pruning techniques. The right table is comparing our preclustering with an alternative called SortSeq on VGFDs using the clusters and abnormal data found based on these VGFDs.

	None	Static	Runtime	Both
Time (s)	4047	1529	1817	687
VGFDs	746	741	729	721
Abnormal	2923	2915	2887	2868

	Preclustering	SortSeq
Time (s)	114	83
VGFDs	42	23
Abnormal	625	391

Besides pruning, we also checked the impact of our pre-clustering. Because our approach is based on a generic pair-wise distance, we wanted to compare it with a simpler one based on the linear ordering of values where the distance is just the difference between numbers. After each iteration of clustering around the mean, this alternative, referred to as SortSeq, recursively clusters on two remaining value sets: one is above the mean and the other below the mean. To handle strings in this approach, we sort them alphabetically and assign each a sequence number. The right of Table 5 shows that VGFDs and abnormal

data that are based on the baseline clustering are both less than that of our approach. Among the VGFDs not found by the SortSeq, most are for string values. SortSeq finds fewer VGFDs and less abnormal data, because it naively assumes that the more common leading characters two strings have, the more similar they are. Thus, our pre-clustering using cheap and generic computation captures the characteristics of different property values.

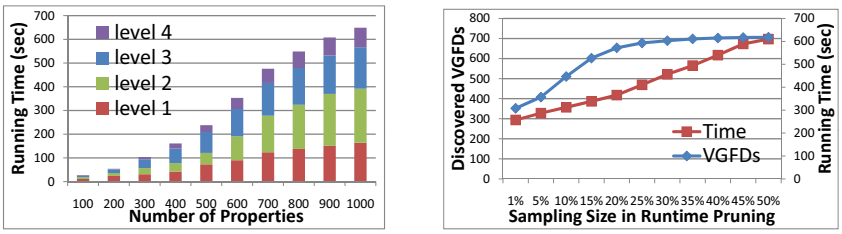

Fig. 2. The left is the effect of number of properties on the VGFD searching time. The right is the effect of sampling size in runtime pruning on the VGFD searching time.

Knowing that pre-clustering and pruning are useful for the system, we systematically checked the trend of system performance, especially time, by using these techniques. To be comparable on data set size, we picked subsets of properties from DBPedia. For each size, we randomly draw 10 different groups of this size and average the time over 10 runs. The left of Fig. 2 shows that the time for every level almost follows a linear trend. The right of Fig. 2 shows the effect of sampling size β used in runtime pruning on the system. We see that the running time is in linear proportion to the sampling size. As the VGFD curve shows, $\beta = 0.2$ is sufficient to find most dependencies for DBPedia.

6 Conclusion

We have presented a system to detect Semantic Web data that are abnormal and thus likely to be incorrect. Inspired by functional dependency in databases, we introduce *value-clustered graph functional dependency* which has three fundamental differences with functional dependency in order to better deal with Semantic Web data. First, the properties involved in a VGFD are across the whole data set schema instead of a single relation. Second, property value correlations, especially for multi-valued properties, are not explicitly given in RDF data. Third, using clusters for values greatly extends the detection of dependencies. Focusing on these unique characteristics, our system efficiently discovers VGFDs and effectively detects abnormal data, as shown in experiments on three Linked Data sets. In the future we plan to use subclass relationships to further generalize object property values. We also would like to take into account other features when clustering, for example the string patterns.

References

1. Bohannon, P., Fan, W., Flaster, M., Rastogi, R.: A cost-based model and effective heuristic for repairing constraints by value modification. In: SIGMOD '05, New York, NY, USA, ACM (2005) 143–154
2. Fan, W., Geerts, F., Jia, X., Kementsietsidis, A.: Conditional functional dependencies for capturing data inconsistencies. ACM Trans. Database Syst. **33** (June 2008) 6:1–6:48
3. Cong, G., Fan, W., Geerts, F., Jia, X., Ma, S.: Improving data quality: consistency and accuracy. In: VLDB '07, VLDB Endowment (2007) 315–326
4. Sabou, M., Fernandez, M., Motta, E.: Evaluating semantic relations by exploring ontologies on the Semantic Web. (2010) 269–280
5. Fürber, C., Hepp, M.: Using SPARQL and SPIN for data quality management on the semantic web. In: BIS., Springer (2010) 35–46
6. Tao, J., Sirin, E., Bao, J., McGuinness, D.L.: Integrity constraints in OWL. In Fox, M., Poole, D., eds.: AAAI, AAAI Press (2010)
7. Codd, E.F.: Relational completeness of data base sublanguages. In: Database Systems, Prentice-Hall (1972) 65–98
8. Mannila, H., Räihä, K.J.: Algorithms for inferring functional dependencies from relations. Data Knowl. Eng. **12**(1) (1994) 83–99
9. Huhtala, Y., Krkkinen, J., Porkka, P., Toivonen, H.: Tane: An efficient algorithm for discovering functional and approximate dependencies. The Computer Journal **42**(2) (1999) 100–111
10. Lopes, S., Petit, J.M., Lakhal, L.: Efficient discovery of functional dependencies and armstrong relations. In: EDBT '00, London, UK, (2000) 350–364
11. Beeri, C., Dowd, M., Fagin, R., Statman, R.: On the structure of armstrong relations for functional dependencies. J. ACM **31** (January 1984) 30–46
12. Levene, M., Poulovanssilis, A.: An object-oriented data model formalised through hypergraphs. Data Knowl. Eng. **6**(3) (May 1991) 205–224
13. Weddell, G.E.: Reasoning about functional dependencies generalized for semantic data models. ACM Trans. Database Syst. (1992) 32–64
14. Lee, M.L., Ling, T.W., Low, W.L.: Designing functional dependencies for XML. In: EDBT '02, London, UK, Springer-Verlag (2002) 124–141
15. Hartmann, S., Link, S., Kirchberg, M.: A subgraph-based approach towards functional dependencies for XML. In: SCI '2003, 200–211
16. Brown, P.G., Hass, P.J.: Bhunt: automatic discovery of fuzzy algebraic constraints in relational data. In: VLDB '2003, VLDB Endowment (2003) 668–679
17. Haas, P.J., Hueske, F., Markl, V.: Detecting attribute dependencies from query feedback. In: VLDB '07, VLDB Endowment (2007) 830–841
18. Paradies, M., Lemke, C., Plattner, H., Lehner, W., Sattler, K.U., Zeier, A., Krueger, J.: How to juggle columns: an entropy-based approach for table compression. In: IDEAS '10, New York, USA, ACM (2010) 205–215
19. McCallum, A., Nigam, K., Ungar, L.H.: Efficient clustering of high-dimensional data sets with application to reference matching. In: KDD '00, 169–178
20. Tibshirani, R., Walther, G., Hastie, T.: Estimating the number of clusters in a data set via the gap statistic. J. of the Royal Statistical Society **63**(2) (2001) 411–423