Noisy Type Assertion Detection in Semantic Datasets

Man Zhu, Zhiqiang Gao, and Zhibin Quan

School of Computer Science & Engineering, Southeast University, P.R. China
Key Laboratory of Computer Network and Information Integration,
Southeast University, Ministry of Education, P.R. China
{mzhu,zqgao,zbquan}@seu.edu.cn

Abstract. Semantic datasets provide support to automate many tasks such as decision-making and question answering. However, their performance is always decreased by the noises in the datasets, among which, noisy type assertions play an important role. This problem has been mainly studied in the domain of data mining but not in the semantic web community. In this paper, we study the problem of noisy type assertion detection in semantic web datasets by making use of concept disjointness relationships hidden in the datasets. We transform noisy type assertion detection into multiclass classification of pairs of type assertions which type an individual to two potential disjoint concepts. The multiclass classification is solved by Adaboost with C4.5 as the base classifier. Furthermore, we propose instance-concept compatability metrics based on instance-instance relationships and instance-concept assertions. We evaluate the approach on both synthetic datasets and DBpedia. Our approach effectively detect noisy type assertions in DBpedia with a high precision of 95%.

1 Introduction

Real world data is never as perfect as we would like it to be and can often suffer from corruptions that may impact interpretations of the data, models created from the data, and decisions made based on the data [1][2]. *Accuracy*, *relevancy*, *representational-consistency* and *interlinking* affect approximately 11.93% of DBpedia[1] resources. Among them, the detection of accuracy problem is the least to be automated [3]. We are interested in the factual errors (called noises in this paper) in the accuracy category. To be specific, we focus on the detection of noisy type assertions (asserting *Schubert's last sonatas* is of type *artist* for example), which is suggested to be more severe than noisy property assertions (asserting TV series *Wings*'s *opening theme* is *Schubert's last sonatas* for example) [4].

While there has been a lot of research on noise identification in data mining domain in the past two decades, the topic has not yet received sufficient attention from the Semantic Web community, especially the problem of noisy type detection. Zaveri et al. [3] analysed empirically the DBpedia dataset. They manually evaluated a part of indiviual resources, and semi-automatically evaluated the quality of schema axioms. Fürber and Hepp [5] summarized the important problems in semantic web data, including literal value problems and functional dependency violations, and correspondingly

[1] http://dbpedia.org

P. Mika et al. (Eds.) ISWC 2014, Part I, LNCS 8796, pp. 373–388, 2014.

developed SPARQL queries to identfy them. Yu et al. [6] focused on identifying noisy property assertions. They detected such assertions by using probabilistic rules learned from semantic web data and checked to what extent the rules agree with the context of assertions.

We find that noisy type assertions could be detected from knowledge hidden in real-world datasets.

Example 1. *If we execute the following SPARQL query in DBpedia*

```
select ?x where{?x a dbpedia-owl:Person.
                ?x a dbpedia-owl:Place.}
```

which selects individuals belonging to both concept `Person` *and* `Place`*, we get a list of individuals returned, such as* `Pope`[2]*. Because we, as human-beings, believe that concept* `Person` *and* `Place` *share no individuals, which is hidden in DBpedia because* `Person` *and* `Place` *share a very small number of individuals, it is reasonable to guess that the assertions typing the individuals to concept* `Person` *or* `Place` *are problematic.*

In this paper, we study the problem of noisy type assertion detection in semantic web datasets for the first time. Roughly speaking, our approach contains 2 steps: Firstly we cache the number of individuals belonging to a pair of concepts aiming at detecting abnomal data. We extract conflicts such as `Pope` belongs to both `Person` and `Place`. After that, we transform the detection of noisy type assertions into a multiclass classification problem, where a candidate conflict assertion can be labeled (1) none of them are noisy; (2) first assertion being noisy; (3) second assertion being noisy; (4) both of them are noisy. The conflicts are classified by Adaboost with decision tree algorithm C4.5 as the base classifier. In order to characterize the conflict assertions, we propose two kinds of features: First kind of features make use of type assertions. For example, the assertions "`Pope` is a `Cleric`" and "`Cleric` is subsumed by `Person`" increase the confidence of assertion "`Pope` is a `Person`" Another kind of feature utilizes role information, which we borrowed from [2]. For example, several individuals are linked with `Pope` by role `beatifiedBy`, and from the dataset, `beatifiedBy` is always connected with a `Person`, then "`Pope` is a `Person`" is more probable. To summarize, the main contributions of this paper are to:

- study the novel problem of noisy type assertion detection in semantic web datasets;
- formalize the noisy type assertion detection problem as a multiclass classification problem for the first time;
- propose various effective compatibility metrics that incorporate both concept and role relationships.

The rest of the paper is organized as follows. Section 2 describes decision tree (C4.5) and Adaboost classification algorithm. In Section 3, we motivate the approach in section 3.1 by analyzing the co-occurrence data in DBpedia, and then we formalize the research problem and introduce the framework. Section 4 details the approach focusing on the features. The experimental results are presented in section 5. Section 6 introduces related work, and section 7 concludes the paper and gives future works.

[2] In DBpedia 3.9 there are 17 individuals belonging to both `Person` and `Place`.

2 Decision Tree and Adaboost

We use Adaboost as the meta classifier with C4.5, a popular decision tree algorithm, as the base classifier. Decision tree (DT) is a set of if-then rules representing a mapping between data features and labels. Each internal node in a DT indicates a feature, and each leaf node represents a label. We adopt DT as the base classifier for the following reasons: (1) DT is a white-box model, which is easy to be understood and interpreted; (2) Rule is the suitable representation for the features proposed in this paper.

DTs can be inductively learned from training data. C4.5 is a popular DTs learning algorithm [7]. It builds decision trees from a set of training data using information entropy by divide-and-conquer. At each node of the tree, C4.5 chooses the attribute of the data that most effectively splits the examples into subsets by normalized information gain. The attribute with the highest normalized information gain is chosen. The initial tree is then pruned to avoid overfitting [8].

In order to improve the performance of classification algorithms, boosting iteratively learns a single strong learner from a set of base learners. There are many variations of boosting algorithms varying in their method for weighting training data and classifiers. Adaboost [9] uses an optimally weighted majority vote of meta classifiers. More concretely, the impact on the vote of base classifiers with small error rate is intensified by increasing their weights. The label of a data instance is predicted by the linear combination of meta classifiers, in our case, DTs, as follows:

$$T(x) = \sum_{m=1}^{M} \alpha_m T_m(x) \tag{1}$$

where M DTs are learned, α_m is the weight of the mth DT, and $T_m(x)$ is the output of the mth DT.

3 Approach

In this section, we firstly motivate our approach by a co-occurrence analysis on DBpedia. Then we formalize the research problem and describe the framework.

3.1 Co-occurrence Analysis on DBpedia

Before we analyse the co-occurrence on DBpedia, we first give the definition of co-occurrence matrices as follows:

Definition 1 (Co-occurrence Matirx). *A co-occurrence matrix M is a symmetric matrix defined over a semantic dataset $\mathcal{O}$. Mathematically, a co-occurrence matrix $M_{N\times N}$ is defined over N concepts $\mathbf{C}$ in $\mathcal{O}$, where $M_{st} = |\{i|C_s(i) \in \mathcal{O}$ and $C_t(i) \in \mathcal{O}, C_s \in \mathbf{C}, C_t \in \mathbf{C}\}|$.*

We take 90 concepts in DBpedia containing at least 10,000 individuals, and sort them in descending order in terms of the number of individuals they have. The values in

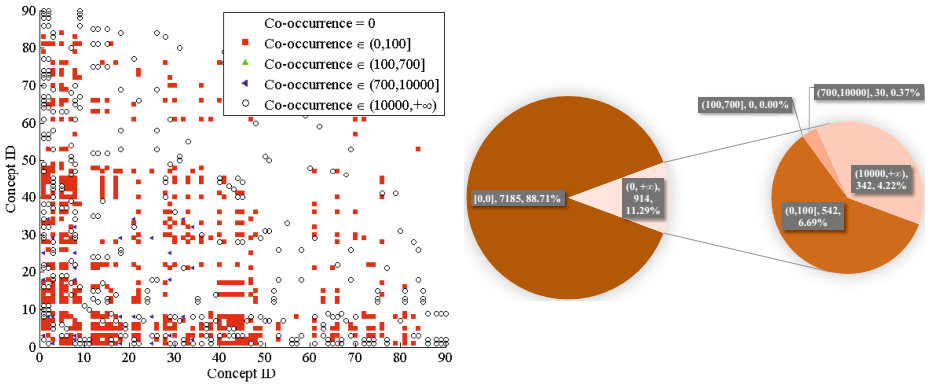

Fig. 1. Co-occurrence matrix of the top-90 concepts in terms of individuals they have in DBpedia. Left shows the co-occurrence values for each pair of concepts. The figure on the right represents the frequency of co-occurrence values in different scopes.

the co-occurrence matrix are retrieved by executing SPARQL queries as shown in Section 1.

The left in Fig. 1 shows directly the co-occurrence matrix. We can easily find from this figure red squares and black circles representing co-occurrence values below 100 and above 10,000. However, the numbers in between, represented by triangles, are quite rare. The numbers above 10,000 indicate highly overlapped concepts, while the numbers below 100, on the other hand, suggest abnormal data. The figure on the right shows the percentage of co-occurrence numbers varying scopes. Besides the largest amount of zero filling the co-occurrence matrix, more than half of the other numbers are below 100 (6.69% in 11.29%), which suggests that the amount of abnormal data can not be ignored and the noisy type assertions can be detected from them. If we take a closer look of the concept pairs sharing less than 100 individuals, we can find, for example, (Person Place), (Person Work), (Place Work), (Place Athlete) etc. These concepts, according to human knowledge, should share no individuals at all.

3.2 Problem Definition

We detect noisy type assertions through conflict in the semantic datasets, which is defined as follows:

Definition 2 (Conflict Type Assertions). *A pair of type assertions* $A(i)$ *and* $B(i)$ *is called conflict if* $A \sqcap B \sqsubseteq \bot$*, written as* $< i, A, B >$*, where* i *is called the* target individual. *A conflict* $< i, A, B >$ *is called* full noisy *if* $A(i)$ *and* $B(i)$ *are both noisy;* 1-st half noisy *if only* $A(i)$ *is noisy;* $< i, A, B >$ *is called* 2-nd half noisy *if only* $B(i)$ *is noisy; It is called* fake conflict *if none of* $A(i)$ *and* $B(i)$ *are noisy.*

where $A \sqcap B \sqsubseteq \bot$ means concept A and B are disjoint. Explicitly asserting individuals to A and B will cause problems, if $A \sqcap B \sqsubseteq \bot$. We make use of $A \sqcap B \sqsubseteq \bot$ *hidden* in the datasets. Without ambiguity, conflict type assertions are called conflicts for short.

According to the definition of conflict type assertions, noisy type assertion detection from conflicts can be formalized as a multiclass classification problem.

Definition 3 (Noisy Type Assertion Detection From Conflicts). *Given a set of conflict type assertions* $\{< i, A, B >\}$*, the goal of noisy type assertion is to find a classifier* $\mathcal{M} :< i, A, B >\rightarrow \{0, 1, 2, 3\}$ *such that* $\mathcal{M}$ *maps the full noisy conflict to class* 0*, 1-st half noisy to class* 1*, 2-nd half noisy to* 2*, and maps fake conflict to class* 3*.*

The multiclass classification problem can be solved by traditional machine learning algorithms, which require multidimensional features as the input. In noisy type assertion detection, we extract a feature vector for each conflict type assertion.

Definition 4 (Feature Vector of Conflict Type Assertions). *The n-dimensional feature vector v of a conflict type assertion* $< i, A, B >$ *consists of n various compatibility metrics of individual* i *with concepts* A *and* B*. Dimension* $v_i = d_i < i, A, B >$*, where* d_i *is the ith compatibility metric function for* $< i, A, B >$*.*

The feature vector of a conflict type assertion indicates the compatibility of an individual i and a pair of concepts, which are computed by several metric functions.

3.3 Framework

We observe that (1) due to the dataset enrichment mechanisms or data intrinsic statistics, when concepts share instances, they generally share a large portion of instances even compare to the number of instances they have themselves; (2) when two concepts share a small amount of instances (in another word, the concepts are suggested to be disjoint according to the data), there tend to be noises inside. Based on these observations, we propose to identify noisy types from conflict type assertions. The framework contains the following 5 steps (cf. Fig. 2):

(1) *Co-occurrence matrix construction*. In this step, we construct the co-occurrence matrix. The values in the co-occurrence matrix signify the relationship between the corresponding concept pair. For example, concepts `Person` and `Place` have 17 instances in common as shown in Fig. 2. Suppose the probability of concepts A and B being disjoint $P(A \sqcap B \sqsubseteq \bot)$ is $1 - P(A \sqcap B) = 1 - |\{a|A(a) \in \mathcal{O}, B(a) \in \mathcal{O}, \top(a) \in \mathcal{O}\}|/|\{a|\top(a) \in \mathcal{O}\}|$, $\mathcal{O}$ is the semantic dataset. If the cooccurrence is very small, the probability that the related concepts being disjoint is relatively large. If we are confident about them being disjoint, then the assertions of instances belonging to both concepts contain problems. The calculation of co-occurrence matrix includes executing $N \times N/2$ SPARQL queries, where N is the number of concepts in the dataset.

(2) *Conflict type assertion generation*. Based on the cooccurrence matrix and by setting threshold, we generate disjoint concepts. By querying the dataset for list of instances belonging to each pair of disjoint concepts, the conflict type assertions are generated. For example, instances $I(A, B)$ belong to disjoint concepts A and B, then $\forall i \in I(A, B)$, we add the triple $< i, A, B >$ to the conflict set.

(3) *Feature extraction*. We generate a feature vector for each conflict type assertion. The details of the compatibility metrics are described in Section 4. We cache all intermediate statistics required to calculate the metrics in a local relational database. Scan the relational database once will get the feature vectors.

(4) *Classification algorithm*. We use Adaboost with C4.5, a well-known classification algorithm, as the base learner to classify the conflicts.

(5) *Classification results*. From the classification results, which contain conflicts belonging to class 0, 1, 2, and 3, we output the final noisy type assertions by seperating conflict with class 0 into two noisy type assertions, output conflicts with class 1 or 2 into one noisy type assertion. To be specific, suppose the conflict is $< i, A, B >$, if its label is 0, $A(i)$ and $B(i)$ are added to the final results; if its label is 1, $A(i)$ is added, and similarly, if the label of the conflict is 2, $B(i)$ is added.

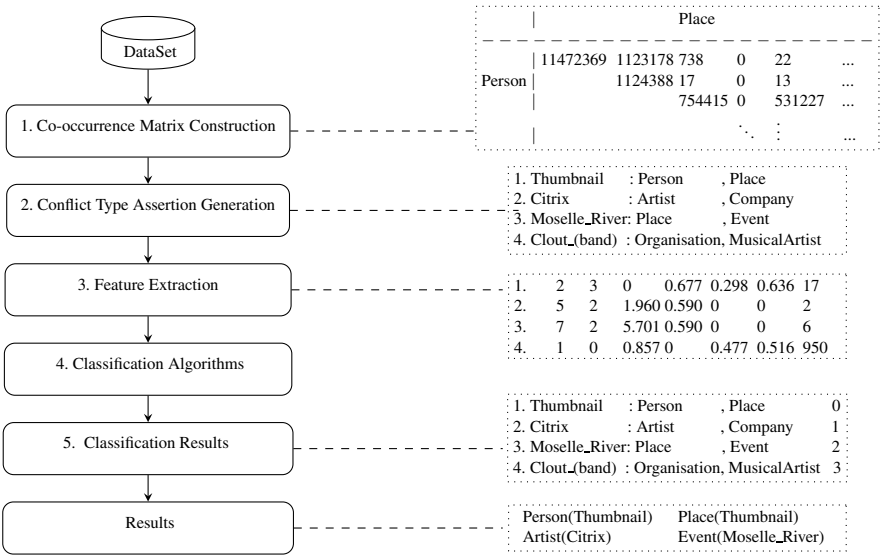

Fig. 2. Overview of the framework

4 Feature Extraction

The compatibility metrics in the feature vector of a conflict type assertion are based on the type assertions and property assertions of the target individual. In this section, we first introduce the weight functions of predicates. Then we describe the details of compatibility metrics in the feature vector.

4.1 Weighted Predicates

The importance of predicates (concepts or roles) playing in classifying conflicts may be different, especially in imbalanced datasets where the number of individuals belonging to different concepts are not approximately equally distributed. Paulheim and Bizer ([2]) defined weight of object properties. In this paper, we extend the weight to predicates defined as follows:

$$w_p := \sum_{\text{all concepts } C} (P(C) - P(C|p')) \quad (2)$$

where

$$p' = \begin{cases} p & \text{if } p \text{ is a concept;} \\ \exists p.\top & \text{if } p \text{ is an object property.} \end{cases}$$

and $P(C) = |\{a|C(a) \in \mathcal{O}\}|/|\{a|\top(a) \in \mathcal{O}\}|$, $\mathcal{O}$ is the dataset. Additionally, the weight of concept $\exists p.\top$ is written as w_p, and the weight of $\exists p^-.\top$ is written as w_{p^-}.

4.2 Compatibility Metrics

We motivate the first kind of features by the following example.

Example 1 Revisited. Let us consider the conflict <Pope, Person, Place>. We want to use compatibility metrics to characterize individual Pope with respect to concept Person and Place. In the dataset, besides Person and Place, Pope also belongs to Agent. We know that Person is subsumed by Agent, and Place is not. Then we are more confident about "Pope is a Person". We simply compute the number of concepts of this kind, such as Agent, and call this feature *supSup* (super support), as shown in Table 1, where $A \sqsubset^+ A'$ means A is indirectly subsumed by A'. Similarly, "Pope is a Cleric" and "Cleric is subsumed by Person also increases the confidence of "Pope is a Person". Based on the subsumed concepts, we define the feature *subSup* (subclass support). Another feature is calculated based on the equivalent concepts (equivSup), such as Pope is asserted to be of type a:Person, an equivalent class of Person. This kind of features is called plain concept related features. The calculation of the concept related features includes transitive subsumption relationships, which can be achieved for example from Virtuoso by:

```
SELECT count(?x) AS ?count WHERE{{
SELECT * WHERE {dbpedia:i    a                    ?x.
                {?x          rdf:type             ?y.} UNION
                {?x          owl:equivalentClass  ?y.} UNION
                {?y          owl:equivalentClass  ?x.}}}
OPTION (transitive, t_distinct, t_in (?x), t_out (?y)).
FILTER (?y=dbpedia-owl:A)}
```

However, the contributions of predicates can be different, as we discussed in Section 4.1. We propose two kinds of features to incorporate the differences. Firstly we simply compute the linear combination of all weights of the predicates related to the target individual by setting the coefficients to be 1. This kind is called *simple weighted concept related features*. Let us consider the following cases to motivate the second kind: Cleric is subsumed by Person and a:Person is equivalent with Person. If the individuals belonging to concept Person are always of type a:Person, the contribution of a:Person is lower than that of Cleric in classifying conflict <i, Person, Place>, if there are a lot of differences between individuals belonging

Table 1. Features used in the classification

Plain concept related features		
Name	Definition	Type
subSup(i, A)	$\|\{A' \| A' \sqsubset^+ A, \text{ and } A'(i) \in \mathcal{O}\}\|$	numeric
supSup(i, A)	$\|\{A' \| A \sqsubset^+ A', A' \not\equiv \top, \text{ and } A'(i) \in \mathcal{O}\}\|$	numeric
equivSup(i, A)	$\|\{A' \| A' \equiv^+ A, \text{ and } A'(i) \in \mathcal{O}\}\|$	numeric
Simple weighted concept related features		
Name	Definition	Type
simpleWSubSup(i, A)	$\sum_{A' \sqsubset^+ A, A'(i) \in \mathcal{O}} w_{A'}$	numeric
simpleWSupSup(i, A)	$\sum_{A \sqsubset^+ A', A' \not\equiv \top, A'(i) \in \mathcal{O}} w_{A'}$	numeric
simpleWEquivSup(i, A)	$\sum_{A \equiv^+ A', A'(i) \in \mathcal{O}} w_{A'}$	numeric
Weighted concept related features		
Name	Definition	Type
wSubSup(i, A)	$\nu_1 \sum_{A' \sqsubset^+ A, A'(i) \in \mathcal{O}} w_{A'}(1 - P(A'\|A))$	numeric
	$\nu_1 = 1/\sum_{A' \sqsubset^+ A, A'(i) \in \mathcal{O}} w_{A'}$	
wSupSup(i, A)	$\nu_2 \sum_{A \sqsubset^+ A', A' \not\equiv \top, A'(i) \in \mathcal{O}} w_{A'}(1 - P(A\|A'))$	numeric
	$\nu_2 = 1/\sum_{A \sqsubset^+ A', A' \not\equiv \top, A'(i) \in \mathcal{O}} w_{A'}$	
Role related features		
Name	Definition	Type
attrSup(i, A)	$\nu_3 \cdot \sum_{\text{all roles r of i}} w_r \cdot P(A\|\exists r.\top)$	numeric
(Paulheim and Bizer [2])	$\nu_3 = 1/\sum_{\text{all roles r of i}} w_r$	

to `Cleric` and that of `Person`'s. This is because the type assertion of `Pope` being a `a:Person` probably due to the mechanisms in constructing the dataset. For this reason, we propose to give weight to the subclass of concept A, A', defined as $(1 - P(A'|A))$. Similarly the weight of the super class of concept A, A', is defined as $(1 - P(A|A'))$. We use the compatibility metric of property assertions as defined in (Paulheim and Bizer [2]). There will be two numbers in the feature vector for each metric listed in Table 1. One calculates the compatibility metric of the first concept in the conflict, and the other one computes the metric of the second concept.

5 Experimental Evaluations

We conduct the evaluations on synthetic datasets and DBpedia. The questions we want to answer using synthetic datasets are: (1) How does the proposed approach work in the semantic web context? (2) What are the advantages and drawbacks of the proposed approach? By applying the proposed method on DBpedia, we show the effectiveness of the proposed method by manually checking the correctness of the detected triples.

5.1 Experimental Settings

For each experiment, we perform 10-fold cross-validation. We use the precision, recall, F1 scores defined as follows:

$$precision = \frac{\text{\# correctly detected noisy type assertions}}{\text{\# detected noisy type assertions}} \tag{3}$$

$$= \frac{2TP_0 + TP_1 + TP_2}{2TP_0 + 2FP_0 + TP_1 + FP_1 + TP_2 + FP_2} \tag{4}$$

$$recall = \frac{\text{\# correctly detected noisy type assertions}}{\text{\# noisy type assertions}} \tag{5}$$

$$= \frac{2TP_0 + TP_1 + TP_2}{2TP_0 + 2FN_0 + TP_1 + FN_1 + TP_2 + FN_2} \tag{6}$$

where TP_0, TP_1, TP_2 are the number of true positives of label 0, 1, and 2 respectively, FP_0, FP_1, FP_2 are the number of false positives of label 0, 1 and 2, and FN_0, FN_1, FN_2 are the number of false negatives of label 0, 1 and 2 respectively. F1 score is the harmonic mean of precision and recall, which is calculated by $2 \times \frac{precision \times recall}{precision + recall}$. In terms of the performance of the classifier, we use average accuracy as the final results. In terms of the classifier implementation, we use Adaboost and J48 - the Weka 3[3] implementation of C4.5. We set the weight threshold of Adaboost to 100, and number of iterations to be 10. We also use resampling. All the experiments are carried out on a laptop computer with Ubuntu 12.04 64-bit with a i7 CPU, 8 GiB of memory.

Feature Schemes. We use different combinations of features described in this paper in the evaluations. The details of the compositions are as follows:

CS: use subSup, supSup, and equivSup features;
WCS: use wSubSup, wSupSup features;
SWCS: use simpleWSubSup, simpleWSupSup, and simpleWEquivSup as features;
AS: only use attrSup in the feature vector;
ALL: use all features.

5.2 Evaluations on Synthetic Datasets

In order to control the amount of noise in the synthetic dataset, we construct datasets containing noises based on LUBM [10] dataset, which is an automatically constructed dataset without any noises in the assertions. LUBM consists of 43 concepts, 25 object properties, 36 subClassOf axioms, 6 equivalentClass axioms, 1555 individuals. We use LUBM in order to get the full control on the noises, and we can also get a benchmark dataset.

Noise Control Strategy. A type assertion $A'(a)$ can be noisy in the following forms (suppose the correct assertion is $A(a)$): (1) A' intersects with A, (2) A' and A share no individuals, and (3) A' is subsumed by A. To simulate these possibilities, we adopt the following method: given a pair of classes (X, Y) and a noise level x, an instance with its label X has a x$\times$100% chance to be corrupted and mislabeld as Y. We use this method

[3] `http://www.cs.waikato.ac.nz/ml/weka/`

because in realistic situations, only certain types of classes are likely to be mislabeled. Using this method, the percentage of the entire training set that is corrupted will be less than x×100% because only some pairs of classes are considered problematic. In the sections below, we construct the following 3 datasets based on LUBM:

RATA: To simulate the noisy type assertion of form (1), we corrupt the individuals of concept `TeachingAssistant` with concept `ResearchAssistant` according to the given noise levels.

UGS: To simulate the noisy type assertion of form (2), we corrupt the individuals of concept `GraduateStudent` with concept `University` according to the given noise levels.

GCC: To simulate the noisy type assertion of form (3), we corrupt the individuals of concept `Course` but not `GraduateCourse` with concept `GraduateCourse` according to the given noise levels.

Data Partition Strategy. Semantic web datasets differ from traditional datasets in the data linkage aspect, which makes data partition different from traditional data partition methods. We sketched the details of partition method used in this paper here, which prevented the training and testing set from containing uncontrolled amount of individuals. The datasets are partitioned by individuals. Given the original dataset, training and testing set individuals, we try to add all concept and property assertions related to the individuals in the corresponding training and testing datasets. Object property assertions can link individuals to others that are not in the individual set. We ignore these property assertions in order to maintain the size of the individual set. In each run, the dataset is randomly divided into a training set and a test set, and we corrupt the training and testing set by adding noise with the above method, and use the testing set to evaluate the system performance.

Experimental Results. Fig. 3 shows the evaluation results on the 4 datasets with noise level 10% - 50% using different feature schemes. From this figure, we find:

- As the noise level grows, we expect to see a decrease in the performance of classification in all evaluations. However, in many cases, we see an increase. This is because after we get more noises, the training data is more balanced to the 4 classes. This is the reason for the increase in the classification performance.
- We may expect the performance better on the disjoint concept pair, a.k.a. UGS. However, this might not be found from the evaluations. Actually, the evaluations on concept pair GCC seem to outperform others. Firstly, we corrupted Course individuals with GraduateCourse types under the condition that the individuals are not GraduateCourse themselves. Because otherwise we are not confident about the corruptions generated are really noises. In this way, the GCC pair is similar to pair UGS. Secondly, in the LUBM dataset, the individuals belonged to Course are less than that of GraduateStudent. Although the noise levels are the same, but the number of noisy type assertions in GCC is smaller than that in UGS.
- Applying the proposed approach with [ALL] gets the best results. Especially on dataset GCC. This indicates that relying solely on concept supports or role supports is not effective enough. Since in several cases, an individual possibly only has

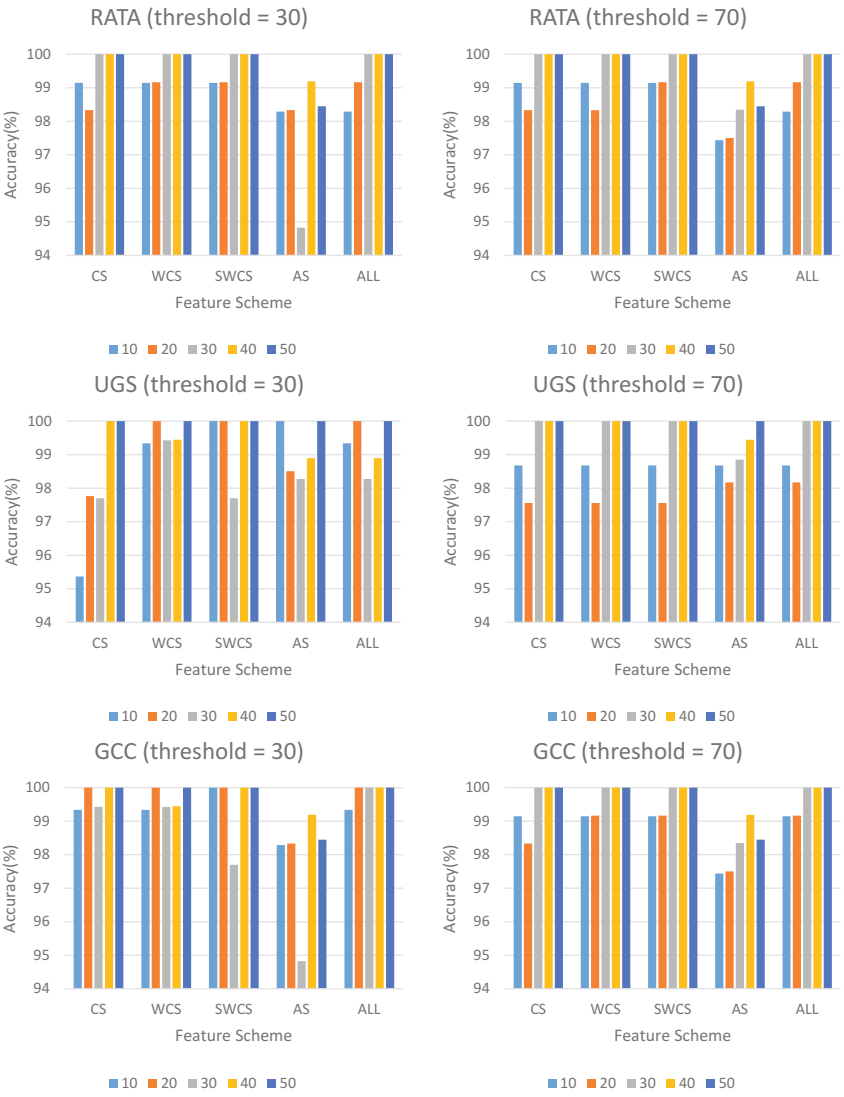

Fig. 3. The average accuracy using various feature schemes with different noise levels by setting thresholds to 30 and 70

concept labels, or only has role links, using one kind of features obviously cannot get enough information for classification.

– On RATA, the concept intersected pair, and GCC, the concept subsumption pair, the performance are also quite good.

In Table 2 we demonstrate the performance of the Adaboost with J48 when noise level is set to be 50%, and the threshold is 70. From this table, we find that in most cases, the

proposed method is able to detect all noisy type assertions. When using [AS] on RATA and GCC, we sometimes missed some conflicts, but the precision is still quite high.

Table 2. Precision, recall, and F1 using different feature schemes (FS) when noise level is 50%, and threshold is 70

	RATA			UGS			GCC		
FS	Precision	Recall	F1 score	Precision	Recall	F1 score	Precision	Recall	F1 score
CS	1.000	1.000	1.000	1.000	1.000	1.000	1.000	0.933	0.965
WCS	1.000	1.000	1.000	1.000	1.000	1.000	1.000	0.933	0.965
SWCS	1.000	1.000	1.000	1.000	1.000	1.000	1.000	0.933	0.965
AS	1.000	0.867	0.929	1.000	1.000	1.000	1.000	0.867	0.929
ALL	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000

5.3 Evaluations on DBpedia

We locally maintained a SPARQL endpoint for DBpedia 3.9, which includes newly created type inference results with estimated precision of 95%. Please refer to `http://github.com/fresheye/NoDe` for the details of packages used in our server.

The essence of our approach is making use of disjoint concepts, however, not stated in the DBpedia ontology yet, to discover the noisy type assertions. This idea can be clear after we look into the frequencies of co-occurrence (the number of instances belonged to a pair of concepts is the co-occurrence frequency for this pair) in DBpedia. We can see one extreme from Fig. 1 which depicts the co-occurrence frequency between 1,000 and 1,000,000, that most pairs of concepts share more than 10,000 instances. The other extreme we can see from Fig. 1 that hundreds of pairs share instances less than 100, however each of the concepts in this pair has more than 10,000 instances itself. We manually construct a benchmark dataset with 4067 data instances, including 170 in (0, 10), 40 in [10, 30), 96 in [30, 50), 51 in [50, 70), 90 in [70, 100), 3673 in [100, 800), and 47 in [800, 1000]

In Fig. 4, we demonstrate the average accuracy of our approach on DBpedia by using difference thresholds by using [ALL] feature scheme. The "all data" lines represent the average accuracy by using all examples in the benchmark dataset. The "same data size" lines show the results of using the same number of examples (170 examples) in the experiment. From Fig. 4 we find:

- The accuracy grows with the threshold, especially when all data are used. This shows that more training examples bring us better model to classify the examples.
- The average accuracy of using J48 with Adaboost is normally better than J48 without Adaboost at approximately 3%.
- When we use same amount of examples in experiments, the accuracy also grows because when the size of the training set grows, the data are more balanced.

Table 3. Average precision (Prec.), recall (Rec.), and F1 score (F1) on DBpedia by setting threshold to 800

	J48			J48(boost)		
Feature	Precision	Recall	F1-score	Precision	Recall	F1-score
CS	0.837	0.831	0.832	0.785	0.782	0.783
WCS	0.768	0.768	0.768	0.844	0.845	0.844
SWCS	0.825	0.824	0.824	0.823	0.824	0.823
AS	0.812	0.761	0.729	0.812	0.761	0.729
ALL	0.936	0.936	0.936	0.956	0.956	0.956

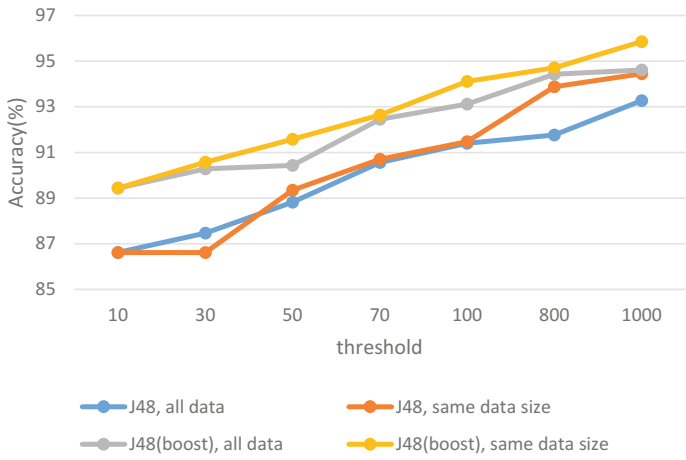

Fig. 4. Average accuracy by J48 and J48(boost) on all data and same size of data with [ALL] feature scheme. Thresholds are set to be 10, 30, 50, 70, 100, 800, and 1000.

We perform the evaluations setting concept disjoint threshold to 800. The evaluation results are shown in Table 3. From Table 3 we find conclusions similar to that in the synthetic evaluations. We expected [WCS] to give high level of statistics in terms of concept support, however the effect of them is limited. Using J48, the best features are [CS] and [SWCS]. Using Adaboost, [WCS] performs the best. Overall, the best features in classifying DBpedia are [SWCS] and [AS]. Combining all features together get the best average F1-score of 95.6%. Table 4 shows some examples of noisy type assertions can be found by our approach.

6 Related Work

Noise detection was mostly studied in the data mining community in the last decades. Zhu and Wu [4] presented a systematic evaluation on the impact of concept and role noises, with a focus on the latter. They concluded (1) Eliminating individuals containing concept noise will likely enhance the classification accuracy; (2) In comparison

Table 4. Examples of noises detected in DBpedia 3.9. The namespaces of the header (concepts) are all http://dbpedia.org/ontology/, and the namespaces of the content (instances) are all http://dbpedia.org/resource/.

ID	Agent	Person	Place
1	Sponge	JSON	PHI
2	Kama	Xbox_Music	England_national_field_hockey_team
3	SQL	Xbox_Video	American_Beaver
4	Free_State_Of_Saxony	Thumbnail	URS
5	Duero	Automobile_Craiova	Al-Qaeda
ID	**PopulatedPlace**	**Settlement**	**Work**
1	Eurovision_Song_Contest_2007	Anglican_Church_of_Southern_Africa	Daugava
2	England_national_field_hockey_team	Byzantine_Catholic_Metropolitan_Church_of_Pittsburgh	North_Coast
3	American_Beaver	U.S._Highway_84_(Alabama)	New_York_State_Library
4	Catholics	River_Blackwater,_Northern_Ireland	Captain_Underpants
5	PHI	British_House_of_Commons	Goodman_School_of_Drama
ID	**Organization**	**MusicalWork**	**Artist**
1	Longfellow_(horse)	Mirage_Press	Citrix
2	Kama	South_African_War	Royal_Pharmaceutical_Society
3	U.S._Geological_Survey	Daugava	Argonne_National_Laboratory
4	Atlantic_ocean	North_Coast	PUC-Rio
5	Juris_Doctor	National_Broadcasting_Network	KOL
ID	**Broadcaster**	**RecordLabel**	**SportsTeam**
1	MHz	Kelin	DOS
2	Tate_Gallery	Velas	Coral_Springs
3	Louisiana_Tech	Central_Europe	Kama
4	TEENick_(block)	Catskills	FSO_Warszawa
5	List_of_Chinese-language_television_channels	Koliba	West_Point

with concept noise, the role noise is usually less harmful. One technique often adopted is voting. Zhu et al. [11] inductively processed partitions of the original dataset; they evaluated the whole dataset using the selected good rules. They adopt majority and non-objection threshold schemes to find noises. Miranda et al. [12] used ML classifiers to make predictions on noisy examples in Bioinfomatics domain. They use majority voting and non-objection voting to filter out erroneous predictions. They concluded that non-objection voting was too conservative and majority voting identified low levels of noise. Kubica and Moore [1] identified corrupted fields, and used the remaining non-corrupted fields for subsequent modeling and analysis. They learned a probability model containing components for clean records, noise values, and the corruption process. Rebbapragada and Brodley [13] assigned a vector of class membership probabilities to each training instance, and proposed to use clustering to calculate a probability distribution over the class labels for each instance. Valizadegan and Tan [14] formulated mislabeled detection as an optimization problem and introduced a kernel-based approach for filtering the mislabeled examples.

Noise detection studies have just begun in the semantic web community. Fürber and Hepp [5] developed generic SPARQL queries to identify (1) missing datatype properties or literal values, (2) illegal values, and (3) functional dependency violations. Yu et al. [6] identified potential erroneous (the degree to which a triple deviates from similar triples can be an important heuristic for identifying "abnormal triples") relational descriptions between objects in triples by learning probabilistic rules from the reference data and checking to what extent these rules agree with the context of triples. Suominen and Mader [15] analysed the quality of SKOS vocabularies, and proposed heuristics to correct the problems in the vocabularies. The focus was mainly on syntax level, made the use of labels consistent for example.

Besides these works dealing with noises detection, type inference works are also related. Paulheim and Bizer [2] studied type inference on dataset like DBpedia. They use role links to infer types of individuals, but they do not detect noises. Gangemi et al. [16] automatically typed DBpedia entities by interpreting natural language definition of an entity. Lehmann et al. [17] validated facts by a deep fact validation algorithm, which provided excerpts of webpages to users who create and maintain knowledge bases. Fanizzi et al. [18] adopted a self-training strategy to iteratively predict instance labels. Fleischhacker and Völker [19] enriched learned or manually engineered ontologies with disjointness axioms. dAmato et al. [20] used inductive methods to handle noises in semantic search.

7 Conclusion and Future Work

In this paper, we study the problem of noisy type assertions, which plays an important role in the performance of semantic web applications. In large datasets, such as DBpedia, the numbers of type assertions are too large to be processed by most ML classifiers, we propose a novel approach that transforms the problem into multiclass classification of a pair of type assertions related to the same individual. We perform evaluations on both synthetic datasets and DBpedia. From the evaluations, we conclude that: (1) Our approach can be applicable to most situations where noises exist; (2) The feature composition that use both concept knowledge and role knowledge outperforms others by conducting evaluations using different feature compositions; (3) Our approach is effective in detecting noisy type assertions in DBpedia with the average precison of 95%.

In the future, we will try to explore the following issues: (1) We will study the impact of noisy types in other assertions in the dataset; (2) We will extend conflict type assertion extraction to the general type of disjointness, to be specific, the concept in the disjoint pair may not be atomic. (3) Currently the detected noises are recorded in a local DB. We will study how to correct them or remove them in the future.

Acknowledgements. This work is partially funded by the National Science Foundation of China under grant 61170165. Additionally, we thank Ying Xu's suggestions.

References

1. Kubica, J., Moore, A.W.: Probabilistic noise identification and data cleaning. In: ICDM, Citeseer, pp. 131–138 (2003)

2. Paulheim, H., Bizer, C.: Type inference on noisy RDF data. In: Alani, H., et al. (eds.) ISWC 2013, Part I. LNCS, vol. 8218, pp. 510–525. Springer, Heidelberg (2013)
3. Zaveri, A., Kontokostas, D., Sherif, M.A., Bühmann, L., Morsey, M., Auer, S., Lehmann, J.: User-driven quality evaluation of DBpedia. In: Proceedings of the 9th International Conference on Semantic Systems, pp. 97–104. ACM (2013)
4. Zhu, X., Wu, X.: Class noise vs. attribute noise: A quantitative study. Artificial Intelligence Review 22, 177–210 (2004)
5. Fürber, C., Hepp, M.: Using semantic web resources for data quality management. In: Cimiano, P., Pinto, H.S. (eds.) EKAW 2010. LNCS, vol. 6317, pp. 211–225. Springer, Heidelberg (2010)
6. Yu, Y., Zhang, X., Heflin, J.: Learning to detect abnormal semantic web data. In: Proceedings of the Sixth International Conference on Knowledge Capture, pp. 177–178. ACM (2011)
7. Quinlan, J.R.: C4. 5: programs for machine learning, vol. 1. Morgan kaufmann (1993)
8. Wu, X., Kumar, V., Quinlan, J.R., Ghosh, J., Yang, Q., Motoda, H., McLachlan, G.J., Ng, A., Liu, B., Philip, S.Y., et al.: Top 10 algorithms in data mining. Knowledge and Information Systems 14, 1–37 (2008)
9. Freund, Y., Schapire, R.E.: Experiments with a new boosting algorithm. In: ICML, vol. 96, pp. 148–156 (1996)
10. Guo, Y., Pan, Z., Heflin, J.: LUBM: A benchmark for owl knowledge base systems. Web Semantics: Science. Services and Agents on the World Wide Web 3, 158–182 (2005)
11. Zhu, X., Wu, X., Chen, Q.: Eliminating class noise in large datasets. In: ICML, vol. 3, pp. 920–927 (2003)
12. Miranda, A.L.B., Garcia, L.P.F., Carvalho, A.C.P.L.F., Lorena, A.C.: Use of classification algorithms in noise detection and elimination. In: Corchado, E., Wu, X., Oja, E., Herrero, Á., Baruque, B. (eds.) HAIS 2009. LNCS, vol. 5572, pp. 417–424. Springer, Heidelberg (2009)
13. Rebbapragada, U., Brodley, C.E.: Class noise mitigation through instance weighting. In: Kok, J.N., Koronacki, J., Lopez de Mantaras, R., Matwin, S., Mladenič, D., Skowron, A. (eds.) ECML 2007. LNCS (LNAI), vol. 4701, pp. 708–715. Springer, Heidelberg (2007)
14. Valizadegan, H., Tan, P.N.: Kernel based detection of mislabeled training examples. In: SDM, SIAM (2007)
15. Suominen, O., Mader, C.: Assessing and improving the quality of skos vocabularies. Journal on Data Semantics, 1–27 (2013)
16. Gangemi, A., Nuzzolese, A.G., Presutti, V., Draicchio, F., Musetti, A., Ciancarini, P.: Automatic typing of dBpedia entities. In: Cudré-Mauroux, P., et al. (eds.) ISWC 2012, Part I. LNCS, vol. 7649, pp. 65–81. Springer, Heidelberg (2012)
17. Lehmann, J., Gerber, D., Morsey, M., Ngonga Ngomo, A.-C.: DeFacto - deep fact validation. In: Cudré-Mauroux, P., et al. (eds.) ISWC 2012, Part I. LNCS, vol. 7649, pp. 312–327. Springer, Heidelberg (2012)
18. Fanizzi, N.: Mining linked open data through semi-supervised learning methods based on self-training. In: 2012 IEEE Sixth International Conference on Semantic Computing (ICSC), pp. 277–284. IEEE (2012)
19. Fleischhacker, D., Völker, J.: Inductive learning of disjointness axioms. In: Meersman, R., et al. (eds.) OTM 2011, Part II. LNCS, vol. 7045, pp. 680–697. Springer, Heidelberg (2011)
20. Damato, C., Fanizzi, N., Fazzinga, B., Gottlob, G., Lukasiewicz, T.: Ontology-based semantic search on the web and its combination with the power of inductive reasoning. Annals of Mathematics and Artificial Intelligence 65, 83–121 (2012)