Detecting Abnormal Data for Ontology Based Information Integration

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

ABSTRACT

To better support information integration on Semantic Web data with varying degrees of quality, this paper proposes an approach to detect triples which reflect some sort of error. In particular, erroneous triples may occur due to factual errors in the original data source, misuse of the ontology by the original data source, or errors in the integration process. Although diagnosing such errors is a difficult problem, we propose that the degree to which a triple deviates from similar triples can be an important heuristic for identifying errors. We detect such "abnormal triples" by learning probabilistic rules from the reference data and checking to what extent these rules agree with the triples. The system consists of two components for two types of abnormal relational descriptions that a Semantic Web statement could have, whether accidentally or maliciously: a statement could relate two resources that are unlikely to have anything in common or an inappropriate predicate could be used to describe the relation between the two resources. The classification technique is adopted to learn statistical characteristics for detecting a suspect resource pair, i.e. there is no significant relation between the subject and the object in the statement. For the suspect usages of a predicate, the system learns semantic patterns for each predicate from indirect semantic connections between the subject / object pairs.

KEYWORDS: Detecting abnormal data, Ontology based information integration.

1. INTRODUCTION

Low data quality (DQ) is a pressing problem for consumers of integrated information. Having access to information of known quality becomes critical for the well-being and indeed for the functioning of modern information integration systems [6]. DQ research has been intensively studied on traditional information formats, e.g. databases and web pages. Recently some works [1, 2, 10, 12] began to focus on the quality on Semantic Web data. Data quality is often used synonymously with correctness [6]. However since the Semantic Web represents many points of view, there is no objective measure of correctness for all semantic web data. Therefore, we consider using an abnormality heuristic that could indicate a data quality problem at the triple level. We recognize that not all abnormal data is incorrect (in fact, in some scenarios the abnormal data may the most interesting data) and thus leave it up to the application to determine how to use the heuristic. If an answer is derived from abnormal data, it may be ranked lower, filtered out, or flagged for a user to confirm or deny the quality of data. A typical use case could be that the system attempts to integrate or query over some new Semantic Web data sources. These sources may be converted to use a schema that is compatible with the system, or may already conform to ontologies that are aligned with those used by the system. Then the system needs certain indication of quality of these sources in order to flag them or rank the query results. The triple-level assessment done in this work can be easily aggregated at the granularity of sources, topics or providers. The essential idea of this work is based on the fact that a statement can get supporting evidence if it can be entailed from other data. Similarly, if the statement cannot be derived through probabilistic entailment from other data and the probabilistic entailment is applicable for most data that is assumed or verified correct (e.g. the Semantic Web Research Corpus (SWRC) and DBPedia[1], some other reputed sources), the statement would be considered abnormal.

Several observations show that the probabilistic entailment

[1] http://dbpedia.org/

by applying probabilistic rules on the context is applicable to many Semantic Web data. First, the ontology limits data to certain topics or domains. Thus the objects in the data that conform to the same ontologies are usually described and connected in certain common ways. Second, most data have some supporting evidence in the context, e.g. the papers, colleagues, co-authors and students in the context can support the description about a professor. Third, usually the relation context for a pair of objects is similar to the contexts for other pairs having the same relationship. Consider the statement A advises B: in some situations where this is true, there are also statements such as A is the principal investigator of project C, B works in C. This rule is clearly not certain. Yet, when combined with other forms of evidence, it can provide support for the advises relation. The notion of *significant relation* used in this work is tied to the ontologies used by the system.

Due to misunderstanding of ontologies or other reasons, an author could misuse an object value for a property or misuse a property relating two URIs. Although a triple could have these two cases at the same time, it can be evaluated by checking each. Currently, the system does not consider literals and datatype properties. Corresponding to these two types of doubtful relational descriptions in a statement, our system has two components to solve each. Using supporting evidence existing in data, the system only need input a set of generally correct sources. Classification techniques are adopted to learn the parameters of statistical rules used to determine the existence of a significant relation. To determine which relation exists between a pair, the system tries to find potential semantic patterns existing in context. We demonstrate that this work is applicable to systems integrating large Semantic Web data sets with rich descriptions over limited vocabularies. Our experimental results showed that the training process can be done by sampling on a small fraction of the data set without significantly compromising quality. Finally the runtime computation for getting the context and evaluating the triple is also efficient.

This work has the following contributions:

- a means for detecting abnormal triples without using any foundational assessments or metadata
- a novel algorithm for constructing the context of a triple by finding all paths between the subject and the object
- an experiment that validates the system using two real world data sets

The paper is organized as following. Section 2 introduces related works. Section 3 describes the process to build the context for a triple. The detailed statistic features for determining the existence of a significant relation is described in section 4. Section 5 discusses the semantic rules used to derive the type of relations. The last two sections present experimental results and the conclusion.

2. RELATED WORK

Hartig et al. [10] proposed a framework to assess the information quality of data sources based on provenance. Bizer et al. [1] described a framework to filter poor information in Web-based information systems according to user defined quality requirements. Both of them use Semantic Web techniques to focus on the subjective assessment by users which needs human effort and may not be objective and automatic enough. Lei, et al. [12] proposed an approach to identify data quality problems in semantic annotations during the creation. Sabou et al. [9] evaluate semantic relations between concepts by counting the repeating (both explicit and entailed) of the similar axioms in online ontologies and their derivation length. But an objective quality assessment on existing Semantic Web instance data is more important for ontology based information integration.

Previous instance data evaluation mainly focused on two types of errors. First, the data usages are explicitly inconsistent with the syntax of the ontologies, e.g. what the W3C RDF Validator checks. The second type of errors is logical inconsistency. They may deploy tools such as Pellet to check logic consistency. Tao et al. [7] captured other two potential issues through SPARQL queries of a conjunctive combination of the presence and absence of certain triple patterns. The first one is that an instance is used as an unexpected individual type. For instance, the individual is used as a subject of a property while the domain of this property is not the type of this individual. The second issue is that the usage of a property on an individual violates the cardinality restriction. Furber et al. [2] tried to improve the quality of literal values by using SPARQL queries to identifying missing literals, illegal literals, etc. It is noted that using SPARQL query syntax can only describe limited triple patterns and the issues detected are still like logic inconsistencies. However many more potential and general semantic issues can not be detected.

Additionally, since our work is to check the relation between two URIs, it is related to the link prediction problem in social network analysis (SNA). Liben-Nowell [4] and Getoor [8] both compared most popular link prediction techniques. But we note that most of these techniques are applied on the network of single or a few types of links. Some SNA researchers began to explore the help of ontologies, e.g. [3], but they use the ontology as a dictionary to help determine the distance between concepts mentioned

in user profiles and still only predict the single friendship link. The link discovery in multi-relational data [11] tried to find novel interesting paths between entities rather than a normal link prediction.

3. CONTEXT CONSTRUCTION

Our approach is based on the context which includes entities having certain direct or indirect relationships with the pair of instances in a triple. This section discusses how to build the context. Four preliminary definitions are given first.

Definition 1. A RDF graph is a structure G := (I, E, R). Two disjoint sets I and R are instance identifiers and relationship identifiers respectively. 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 p in graph G is a tuple $\langle I_0, r_1, I_1, r_2, I_2, ..., r_n, I_n \rangle$ where $\forall$ i, $0 \leq i < n$, $I_i \subseteq I$, $(I_i, r_{i+1}, I_{i+1}) \subseteq E$ or $(I_{i+1}, r_{i+1}^{-}, I_i) \subseteq E$ and $\forall$ j, $0 \leq j \leq n$, $I_i = I_j$ iff. i = j. Length(p) is defined as the number of relations in the tuple.

Definition 3. The path set for a pair $\langle s, d \rangle$ is: $PATHS_G(s, d) = \{p_i | p_i = \langle I_{i0}, r_{i1}, I_{i1}, r_{i2}, I_{i2}, ..., r_{in}, I_{in} \rangle$, p_i is a path in G and $I_{i0} = s$, $I_{in} = d\}$.

Definition 4. A context of size n for a triple $\langle$sub, pred, obj$\rangle$ is a function $C_{G,n}: I \times R \times I \rightarrow \mathcal{G}$. It produces a subgraph S of G such that S := (I', E', R'), $I' \subseteq I$, $R' \subseteq R$, $E' \subseteq E$, and $\forall e \subseteq E'$, $\exists p$, $p \in PATHS_G$(sub, obj), Length(p) $\leq$ n where e is on p.

3.1. Example Contexts

We first show two small example contexts drawn from SWRC data set (Fig. 1) to demonstrate the difference between the contexts for pairs with and without significant relation. From the graph on the right hand side, we see the key connections are through a node, U.S. This node is not specially involved in this context since this node is also connected via the same relation with many other nodes. And few of these other nodes are involved in this subgraph. On the contrary, in the left example, many nodes are distinctive for the pair, such as colleagues and co-authored papers, because a majority of neighbors of these colleagues and papers are involved in the context. Specifically, all the neighbors of the node Li-ding are involved in the actual context for the left hand side, while less than 5% neighbors of it are involved in the actual context for the right hand side (Due to space, we do not show this in the figure). A

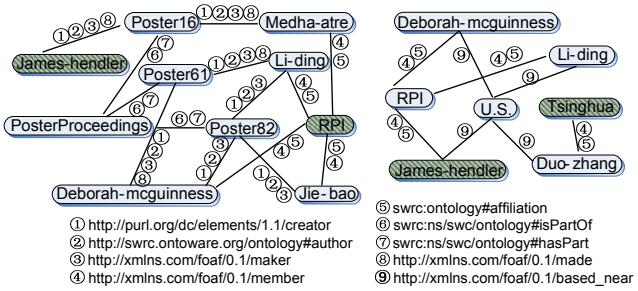

Figure 1. Part of two contexts from SWRC data set, the left is for James Hendler and RPI and the right is for James Hendler and Tsinghua University. The namespace of swrc is $\langle$"http://data.semanticweb.org/"$\rangle$.

similar situation exists for the node Deborah-mcguinness. Besides the different involvement of the nodes, the predicates are also used in different ways. Most predicates on the left are used among different instances. While in the right subgraph, each predicate is used on less instances, e.g. U.S. is the only subject of property 9. Therefore, we can see that the distinctiveness is not a global static concept, and instead it is dynamic with respect to different contexts.

3.2. Construction Process

Since the context is a connection subgraph which brings abundant information, the best way is to extract all paths connecting the pair. Thus greedy algorithms that only pick several connections are not suitable here. To avoid an infinite number of paths, cycles are not allowed. Since breadth-first search has exponential space complexity in order to remember all visited nodes, we use depth-first search. The parameter of this algorithm is the length limit d, i.e. the path at most consists of d relations among d+1 different consecutive objects. To find more explicit relations, two paths are treated different if nodes are same but the predicates are different, including the inverse property.

To the best of our knowledge, there is no well-known algorithm constructing this kind of connection subgraph. Algorithms for finding the shortest paths between all pairs of vertices are not applicable here, because we want to find all paths including non-shortest paths between a single pair. The solutions for graph reachability are also not appropriate since they only return a binary answer about the reachability between two nodes and do not record the nodes and edges connecting them. Bidirectional search could be an option, but it would need to keep the two sets of size $b^{d/2}$ each (b is the branching factor). So in addition of expansion cost, the extra complexity to get their intersection is $b^{d/2}$. Thus when d is small, the time efficiency saved during expansion is traded off by its postprocessing of sub-

```
Function getContext(front, dest, maxd, depth) returns a table DP representing the context
    Input: front is the frontier of the current path, dest is the destination, maxd is the max length of
    a path, depth is the distance from the frontier to the source
1   Let dist2dest = maxd – depth
2   IF depth < maxd
3       For each edge <front, prop, child> or <child, prop, front>
4           IF child = dest
5               DP[front, 1] ← <dest, prop>
6           ELSE
7               IF STATE[child, dist2dest - 1] ≠ DONE
8                   getContext(child, dest, maxd, depth + 1)
9               DP[front, d + 1] ← <child, prop>, □d, 0 < d < maxd and DP[child, d] is not empty
10      Set STATE[front, d] = DONE, □d, 0 < d ⩽ dist2dest
```

Figure 2. Algorithm of context construction.

graph building. Therefore, to better deal with scalability, we create a bottom-up dynamic programming algorithm which maintains a table recording all computed subpaths leading to the destination. Because only the nodes that can lead to the destination have entries in the table and only the path that can reach the destination are recorded, the table is sparse. And there is no postprocessing since the table is an encoding of the result subgraph.

The algorithm shown in Fig. 2 starts from the source as the frontier with depth zero, though there is no difference which URI is the source. During the search, it treats the same neighbor connected by different predicates as different neighbors in order to get all possible connections. The algorithm iteratively expands each neighbor of the current node. There are several situations when expanding neighbors. First (line 4-5), if this neighbor is the destination, it records the current node as being one step from the destination. Second (line 7-8), if the STATE table shows that this neighbor with certain distance away from destination is not expanded before, expand it. After this neighbor is expanded (line 9), record the subpaths from this neighbor as subpaths with one more step starting from the current frontier. After all neighbors of the current frontier are expanded (line 10), its state is set as DONE. The algorithm traverses all nodes once and the DP table records every subpath connecting the input pair, so it correctly returns all paths between them. The extracted context subgraph will be our primary input for two components in the system, because it carries sufficient entities and relationships related to the pair of instances being investigated.

4. INDICATORS OF SIGNIFICANT RELATION

Having the context for a triple, this section demonstrates the indicators of a significant relation (we call this component SR) between the pair of instances in a triple.

4.1. Class Distinctiveness

The indicator Class Distinctiveness (CD) is used to measure the information content of each node. In information theory, the amount of information contained in an event is measured by the negative logarithm of the probability of occurrence of the event. The amount of information gained or uncertainty removed by knowing that χ has the outcome x_i is given by I(χ = x_i) = -log Pr(x_i). For any class c ∈ C, the probability that χ = c is given by Pr(χ = c) = |c| / |I|, where I is the set of all instances. Then we define the CD as the average of the information content of all the URIs in the subgraph (shown in Definition 5), where $c(i)$ is the class type of instance i. The intuition is the contexts with more specific concepts are more precise.

Definition 5. Given a subgraph S = (I', E', R') of graph G = (I, E, R), the CD of S is defined as

$$\begin{aligned} CD(S) &= -\frac{1}{|I'|}\sum_{i\in I'} logPr(\chi = c(i)) \\ &= -\frac{1}{|I'|}\sum_{i\in I'} log\frac{|c(i)|}{|I|} \end{aligned} \quad (1)$$

4.2. Node Distinctiveness

The indicator Node Distinctiveness (ND) is used to measure how the nodes in the subgraph are special to this subgraph. A node is special to a subgraph if it is connected more strongly to nodes in this subgraph than to those outside of the subgraph. A node could be the subject or the object for a connection and be special when it is special on either one. So for each instance, we separately compute and average them to reflect the distinctiveness.

Definition 6. Given a graph G = (I, E, R) and an instance U ∈ I, in-degree of U w.r.t G is defined as In_G(U) = |{e|e = (s, p, U) and e ∈ E }| , and similarly out-degree of U w.r.t G is defined as Out_G(U) = |{e|e = (U, p, o) and e ∈ E }|.

Definition 7. Given a subgraph S = (I', E', R') of graph G = (I, E, R) and an instance U ∈ I', the Node Weight (NW) of U w.r.t S is defined as

$$NW(U,S) = \frac{1}{2}\left(\frac{In_S(U)}{In_G(U)} \times \frac{In_S(U)}{|E'|} + \frac{Out_S(U)}{Out_G(U)} \times \frac{Out_S(U)}{|E'|}\right) \quad (2)$$

Definition 8. Given a subgraph S = (I', E', R') of graph G = (I, E, R), the ND of S is defined as

$$ND(S) = \sum_{i\in I'} NW(i,S) \quad (3)$$

4.3. Predicate Distinctiveness

The indicator Predicate Distinctiveness (PD) is to measure how special the predicates are with respect to this subgraph.

Definition 9. Given a graph G = (I, E, R) and a predicate P $\in$ R, the number of edges of P is Edges_G(P) = $|\{e|e = (s, P, o)$ and e $\in$ E$\}|$; the number of distinct subjects of P is Sub_G(P) = $|\{s|e = (s, P, o)$ and e $\in$ E$\}|$; the number of distinct objects of P is Obj_G(P) = $|\{o|e = (s, P, o)$ and e $\in$ E$\}|$.

Definition 10. Given a subgraph S = (I', E', R') and a predicate P $\in$ R', the Predicate Weight (PW) of P w.r.t S is defined as

$$PW(P,S) = \frac{Edges_S(P)}{Edges_G(P)} \times \frac{Sub_S(P) + Obj_S(P)}{Sub_G(P) + Obj_G(P)} \quad (4)$$

Definition 11. Given a subgraph S = (I', E', R'), $r_i \in$ R', 0 $<$ i, the PD of S is defined as

$$PD(S) = |R'| \sum_i \frac{(PW(r_i,S) \times Edges_S(r_i))}{\sum_i Edges_S(r_i)} \quad (5)$$

There are several considerations for designing the PD. First, how much the percentage of usages of each predicate is within the subgraph. Second, how many distinct subjects and objects of each predicate are used in the subgraph. Third, the variety of subjects and objects should be considered together. Because some predicates have very few distinct subjects or objects but many more of the other, like citizenship, the variety of subject values would make it seems distinctive if we separate it from the object values. Fourth, the sum of all the predicates are weighted based on each predicate's contribution to number of the edges in the subgraph. Fifth, using the number of distinct predicates as a factor can compensates some extreme cases. For example, the subgraph only has two edges connecting a third node with the target pair; these two edges are the only two usages of a predicate and the only links in knowledge base (KB) for these three nodes. So based on previous definition, we know PD and ND of this subgraph both be one. However from context perspective, the context lacks a variety of relations and hence sufficient evidence.

5. PATTERNS OF RELATION

After introducing the indicators of significant relations, here we give the algorithm to check if the relation type (we call this component RT) entailed by the context agrees with the predicate in the triple.

5.1. Definition of Patterns

Because the entailment is on a per predicate basis and the number of predicates is relatively small compared to that of the instances or the triples. So the algorithm essentially utilizes predicate co-occurrence which is often used in ontology alignment and coreference resolution. The idea is that if a context shows patterns of predicate usage that also appeared in contexts around other pairs, then the relation between those pairs are probably similar to this relation.

The input of the algorithm is the context introduced in section 3. The patterns for a predicate consist of predicates extracted from contexts of triples with this predicate, but they should not be treated simply as a bag of predicates. The first reason is that the same set of predicates would reflect different meaning if the order of their usages is different. Second, different join conditions among triples convey different interconnecting semantic patterns and relations between two end points. For instance, we have four triples: $\langle$A1, studentOf, B1$\rangle$, $\langle$B1, advisorOf, C1$\rangle$, $\langle$A2, studentOf, B2$\rangle$ and $\langle$C2, advisorOf, B2$\rangle$. The sequence of the first two predicates is the same as that of the last two predicates. But A1 and C1 are connected because they are both students of B1 while A2 and C2 are connected because A2 is academic descendant of C2. The two relations are totally different. Considering the points above, we define predicate patterns below. We use traditional inverse property representation to indicate that the triple of this predicate is joined via object with previous triple and via subject with next triple.

Definition 12. A single pattern for a predicate pr is p(pr) = $\langle r_1, r_2, ..., r_n\rangle$, where $\exists I_0, I_1, ..., I_n$, such that $\langle I_0, r_1, I_1, r_2, I_2, ..., r_n, I_n\rangle \in \text{PATHS}_G(I_0, I_n)$ and $(I_0, pr, I_n) \in$ E.

Definition 13. The pattern template for a predicate Patt(pr) = $\{(p_i, w_i) \mid p_i$ is a single pattern for pr, w_i is the number of instantiations of this single pattern in the contexts for all triples divided by the inverted frequency of predicates that have this single pattern and $\sum w_i = 1.\}$

5.2. Pattern Matching

The pattern weight is based upon the number of instantiations of patterns in all triples. If we make an analogy between a predicate and a document class, the patterns and the terms respectively, the pattern weight is similar to a tf/idf term weight used in information retrieval. In tf/idf, a term is weighted as the number of its usages in a document divided by the inverted frequency of documents that contain the term. Similarly a pattern (Definition 13) is weighted as the number of its instantiations divided by the inverted frequency of predicates that have this pattern.

$$pred(t) = \max_{pr \in R} \sum_{<p_i, w_i> \in Patt(pr)} w_i \times match(t, p_i) \quad (6)$$

At runtime, we extract the patterns appearing in the context of the triple. Then we match the extracted patterns with the learned patterns. Equation 6 describes the criterion for the best matched predicate for triple t. The best predicate has the largest sum of matched pattern weights. In the equation, the match() function is used as an boolean variable: whether the triple's context has this pattern and w_i is the weight of this pattern. The matching complexity is Θ(mn), where m is the number of patterns from the target triple and n is the total number of patterns in KB.

6. RESULTS AND EVALUATION

In our experiments, we selected SWRC data set which has 100K triples and 67K resources and DBPedia infobox data set which has 10M triples and 3M resources. They are widely used and from different domains.

6.1. Experimental Setup

Because the reference data is assumed or verified generally correct and comprehensive, when training, the system uses some existing triples for each predicate as posistive examples and all other triples neither in nor can be inferred from the original data set as incorrect. Thus this training process basically is an unsupervised learning, since there is no labeled data as input. When testing, for each predicate, we randomly pick 200 random triples which are not in training set as positive test examples. Because almost no data sets have the negation of triples (recently OWL 2 [2] added this function), the negative triples used in test are generated through the following process. For each predicate used in positive examples, we create a domain set consisting of all the distinct subjects of positive example 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 that the synthetic triple still conforms to the ontologies of this data set. Otherwise it would be trivial to find that it is suspect. Finally if the generated triple neither is in nor can be inferred from the original data set, it is qualified as a negative example. To verify the reliability of test set, four Semantic Web experts verified randomly sampled data by using a simple interface through which they can explore relevant triples in the knowledge base and Sindice[3], a popular Semantic Web search engine. The experts verified that all the positives are correct and all the negatives are incorrect.

[2]http://www.w3.org/TR/owl2-overview/

[3]http://www.sindice.com

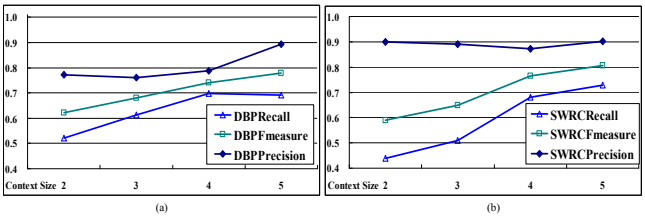

Figure 3. (a)(b) Results of the component SR on determining significant relations on SWRC and DBPedia.

The training process is to establish the parameters of probabilistic rules. The process to get the pattern weights used to entail the predicate is introduced in section 5.2. To get the weights of indicators for the existence of a significant relation, we compute three indicator values for every training triple and put them into a classifier as feature values of these triples. To avoid bias, we removed the original direct links between the pair of objects in positive triples for all experiments so that both positive and negative triples are unknown to the system. We compared the performance of several popular classifiers, such as decision tree, naive Bayesian, kth nearest neighbor and binomial logistic regression. Overall, the decision tree is the best based on the time and performance, though others are not far behind.

The experiments are primarily designed as follows. Given a reference data set that is generally correct and conform to certain ontologies, we check if the system can make distinctions between ordinary triples and abnormal triples in the test set that conform to the same ontologies by applying learned probabilistic rules on their contexts. Component SR and RT are separated to test if each functions well.

6.2. Results

The first group of experiments show the performance on determining the existence of significant relations when context size (defined in Definition 4) varies. Both test sets consist of equal number of positive and negative examples. The results reflect similar trend on two data sets (Fig. 3(a)(b)). The precision does not drop much when the context size decreases. The reason is when subgraph is smaller or context information is fewer, the links in the contexts of both positive and negative samples are easily broken in negative contexts. So it's harder for the negatives to have a well clustered supporting evidence that is necessary to be classified as positive. Thus the false positives become fewer when the context shrinks. Similarly due to the removed links in positive contexts, some of the

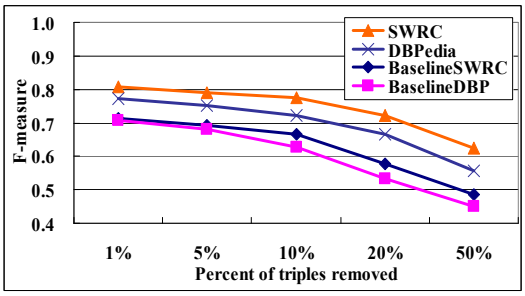

Figure 4. Impact of less complete data on the systems ability to detect significant relations.

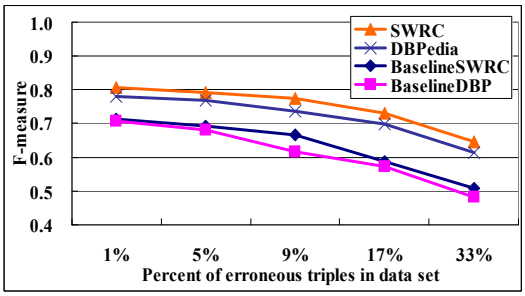

Figure 5. Impact of erroneous data on the systems ability to detect significant relations.

positive examples will lose some clues for the system to determine positive when the context shrinks. So the number of false negative increases and the recall drops more than precision. In addition, we notice that the improvement on recall on SWRC data set is more than that on DBPedia data set when context size increases. We believe the reason is that SWRC data set has more relational descriptions among instances, specifically the average density (number of edges divided by square of number of nodes) of context graphs on SWRC data set is around five times of that on DBPedia data set. So it gains better clustered descriptions when allowed path length is larger. We did not show bigger context size here, because when the size is bigger than five, the number of nodes and edges in the context do not increase much, specifically the increase on the number of nodes from five to six is less than 10% on two data sets.

To check how the system relies on the assumption on the reference data, the second group of experiments checks how system performs when some random triples in reference data are removed (Fig. 4) and when reference data has some erroneous triples (Fig. 5). Removing triples gives two aspects of impact to the system. One is that the context information becomes less for all triples and the other is that some missing triples that the system assumes incorrect are factual true. We see that when lots of triples are removed, the system still can give decent performance. Similarly when 10% of triples are incorrect, the performance only drops several percent. Comparing Fig. 4 and 5, the effect of erroneous triples is not as much as that of the triples removed. The reason is that the learning is based on the agreement among the majority of the data. Some erroneous triples would probably incur some patterns that others hardly agree with, while removing triples makes many agreed patterns disappeared or vague. Since our component SR for determining significant relation is similar to the link prediction in SNA, we also compared with a baseline classifier. Among popular link predictors in SNA, such as jaccard, katz weighted, katz unweighted, common neighbors and preferential attachment [4], we observed that the baseline using katz weighted and prefential attachment is almost as good as using all predictors here and so the baseline in Fig. 4 and 5 uses these two key predictors. Our system is much better than it.

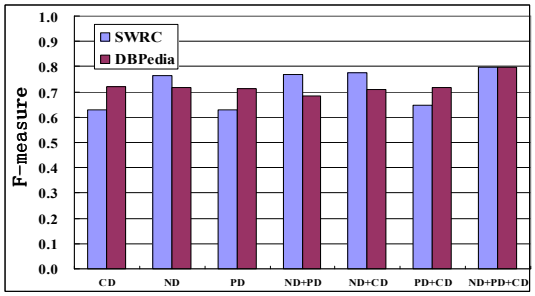

Figure 6. Results of the component SR using subsets of indicators on two data sets.

The third group of experiments is to check the component SR with different subsets of indicators (shown in Fig. 6). We see that only using the ND can get better performance than the other two on SWRC, while only using the PD or CD is better in DBPedia. Combinations of two indicators are better than using single ones and none of them can dominate the results over different domains. Finally the combination of the three is the best. It proves that three indicators capture different aspects of a context.

In the last group of experiments (Fig. 7), we tested the component RT to check if the probabilistic rules are useful to determine the relation type between the pair of objects in a triple. If the system can accurately determine the best predicate for their relation, the system can also differentiate the triple with the correct predicate from that with an incorrect predicate. In this experiment, the positive results from the SR are then used to evaluate the correctness of the predicate. Thus we should note that the performance of this component is affected by that of SR. In order to understand our system's capabilities, we compare it to a base-

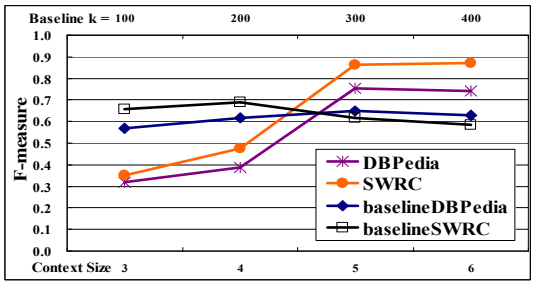

Figure 7. Comparison between component RT and the baseline on determining relation types on two data sets.

line system based on predicate suggestion systems. For an instance, such systems usually find similar instances and suggest some predicates used on those instances but not on this instance [5]. If two objects in a triple have similar instances respectively, the predicates connecting these two groups of instances could also be the predicates between them. So the baseline system is built by finding a set consisting of the top kth similar instances for each object and then picking the top ranked predicate connecting these two sets of instances as the predicted relation. The similarity between instances is measured by counting the number of same predicate and object pairs. In this experiment, we are comparing the best performance of each system when each system's variable changes. The figure shows that although RT performs worse than the baseline for context sizes of three and four, when the context size is bigger than five, it performs better than the best configuration of the baseline.

7. CONCLUSION

We have presented an approach to detect abnormal triples that are probably incorrect within a knowledge context. This technique is important for both ontology based information integration and query answering. The essential idea is to use statistical information about the context of a triple and the context of typical triples to generate a measure of abnormality. One component suggests if a significant relation could exist between two resources by using quantified indicators for the context, and the other component assesses what the best predicate is to describe the relationship. The technique is general, since the approach is mainly based on the data itself without complex ontological inference and only requires a set of unlabeled data sources that are generally correct. The experiments show that the system performs well on detecting both types of abnormal triples in data sets from different domains. In the future we plan to account for literals and datatype properties and to integrate the two components because they have useful information for each other. Additionally, we will try to improve the recall when there are few indirect semantic connections between a pair of objects. Furthermore we will utilize our algorithms on other problem domains, like knowledge discovery and inconsistency solving.

REFERENCES

[1] C. Bizer and R. Cyganiak, "Quality-driven information filtering using the wiqa policy framework," Web Semantics., vol. 7, pp. 1-10, January 2009.

[2] C. Furber and M. Hepp, "Using semantic web resources for data quality management," 17th International Conference on Knowledge Engineering and Knowledge Management, EKAW 2010, Lisbon, Portugal, pp. 211-225, 2010.

[3] D. Caragea, V. Bahirwani, W. Aljandal, and W. H. Hsu, "Ontology-based link prediction in the livejournal social network," Eighth Symposium on Abstraction, Reformulation, and Approximation, Lake Arrowhead, USA, 8-10, August 2009.

[4] D. Liben-Nowell and J. Kleinberg, "The link prediction problem for social networks," in Proceedings of the twelfth international conference on Information and knowledge management, CIKM' 03. New York, NY, USA: ACM, pp. 556-559, 2003.

[5] E. Oren, S. Gerke, and S. Decker, "Simple algorithms for predicate suggestions using similarity and co-occurrence," in Proceedings of the 4th European conference on The Semantic Web, ESWC' 07, Innsbruck, Austria, pp. 160-174, 2007.

[6] F. Naumann, "Quality-driven query answering for integrated information systems," Berlin, Heidelberg: Springer-Verlag, 2002.

[7] J. Tao, L. Ding, and D. L. McGuinness, "Instance data evaluation for semantic web-based knowledge management systems," in 42nd Hawaii International Conference on System Sciences, Waikoloa, Big Island, Hawaii, pp. 1-10, 2009.

[8] L. Getoor and C. P. Diehl, "Link mining: a survey," SIGKDD Explor. Newsl., vol. 7, pp. 3-12, December 2005.

[9] M. Sabou, M. Fernandez and E. Motta, "Evaluating semantic relations by exploring ontologies on the Semantic Web," In: 14th International Conference on Applications of Natural Language to Information Systems, Saarland, Germany, 23-26, June 2009.

[10] O. Hartig and J. Zhao, "Using web data provenance for quality assessment," The 1st International Workshop on the role of Semantic Web in Provenance Management, Washington D.C., USA. 2009.

[11] S. Lin and H. Chalupsky, "Unsupervised link discovery in multirelational data via rarity analysis," in Proceedings of the Third IEEE International Conference on Data Mining, ICDM' 03. Washington, DC, pp. 171-178, 2003.

[12] Y. Lei and A. Nikolov, "Detecting Quality Problems in Semantic Metadata without the Presence of a Gold Standard," the 5th International Workshop on Evaluation of Ontologies and Ontology-based Tools, Busan, Korea, 51-60, November 2007.