Detecting Abnormal Semantic Web Data Using Semantic Dependency

Yang Yu
Computer Science and Engineering Dept.
Lehigh University
Email: yay208@cse.lehigh.edu

Yingjie Li
Computer Science and Engineering Dept.
Lehigh University
Email: yil308@cse.lehigh.edu

Jeff Heflin
Computer Science and Engineering Dept.
Lehigh University
Email: heflin@cse.lehigh.edu

***Abstract*—Data quality is a critical problem for the Semantic Web. We propose that the degree to which a triple deviates from similar triples can be an important heuristic for identifying errors. Inspired by data dependency, which has shown promise in database data quality research, we introduce *semantic dependency* to assess quality of Semantic Web data. The system first builds a summary graph for finding candidate semantic dependencies. Each semantic dependency has a probability according to its instantiations and is subsequently adjusted based on the inconsistencies among them. Then triples can get a posterior probability of normality based on what semantic dependencies can support each of them. Repeating the iteration above, the proposed approach detects abnormal Semantic Web data. Experiments have shown that the system is efficient on data set with 10M triples and has more than a ten percent F-score improvement over our previous system.**

***Index Terms*—Detecting Abnormal Semantic Web Data; Semantic Dependency**

I. Introduction

Recently, data dependencies [1], [2] have been used in promising data quality (DQ) research efforts on databases. One of the most important data dependencies is functional dependency (FD), which is a constraint between two sets of attributes in a relation from a database. 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. Customarily, X is called the determinant set and Y the dependent attribute. An example FD *affiliation* $\rightarrow$ *based_near* means the value of property *affiliation* determines the value of property *based_near*.

Inspired by FD, we introduce the semantic dependency (SD) for data quality on Semantic Web data. Compared to a FD, there are three fundamental differences in a SD (formal definitions are given in Section IV).

1) The determinant set consists of chains of properties instead of single attributes in database. In this paper, the left-hand side (LHS) of a SD is called the premise and the right-hand side (RHS) is called the conclusion.
2) The first subject value and the last object value of the LHS property chain are the same as those of conclusion property and other attribute values in SD do not need to be specified.
3) To make SDs more flexible and meet the real world situations involving noise and exceptions, we relax SD rules to be probabilistic rather than deterministic, i.e. each SD has a probability.

Here are some observations for the second difference. A FD actually captures an implicit relation between the determinant attribute value and the value of the dependent attribute. In the above FD example, there is an implicit relation between the value of *affiliation* and the value of *based_near*, i.e. people is located near the place where their affiliation is located. But RDF data is abundant of relational descriptions and these implicit relations are oftentimes stated explicitly. For instance, a SD *affiliation* $\circ$ *location* $\rightarrow$ *based_near* captures the same semantics of the above FD. Additionally, the object value of *affiliation* (also the subject value of *location*) needs not to be specified. Then, the flexibility enables the SD to capture more fundamental characteristic of this dependency, because, in this example, if the data is about a global organization with multiple locations, one affiliation can no more determine a place. Such SDs are many, especially in RDF data, because RDF data usually is instance centric and objects on a RDF graph are often connected through multiple predicates with other objects. Here are more examples: *worksIn* $\circ$ *hasPI* $\rightarrow$ *supervisor*, which means the principal investigator of a project that a person works in usually is the supervisor of that person and *make* $\circ$ *hasTopic* $\rightarrow$ *interest*, which means the topic of a paper is the same as the research interest of that author. We note that some SDs above may not always be deterministic in real world data. This is the reason of the third difference between a SD and a FD. Although these rules are not deterministic, they collaboratively can give enough confidence to detect abnormal data in many cases.

The main process of the system is as follows. First, we assign every triple a prior probability. A straightforward assignment without prior knowledge would be using a uniform distribution, i.e. all triples are equal likely (in)correct, e.g. 0.5. Another prior probability assignment based on prior knowledge could be a function of the source that contains the statement, e.g. all triples in a certain source have a particular value. Next (Section II), a summary graph is built for efficiently extracting candidate SDs and their instantiations. Then each SD's truth probability (Section III) is based on the instantiations and prior probability of triples. Taking a logic perspective, these SDs are also probabilistic axioms

over the original ontology, the next step (Section III) is to revise the beliefs of them according to their logical relations, i.e. inconsistencies. The final step in an iteration is to get a score for each triple based on the number and the probability of the SDs that could corroborate it. Then transforming the triple score into prior probability of the next iteration until the difference between posterior and prior probability for all triples is less than certain threshold.

II. Semantic Dependency

We first give some preliminary definitions and then introduce how to efficiently find SDs in the given RDF data set.

Definition 1 A RDF graph is a structure $G := (I, E, R)$. Two disjoint sets I and R are instance identifiers and relation identifiers. The set of directional edges is $E \subseteq I \times R \times I$. Let $\mathcal{G}$ be the set of all possible graphs and $G \in \mathcal{G}$.

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

Definition 3 A Joined Relation Pattern (JRP) j in graph G is $\langle r_1, r_2, ..., r_n \rangle$, where $\exists I_0, I_1, ..., I_n$ and $\langle I_0, r_1, I_1, ..., r_n, I_n \rangle$ is a Path. $Inst(j, G)$ is the set of all such Paths in G for j.

Definition 4 A Semantic Dependency (SD) s in graph G is $X \rightarrow Y$, where X is a JRP, $Y \in R$ and $\exists p \in Inst(X, G)$, I_0 and I_n are the first and last instances on p respectively, s.t. $(I_0, Y, I_n) \in E$. $LHS(s) = X$ and $RHS(s) = Y$.

A. Summary Graph

As the definition shown, the LHS of a SD is based on a JRP. So we need to find all JRPs that can contribute to SDs first. But enumerating all possible JRPs by computing the instance intersections between all pair wise relations is very time consuming and especially most of the pairs do not have intersections. Based on this observation, we want to first prune those pairs of sets that cannot have overlap. Since these JRPs are similar to Basic Graph Patterns (BGP) in a SPARQL query [3], our solution to find the premise of a SD is inspired by the BGP optimization. We build a summary graph G' corresponding to the original graph G. An individual in G' represents individuals in G which are used as the domains or ranges of same group of properties. Formally an graph G' is a summary of a graph G if there is a mapping function $f : \mathcal{G} \rightarrow \mathcal{G}'$ that satisfies the following constraints:

1) if $\langle a, p, b \rangle \in E$ then $\langle f(a), p, f(b) \rangle \in E'$.
2) if $\langle a', p, b' \rangle \in E'$ then there exists two individuals $a, b \in I$, s.t. $a' = f(a), b' = f(b)$, and $\langle a, p, b \rangle \in E$.
3) if $\langle a', p, b' \rangle \in E'$ and $\langle a'', p, b'' \rangle \in E'$, then $a' = a''$ and $b' = b''$.

The construction process can be done efficiently from $\mathcal{A}$ using conventional relation database queries. The process to construct the summary graph compresses each predicate by merging all the domain objects and range objects into two summary representative nodes. Meanwhile it merges all other property links connected with nodes being merged. In the example summary building process shown in Fig. 1, (b) is a intermediate state that the node *P1* and *P2* are merged because they are both used as the object value of the predicate author. If two edges are connected through a common node on summary graph, the two properties represented by them possibly can be joined.

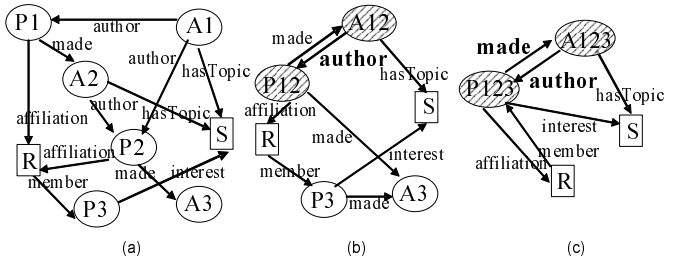

Fig. 1. An example process building a summary graph. The summarized predicates are highlighted and the summarized nodes are shaded. (a) The original RDF graph. (b) The predicate author is summarized. (c) The predicate made is summarized.

B. Finding Candidate Semantic Dependencies

In this and following subsection, we discuss how to find and determine a SD. As we defined, one pre-condition of a SD is that its premise and conclusion have the same subject and object value. On the summary graph, this condition means a JRP, a path of multiple edges on summary graph, connecting two end points of a relation which is a direct edge on summary graph. Then if we find such a case on summary graph, there could be a semantic dependency whose premise is that JRP and conclusion is the direct relation. For example, the JRP consisting of *made* and *hasTopic* on Fig. 1 connects the two end points of the *interest*. So a candidate SD could be *made* $\circ$ *hasTopic* $\rightarrow$ *interest*. Note the direction on the summary graph only reflects the predicate used in real data, however the semantics of the inverse relation of that predicate can also be considered, even if the inverse property is not defined in data. For example, *hasTopic* $\circ$ *interest*$^-$ $\rightarrow$ *author* is also a candidate SD, though the inverse of interest may not be defined. Based on this point, a SD is embedded as a cycle on the undirected summary graph. Then the algorithm finding all possible SDs is transformed to find all undirected cycles on the summary graph and then recover the directionality of properties in premise according to the direction of conclusion.

C. Determining a Semantic Dependency

A candidate SD is true only if it has instantiations. Since each SD is a cycle on the summary graph consisting of its premise and conclusion, a SD is a special JRP whose head and tail are the same. Thus to determine a SD, we need a function $Inst()$ to find all the Paths that are instantiations of a JRP in the original graph. Since an instantiation of a JRP should satisfy all the join conditions along the chain, the selectivity of each relation, i.e. the number of instantiations, is a good greedy heuristic to use. In each iteration, the most selective unsolved relation that is connected with some solved part is picked as the target to solve. Then the instances in the connected solved part is used as the constraint to find the instances satisfying the

target relation. After that, since the set of triples satisfying the solved part could be shrunk, it propagates the changes over all other parts solved previously. A new iteration begins if there is still any unsolved part. Finally, the set of Paths based on these triple sets are constructed and returned.

III. Probability Computation

Given the pairs of a triple and its prior probability, subsection A and B will introduce how to determine SD's probability. Subsection C is about computing posterior probability of triples based on SDs.

A. Probability of Semantic Dependency

Each cycle we found on the summary graph can be interpreted as multiple SDs. For example, in Fig. 1, the cycle consisting of *make*, *topic* and *interest* can be interpreted as three SDs whose conclusions are three relations respectively. Intuitively, the conclusion of *interest* derived from the premise *made* $\circ$ *hasTopic* is more believable than the conclusion of *made* derived from *interest* $\circ$ *hasTopic*$^-$. The analysis under the intuition is that the premise is more specific than the conclusion and there are less counter examples. Thus the probability of SDs in the same cycle should be different and we consider the following in its computation. First, the greater the number of instantiations of a SD, the more believable a SD is. Second, each instantiation of premise only, i.e. without the conclusion part, decreases the belief of this SD, because it can be seen as a counter example of this SD. Third, the belief of triples involved in these instantiations affects the belief of a SD. Therefore, taking the triples and their prior probabilities as inputs, the probability of a SD s is defined as equation 1, which is the sum of the probability of SD instantiations divided by the number of premise instantiations. The common naive Bayes assumption is used here. Since the multiplication of the probability of triples on a Path is less than or equal to 1 and the number of Paths in *Inst(s)* is less than or equal to that in *Inst(LHS(s))* due to stronger constraint on s than that on *LHS(s)*, the equation guarantees the probability is in [0, 1].

$$P(s, G, P_G) = \frac{\sum_{i \in Inst(s,G)} \prod_{t \in i, (t,p) \in P_G} p}{|Inst(LHS(s), G)|} \quad (1)$$

B. Belief Revision

From a logic perspective, the SDs that we found are also axioms defined over the original ontology. In a well-formed ontology, the TBox concepts should be consistent and all concepts and properties should be satisfiable. Thus, based on the explicitly defined original TBox and these probabilistic axioms we found from data, we need to do a consistency check on them and accordingly revise the beliefs on these axioms.

Axioms that infer inconsistencies should be penalized if they are involved in this inference, which also indirectly rewards other axioms. Every group of inconsistent axioms we tracked is the minimal set, i.e. no proper subset of this group can make this inconsistency. Intuitively, if only one possible world that has very small probability satisfies these axioms, the degree of inconsistency of this group is very low and the axioms in it may not need to be blamed too much. Thus, the likelihood of possible worlds, whose set of axioms is inconsistent, reflects the degree of inconsistency of this group of axioms. Therefore we define that each possible world contribute equally to the degree of inconsistency of all groups of inconsistent axioms that it can entail. The probability of each possible world is equal to the multiplication of the probabilities of every axiom in it and they naturally sum to 1.

Since the inconsistency of every group is caused by axioms in it, we partition the degree of inconsistency on each axioms within the group. Therefore, the more groups of unsatisfiable inference an axiom is involved in, the more penalties it will get. The reason is that usually the majority of the world knowledge is compatible, so it is more likely to be erroneous when it conflicts more with other world knowledge.

C. Posterior Probability of a Triple's Normality

The purpose of finding semantic dependencies and computing their probabilities is to compute triple scores by checking how each triple is supported by these probabilistic axioms. This is the last step in each iteration of the system. For each triple, the system iterates through all the SDs whose conclusion is the same as the predicate of this triple. If the subject and object of this triple appear as the first and last instance of an instantiation of the premise of a SD, this premise instantiation can leads to the conclusion as this triple, which is an evidence supporting this triple. The normality score of this triple will be the sum of the probability of all these SDs. Thus the minimum of the score is 0, i.e. no such SDs can support it, and the largest theoretical score is the total probabilities of a set of SDs with a certain conclusion, i.e. all these SDs can support it. Because the scores in this range reflect certain probability of a triple's normality, it is sound in mathematics that the scores can be projected onto the range [0, 1] as probabilities by a normalization with the largest theoretical sum. Finally, the algorithm returns the whole set of triples and each associated with a probability and we put them in next iteration as triples' prior probability.

The whole iterative process of system is guaranteed to terminate under all three possible situations about the change of a triple's score. First, the score is monotonic, either up or down. Since the triple's score is within [0, 1], in the extreme case, a triple will finally get 1 or 0 and there is no change any more. It will satisfy the stopping condition. Second, the score is not monotonic but the changes decrease. That will also satisfy the stopping condition at certain time when the change falls into a given threshold. The last case is that the triple score is oscillating and the changes do not decrease. It means the system can only bound the triple score into a range, so we take the average over previous iterations as its score which is expected to be close to the median of the range.

IV. Experiments

For our experiments, we selected two representative data sets, the Semantic Web Research Corpus[1] (SWRC) and DB-

[1] http://data.semanticweb.org/

TABLE I
SEVERAL EXAMPLE SEMANTIC DEPENDENCIES IN SWRC AND DBPEDIA.

Semantic Dependency	Description
made$^-$ ∘ affiliation ∘ member ∘ maker$^-$ ∘ hasTopic → hasTopic	Colleagues have papers with the same topic.
isPartOf ∘ isPartOf$^-$ ∘ hasTopic → hasTopic	Papers that are in the same part of a proceeding have the same topic.
author$^-$ ∘ creator ∘ author$^-$ ∘ made$^-$ ∘ heldBy$^-$ → holdsRole	A co-author of a co-author of this paper holds a role in this conference.
publisher ∘ country ∘ language → language	The language of publisher's country is the same as the work's language.
parentCompany ∘ keyperson → owner	The key person of the parent company is the owner of this company.

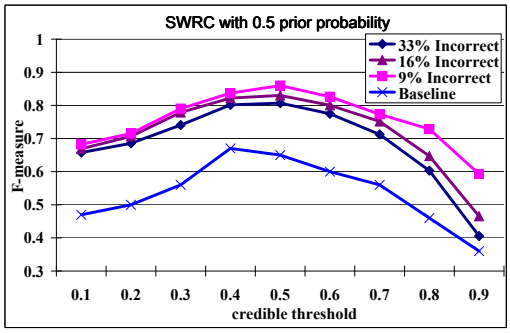

Fig. 2. The effect of different percentage of abnormal data in SWRC and different credible relation threshold.

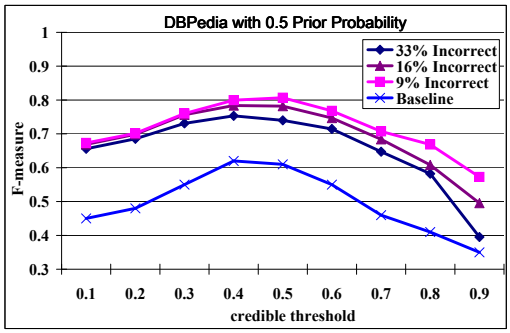

Fig. 3. The effect of different percentage of abnormal data in DBPedia and different credible relation.

Pedia[2]. Both are real world data and on different domains.

Without identification of any types of triple for system in advance, we wanted to test the system's ability to differentiate the abnormal (incorrect) data from other data. However almost no data set has the negation of triples (recently OWL 2 added this function) and to statistically validate the system, the abnormal triples are generated by the following process. For each predicate, we create a domain set consisting of all the distinct subjects of triples using this predicate and similarly a range set consisting of all objects from them. Then a subject and an object from each set are randomly selected to compose a synthetic triple of this predicate. This step can ensure the synthetic triple still conforms to the ontologies of this data set. Four Semantic Web experts verified 200 randomly sampled SWRC test data that all the positives are correct and all the negatives are incorrect.

Before looking at the performance on differentiating triples, we first show several example SDs in SWRC and DBPedia in Table I. In the SD on the first row, the LHS of this SD says the following. A paper P is made by a person who has a certain affiliation and that affiliation has a member who is the maker of another paper which has a certain topic. The RHS of it says that the paper P also has that topic. Thus the underlying meaning is that colleagues often have papers with the same topic (shown as description in the table). We note that they may not be always true, but they do give some sense about the usually expected context for a triple.

[2]http://www.dbpedia.org

TABLE II
THE EFFECT OF STOPPING THRESHOLD ON THE SYSTEM IN SWRC.

Threshold	0.001	0.005	0.01	0.05	0.1
time (hours)	5.8	3.9	2.9	1.8	0.8
iterations	28	19	13	8	3
F-measure	88.16%	87.44%	85.97%	81.71%	68.65%

First, we show the system's performance on data sets with different percentage of abnormal data. All triples are assigned prior probability 0.5 and the stopping threshold is set 0.01. We tested three ratios of abnormal triples to normal triples, 1 to 10 (9%), 1 to 5 (16%) and 1 : 2 (33%). From Fig. 2 and Fig. 3, we see the loss on the best performance in each case is less than the corresponding increase of the number of abnormal triples, e.g. there is only 3% loss on the best F-measures from 16% to 33% abnormal triples on SWRC and 4% on DBPedia. In addition, we compared with our previous system [4] which is shown as a baseline in the figure. To not overwhelm readers, we only show the baseline system on the data set with 33% abnormal data. We see that our previous system performs at least 10% worse than this proposed system in two data sets.

Table II shows the effect of stopping threshold on the system. The tests are done on SWRC with 0.5 prior probability. We see that the system can achieve good F-score within reasonable time length and the stopping threshold does not need to be too small.

REFERENCES

[1] P. Bohannon, W. Fan, M. Flaster, and R. Rastogi, "A cost-based model and effective heuristic for repairing constraints by value modification," in *SIGMOD '05*. New York, NY, USA: ACM, 2005, pp. 143–154.

[2] W. Fan, F. Geerts, X. Jia, and A. Kementsietsidis, "Conditional functional dependencies for capturing data inconsistencies," *ACM Trans. Database Syst.*, vol. 33, pp. 6:1–6:48, June 2008.

[3] M. Stocker, A. Seaborne, A. Bernstein, C. Kiefer, and D. Reynolds, "Sparql basic graph pattern optimization using selectivity estimation," in *WWW '08*. New York, NY, USA: ACM, 2008, pp. 595–604.

[4] Y. Yu and J. Heflin, "Detecting abnormal data for ontology based information integration." in *International Workshop on Semantic Technologies for Information-Integrated Collaboration.*, Philadelphia, PA, USA. May 2011, pp. 431–438.