

Ontology Verification Using Contexts

Aviv Segev and Avigdor Gal
Technion – Israel Institute of Technology
{asegev@tx, avigal@ie}.technion.ac.il

Abstract. Ontologies have become the de-facto modeling tool of choice, used in a variety of applications and prominently in the Semantic Web. Their design and maintenance, nevertheless, have been and still are a daunting task. As a result, ontologies quickly become underspecified. Therefore, if ontologies do not evolve, the semantic infrastructure of the information system can no longer support the changing needs of the organization. In this work we provide a model and a set of algorithms to semi-automatically support relationship evolution in an ontology using contexts. We propose to use (machine-generated) contexts as a mechanism for quantifying relationships among concepts. To do so we compare the contexts that are associated with the ontology constructs. We believe that such a solution will provide significant assistance in supporting ontology design and evolution. The main contribution of this work is twofold. On a conceptual level, we introduce an *ontology verification* model, a quantified model for automatically assessing the validity of relationships in an ontology. On an algorithmic level, we provide a mapping of several ontology operators for defining relationships into context relationships. We motivate our work with examples from the field of eGovernment applications and support our model with an empirical analysis, using real-world traces of RSS and Reuters.

1 INTRODUCTION

Ontologies have become the de-facto modeling tool of choice, used in a variety of applications and prominently in Semantic Web applications. For example, ontologies can be used in discovering Web services [11]. Ontology design, nevertheless, has been and still is a daunting task. It requires collaboration of domain experts with ontology engineers, which may consume many organizational resources in terms of both time and monetary units. Once the ontology is designed, evolving it becomes difficult due to the need for availability of domain experts on the one hand, and costs related with hiring ontology engineers on the other hand. To illustrate this point, consider an eGovernment application, for which an ontology was designed and tailored by an ontology engineer. Once the ontology is installed, changes in the real world require a renewed collaboration of civil servants with ontology engineers to reflect such changes in the ontology. A typical outcome of such difficulties is that ontologies quickly become underspecified. New concepts are introduced in the domain while others become obsolete. Also, shifts of focus in the application domain require the refinement of a concept into a hierarchy of concepts, while in other cases hierarchies should be collapsed. Meeting these challenges requires ontologies to evolve or else the semantic infrastructure of the information system can no longer support the changing needs of the organization.

In [10] we introduced a model for compensating for ontology un-

derspecification using a combination of ontologies with contexts. *Contexts* were defined to be first class objects [5] and will be formally presented later in this work. As an example, a context can be defined to be a set of words, possibly associated with weights that represent the relevance of a word to a document. Ontologies and contexts are both used to model different perspectives of a domain (views). Ontologies represent shared models of a domain and contexts are local views of a domain. We also promote an orthogonal classification in which ontologies are considered a result of a manual effort of modeling a domain, while contexts are system generated models [9]. Ontologies and contexts are joined together, as formally described in [10]. In a nutshell, each concept in an ontology is represented by a name and a context. In this model, contexts serve as an easy-to-use “semantic glue,” in which underspecifications are compensated for with a syntactic, machine generated context, which highlights the intentions of a local designer when using a specific ontology concept, possibly differently from the way it is semantically captured in the ontology using relationships.

In this work we provide a model and a set of algorithms to semi-automatically support relationship evolution in an ontology using contexts. The main motivation for this work stems from the difficulty in supporting ontology evolution. Specifically, this problem was raised within the framework of TerreGov, a European eGovernment project. In this project, ontologies serve as the driving force behind the application and thus affect government processes and Web services, among other things. Therefore, we propose to use (machine-generated) contexts as a mechanism for quantifying relationships among concepts. Specifically, given an ontology operator (e.g., *disjoint*, representing the knowledge that an instance of one concept cannot be an instance of another) and operands (e.g., two concepts or classes), we aim at quantifying the extent to which this relationship is valid. We do so by comparing the contexts that are associated with the operands. We believe that such a solution would significantly assist in the support of ontology design and evolution.

The main contribution of this work is thus twofold. On a conceptual level, we introduce an *ontology verification* model, a quantified model for automatically assessing the validity of relationships in an ontology. On an algorithmic level, we provide a mapping of several ontology operators for defining relationships into context relationships. We motivate our work with examples from the eGovernment domain. However, due to the absence of large scale data sets for this domain, we support our model with an empirical analysis using real-world traces of Reuters data and RSS data.

The rest of the paper is organized as follows. We start with preliminaries, formally defining ontologies and contexts in Section 2. In Section 3 we introduce the ontology verification model, followed by a proposal of a mapping of the ontology verification problem to

contexts in Section 4. We then provide in Section 5 some empirical experiences. We conclude with related work in Section 6 and a short summary in Section 7.

2 ONTOLOGIES AND CONTEXTS

According to Gruber [2], an *ontology* is an explicit specification of a conceptualization of a domain. Several models for ontologies exist, and we follow here that presented in [1]. In the discussion below, we assume reader familiarity with basic concepts in conceptual modeling.

Banerjee [1] defined a *root class* as an object that represents anything from a simple number to a complex entity. An edge between a node and a child node in a class represents an IS-A relationship. Objects that belong to a class are called *instances* of that class. A class describes the *form* (instance variables) of its instances and the *operations* (methods) applicable to its instances.

Moving from ontologies to contexts, the context model we use is based on the definition of context as first class objects formulated by McCarthy [5]. McCarthy defines a relation $ist(\mathcal{C}, P)$, asserting that a proposition P is true in a context \mathcal{C} . We define a *context* $\mathcal{C} = \{\{ \langle c_{ij}, w_{ij} \rangle \}_j\}_i$ as a set of finite sets of descriptors c_{ij} from a domain \mathcal{D} with appropriate weights w_{ij} , representing the importance of c_{ij} . For example, a context \mathcal{C} may be a set of words (hence, \mathcal{D} is a set of all possible character combinations) defining a document Doc , and the weights could represent the relevance of a descriptor to Doc . In classic Information Retrieval, $\langle c_{ij}, w_{ij} \rangle$ may represent the fact that the word c_{ij} is repeated w_{ij} times in Doc .

The context of a class is defined as a set of contexts describing instances that belong to this class. Following [10], we define a class context \mathcal{C}_{CL} of a class CL to be the union of its instance contexts.

Segev and Gal [10] aimed at formalizing the inter-relationships between an ontology, a manually generated domain model, and contexts, partial and automatically generated local views. According to their work, a context can belong to multiple context sets, which in turn can converge to different ontology concepts. Thus, one context can belong to several ontology concepts simultaneously. The appropriate interpretation of a context leads to its relevance to different given concepts.

3 ONTOLOGY VERIFICATION USING CONTEXTS

Ontology verification is the process by which semantic relationships are identified. We term this process verification, since we assume an ontology exists and may need to evolve. Therefore, semantic relationships in an ontology need to be continuously monitored and if necessary, revised. Here we follow the work of [6] on ontology changes and assume a given closed set of operators OT , to be applied on a set of operands OD , taken from the set of all ontology elements. As an example, a change operator may be the *disjoint* operator, resulting in the creation of a semantic relationship called “disjoint” between two classes, given to it as operands.

Figure 1 provides a pictorial representation of the process. Formally, ontology verification is a function $OV : OT \times OD^* \rightarrow [0, 1]$. Ontology verification is given as input a hypothesis regarding the possible operator to be applied to one or more operands and returns a level of certainty μ regarding the truth in this hypothesis. A certainty of 1 indicates full certainty in the hypothesis, while a certainty of 0 means that the hypothesis is definitely incorrect. In Figure 1, the

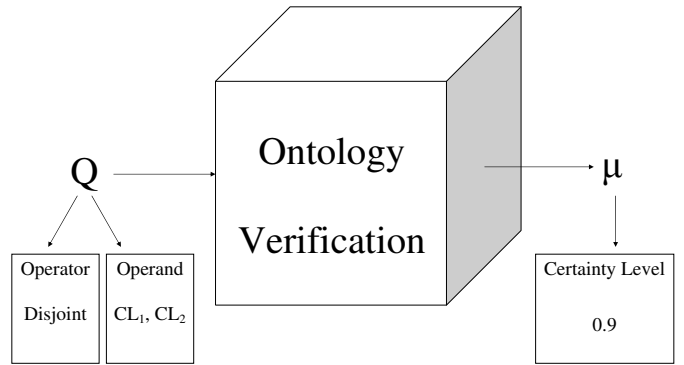


Figure 1. Ontology Verification Model

ontology verification function determines that the disjointness of classes CL_1 and CL_2 has a certainty level of 0.9.

4 MAPPING ONTOLOGY VERIFICATION TO CONTEXT

Having introduced ontology verification, we now focus on the details of change operators. Noy and Klein [6] describe a set of 22 ontology change operators and their impact on ontology elements (both classes and instances). For nine of these operators we show how verification can be accomplished using contexts. Verification of the remaining operators remains open for future research. The nine operators can be grouped into three groups as follows:

Class creation and deletion Class creation involves grouping a set of instances based on some common properties. When such commonality no longer exists, a class is deleted.

Hierarchy linking and restructuring In hierarchy linking one class becomes a superclass of another, making the latter class a subclass of the former. Hierarchy restructuring involves reversing the roles of a superclass and a subclass.

Disjoining, merging, and splitting classes A disjoint operator involves adding a disjoint link between two classes. Merging two classes involves joining all instances of both classes into a single class. Splitting entails the partition of instances of one class into several classes.

For each operator group, we now provide an instantiation of the verification function, defining a level of certainty μ using context comparison.

4.1 Class Creation and Deletion

For these two operators, decisions are taken based on the individual instance contexts. Given a set of instances and a hypothesis of either class creation or class deletion we generate a context for each instance (e.g., using [9]) and in turn a class context from the instance contexts. We expect instance contexts to be similar, thus resulting in a class context that is similar to all instance contexts. A higher overlap between instance contexts results in a smaller context size of the class (recall that a class context is the union of its instance contexts).

Let CL be a class and $I = \{I_1, I_2, \dots, I_n\}$ be a set of instances. We denote by \mathcal{C}_{CL} the context of a class CL and by $|\mathcal{C}_{CL}|$ the context cardinality. For instance I_i , we denote by \mathcal{C}_i its context and by $|\mathcal{C}_i|$ the context cardinality. Given an operator *Create* and a set of instances I , we hypothesize that CL is a class of I with a context $\mathcal{C}_{CL} = \cup_{1 \leq i \leq n} \mathcal{C}_i$ and compute the certainty value of this operator to be

$$\mu_{Create} = \begin{cases} 1 - \frac{|\mathcal{C}_{CL}| - \max_{1 \leq i \leq n} (|\mathcal{C}_i|)}{(\sum_{1 \leq i \leq n} |\mathcal{C}_i|) - \max_{1 \leq i \leq n} (|\mathcal{C}_i|)} & n > 1 \\ 1 & n = 1 \end{cases}$$

In an extreme case, where for any two instances $\{I_i, I_j\} \subseteq I$, $\mathcal{C}_i = \mathcal{C}_j$, $\mathcal{C}_{CL} = \mathcal{C}_i$, $|\mathcal{C}_{CL}| = |\mathcal{C}_i|$ and $\mu_{Create} = 1$. At the other extreme, where for any two instances $\{I_i, I_j\} \subseteq I$, $\mathcal{C}_i \cap \mathcal{C}_j = \emptyset$, $|\mathcal{C}_{CL}| = \sum_{1 \leq i \leq n} |\mathcal{C}_i|$, and $\mu_{Create} = 0$. The special case of $n = 1$ (a single instance) always results in the highest certainty value.

Class deletion is a reverse operator of class creation. Therefore, given a class CL and its instances $I = \{I_1, I_2, \dots, I_n\}$, we compute the certainty value of the operator *Delete* to be

$$\mu_{Delete} = 1 - \mu_{Create} = \begin{cases} \frac{|\mathcal{C}_{CL}| - \max_{1 \leq i \leq n} (|\mathcal{C}_i|)}{(\sum_{1 \leq i \leq n} |\mathcal{C}_i|) - \max_{1 \leq i \leq n} (|\mathcal{C}_i|)} & n > 1 \\ 0 & n = 1 \end{cases}$$

4.2 Hierarchy Linking and Restructuring

Given two classes, CL_i and CL_j , if CL_i is a subclass of CL_j , then its context should contain the context of CL_j . This is because an instance of CL_i is also an instance of CL_j and therefore has a broader context than an instance of the superclass. Therefore, we compute the certainty of a hypothesis that CL_i is a subclass of CL_j to be

$$\mu_{Sub-Sup} = \frac{|\mathcal{C}_{CL_i} \cap \mathcal{C}_{CL_j}|}{|\mathcal{C}_{CL_j}|}$$

In one extreme case, where $\mathcal{C}_{CL_j} \subset \mathcal{C}_{CL_i}$, then $|\mathcal{C}_{CL_i} \cap \mathcal{C}_{CL_j}| = |\mathcal{C}_{CL_j}|$ and $\mu_{Sub-Sup} = 1$. At the other extreme, where $\mathcal{C}_{CL_i} \cap \mathcal{C}_{CL_j} = \emptyset$, then $|\mathcal{C}_{CL_i} \cap \mathcal{C}_{CL_j}| = 0$ and $\mu_{Sub-Sup} = 0$ as well. Removing a hierarchy link is a reverse operator of the linking operator.

When determining a move in a hierarchy, we analyze the number of overlapping contexts between a subclass and a superclass, similarly to the *Sub-Sup* operator. Here, though, we reverse the roles of the subclass and superclass. Thus, given two classes CL_i and CL_j , CL_i is a subclass of CL_j , moving CL_i up the hierarchy (or symmetrically moving CL_j down the hierarchy) has a certainty level of

$$\mu_{up} = \frac{|\mathcal{C}_{CL_i} \cap \mathcal{C}_{CL_j}|}{|\mathcal{C}_{CL_i}|}$$

It is worth noting that μ_{up} for moving CL_i up the hierarchy is the same as determining that CL_j is a subclass of CL_i . When moving up the hierarchy, one more decision may need to be taken. Assume that CL_j is a subclass of some third class CL_k . When we move CL_i up, we need to consider whether CL_i should become a subclass of CL_k . If this is indeed the case, then due to the transitivity of the IS-A relationship, CL_j should be disconnected from CL_k .

4.3 Disjoining, Merging, and Splitting Classes

Two classes, CL_i and CL_j are disjoint if their contexts do not overlap. Therefore, we can compute the certainty of a *Disjoint* operator

among two classes to be

$$\mu_{Disjoint} = 1 - \frac{|\mathcal{C}_{CL_i} \cap \mathcal{C}_{CL_j}|}{|\mathcal{C}_{CL_i} \cup \mathcal{C}_{CL_j}|}$$

If $\mathcal{C}_{CL_i} = \mathcal{C}_{CL_j}$, then $\mathcal{C}_{CL_i} \cap \mathcal{C}_{CL_j} = \mathcal{C}_{CL_i} \cup \mathcal{C}_{CL_j}$ and thus $\mu_{Disjoint} = 0$. If $\mathcal{C}_{CL_i} \cap \mathcal{C}_{CL_j} = \emptyset$, then $|\mathcal{C}_{CL_i} \cap \mathcal{C}_{CL_j}| = 0$ and thus $\mu_{Disjoint} = 1$.

It is worth noting that while *Disjoint* is a symmetric operator, the hierarchy linking and traversal are directional operators. This is reflected in the different methods for computing $\mu_{Sub-Sup}$, μ_{up} , and $\mu_{Disjoint}$, where the denominator for the former is $|\mathcal{C}_{CL_j}|$ (the super-class context cardinality) and for the latter is $|\mathcal{C}_{CL_i} \cup \mathcal{C}_{CL_j}|$.

When two classes are merged, their contexts are analyzed to identify overlappings. Thus, the merge operator can be considered the reverse operator of disjoint, yielding a certainty level computed as follows:

$$\mu_{Merge} = 1 - \mu_{Disjoint} = \frac{|\mathcal{C}_{CL_i} \cap \mathcal{C}_{CL_j}|}{|\mathcal{C}_{CL_i} \cup \mathcal{C}_{CL_j}|}$$

There can be multiple variations of splitting two classes. In one variation, a user provides the verification function two groups of instances $I' = \{I_1, I_2, \dots, I_m\}$ and $I'' = \{I_{m+1}, I_{m+2}, \dots, I_n\}$, constituting a partition of the instances of a class CL . For such a variation, we need to generate a class context for I' and I'' and then determine if they are disjoint or not. Therefore, computing the certainty level of this variation of the *Split* operator is identical to that of the *Disjoint* operator.

In a second variation, no instance partitioning is given. Therefore, we need first to determine the best partitioning of the class, using a statistical method such as the K-Means algorithm (with $K = 2$), and then compute μ_{Split} as before. It is worth noting that while the same function is used to determine class split and disjointedness, these operations are not equivalent. A user is likely to determine to split classes with a lower level of certainty than that needed for establishing a disjoint relationship.

Table 1 provides a summary of the operators and the verification function instantiation for each one.

5 EXPERIENCES WITH CONTEXT BASED ONTOLOGY VERIFICATION

Our experiences are based on data from the Reuters corpus and from the RSS news data trace. In these data traces, data were originally partitioned to topics with no ontological relationships. The Reuters data set was taken from a publicly available trace (<http://about.reuters.com/researchandstandards/corpus/>). We chose 10 news topic categories (referred to hereafter as classes), for a total of 3,125 data, where a datum is a Reuters news article. The RSS trace was collected during August 2005 from the CNN Web site. Here, we also chose 10 news topic categories including 1,130 data, where a datum is a RSS news header or a news descriptor. The main difference between the Reuters trace and the RSS trace is the datum size.

We generated a context for each datum and each class using an automatic context extraction algorithm [9]. The number of context descriptors generated from each datum was set to 10. The data size used for RSS varied from 73 to 1911 per class, while for the Reuters data size per class varied from 126 to 510.

In our experiment we calculated for each class the number of contexts that overlapped with the other nine classes. This asymmetric

Operation (based on [8])	Verification function
Create class CL	$\mu_{Create} = \frac{ C_{CL} - \max_{1 \leq i \leq n}(C_i)}{1 - \left(\sum_{1 \leq i \leq n} C_i \right) - \max_{1 \leq i \leq n}(C_i)}$
Delete a class CL	$\mu_{Delete} = 1 - \mu_{Create}$
Add a subclass-superclass link between a subclass CL_i and a superclass CL_j	$\mu_{Sub-Sup} = \frac{ C_{CL_i} \cap C_{CL_j} }{ C_{CL_j} }$
Remove a link	$\mu_{ReSub-Sup} = 1 - \frac{ C_{CL_i} \cap C_{CL_j} }{ C_{CL_j} }$
Move a class up in the class hierarchy	$\mu_{up} = \frac{ C_{CL_i} \cap C_{CL_j} }{ C_{CL_i} }$
Move a class down in the class hierarchy	$\mu_{down} = 1 - \frac{ C_{CL_i} \cap C_{CL_j} }{ C_{CL_i} }$
Declare classes CL_1 and CL_2 as disjoint	$\mu_{Disjoint} = 1 - \frac{ C_{CL_1} \cap C_{CL_2} }{ C_{CL_1} \cup C_{CL_2} }$
Merge classes CL_1 and CL_2	$\mu_{Merge} = 1 - \mu_{Disjoint}$
Split a class	$\mu_{Split} = \mu_{Disjoint}$

Table 1. Ontology Verification Model Operations

comparison gave us for any two classes CL_i and CL_j the metric of $|C_{CL_i} \cap C_{CL_j}|$ and $|C_{CL_i} \cup C_{CL_j}|$. Given two classes and their associated contexts, we analyzed the certainty level of all the operators for all the class pairs, using the functions given in Section 4.

Our first experience involves an analysis of hierarchy linking. Figure 2 presents the RSS class relations hierarchy created for $\mu_{Sub-Sup} \geq 0.8$ and $\mu_{Sub-Sup} \geq 0.5$. Similarly, Figure 3 presents the Reuters class relations hierarchy created for $\mu_{Sub-Sup} \geq 0.7$, $\mu_{Sub-Sup} \geq 0.6$, and $\mu_{Sub-Sup} \geq 0.5$. As the value of $\mu_{Sub-Sup}$ decreases, the hierarchy and the relations between the classes become more elaborated. For example, in the RSS data for $\mu_{Sub-Sup} \geq 0.8$ the superclass Money Latest has four subclasses. If we examine the same classes for a lower verification level of $\mu_{Sub-Sup} \geq 0.5$ we receive a three level hierarchy. Similar results were observed for the Reuters data. Here we see that Ship is a superclass for all the shipment topics displayed, which include Money-Fx, Earn, and Crude. Money-Fx includes another set of shipment and economic topics that form the subclasses Sugar, Corn, and Money Supply. Earn, conversely, includes only the subclass of Money Supply.

Class Sets	Merge	Link Subclass	Disjoint
Money Latest	19.2%	86.7%	80.8%
Money News International	19.2%	19.8%	80.8%
Money News Economy	12.1%	19.5%	87.9%
Money Markets	12.1%	24.3%	87.9%

Table 2. Operator μ Verification RSS

Table 2 compares the certainty level of three operators, namely Merge, Superclass-Subclasses, and Disjoint, for two class pairs in the RSS data set. When evaluating the classes Money Latest and Money News International, there is a high $\mu_{Sub-Sup}$ level and a

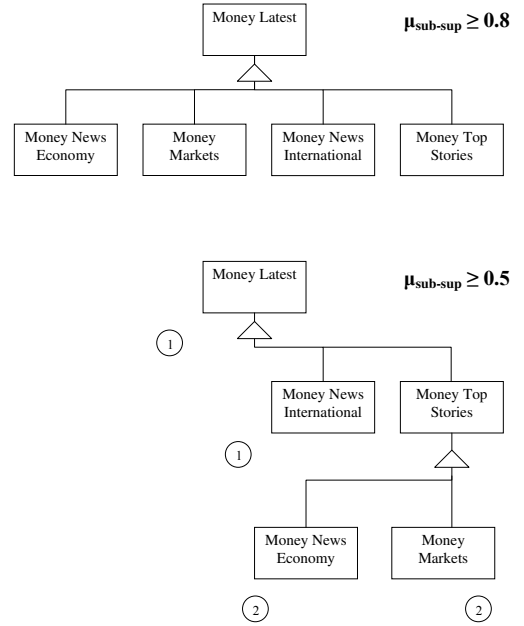


Figure 2. RSS Relations

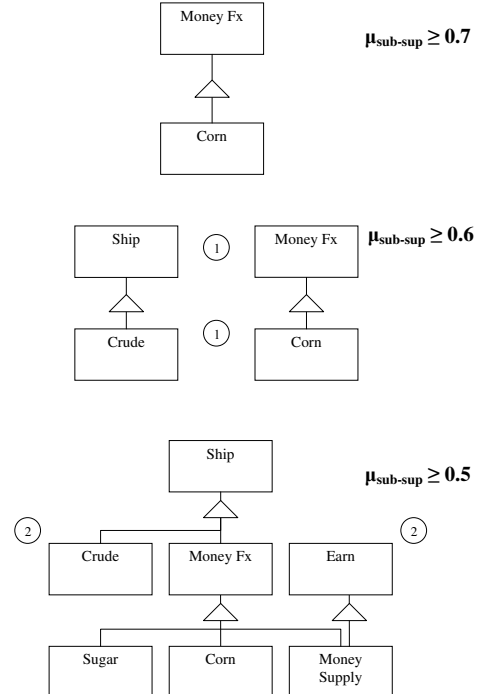


Figure 3. Reuters Relations

Class Sets	Merge	Link Subclass	Disjoint
Money Fx Corn	15.4% 15.4%	75.5% 37.9%	84.6% 84.6%
Crude Earn	12.8% 12.8%	6.2% 42.3%	87.2% 87.2%
Ship Crude	22.0% 22.0%	66.7% 23.0%	78.0% 78.0%

Table 3. Operator μ Verification Reuters

low μ_{Merge} level. Although there is also a high $\mu_{Disjoint}$ level, it is not as high as $\mu_{Sub-Sup}$ and thus it is likely that these two classes should be in a superclass-subclass relationship. When examining two sibling classes in the hierarchy, Money News Economy and Money Markets, the $\mu_{Disjoint}$ level is significantly higher than the other certainty levels, indicating a possible *Disjoint* relationship between the two classes.

Table 3 describes results received for the Reuters data. It is worth noting that here, when comparing Money FX and Corn, although there is a high $\mu_{Sub-Sup}$ value, there is even a higher $\mu_{Disjoint}$ level, which could be indicating that these two classes should not be in a hierarchical relationship if we observe the highest μ value for all operations. However, a user will usually probably prefer to assign a different μ value for each operator. Therefore, for a $\mu_{Disjoint} \geq 0.9$ and $\mu_{Sub-Sup} \geq 0.5$ we receive the above relations verifications. Similarly for classes Ship and Crude the same higher disjoint value and the above relations can be observed.

By analyzing the content of the data in the different classes, we notice that the data set of Corn actually discusses sale prices and purchase quantities and thus matches the relationship with Money Fx. Similarly, Crude is discussed here as raw material, which has no direct relation with the class of Earn but does have a relation with the shipment of the Crude material.

6 RELATED WORK

A formal mathematical framework that delineates the relationships between contexts and ontologies is presented in [10]. To deal with the uncertainty associated with automatic context extraction from existing instances, such as documents, a ranking model was provided, which ranks ontology concepts according to their suitability with a given context.

A semi-automated method for ontology evolution using documents clustering was proposed in [12]. From the results of the clustering ontology enrichments and updates are extracted. In contrast to the above work, which is based on a single word ontology concept description, we use a set of contexts describing each ontology class.

Noy and Klein [6] defined a set of ontology-change operations and their effects on instance data used during the ontology evolution process. They describe ontologies schemas and database schemas from the point of view of evolution and highlight the main differences between them. We presented a model that shows how these ontology change operations can be verified based on context.

Tools for merging and aligning ontologies, such as SMART [7], SENSUS [3], PROMPT [8], and Cyc [4], have been developed in the past. These tools generally present a set of basic operations that are performed during the merge and alignment of ontologies and that determine the effects that the invocation of each of these operations has on the process.

A work on multi-contextual ontology evolution [13] defines a set of properties that by semantic autonomy must hold at the same time.

7 CONCLUSION

This work presents a model and a set of algorithms to semi-automatically support ontology relationship evolution using contexts. Given an ontology operator and operands, the model provides the quantification of the extent to which the relationship is valid. The model is supported by empirical analysis, using initial experiences with real-world RSS and Reuters traces. The experiences with these traces show how relationships between the classes can be created and modified. Preliminary empirical results show that our model can provide good estimations of the need for ontology changes.

To recap, the main contribution of this work is both conceptual and algorithmic. We present an ontology verification model, a quantified model for automatically assessing the validity of relationships in an ontology, and we also provide a mapping of several ontology operators for determining relationships among classes.

The results of this work will be embedded as part of the TerreGov solution. Future research will examine the model performance on eGovernment data and other large data sets. In addition, we plan on extending the usage of the model to additional operators, especially those involving properties.

REFERENCES

- [1] J. Banerjee, H.-T. Chou, J. Garza, W. Kim, D. Woelk, and N. Ballou, 'Data model issues for object-oriented applications', *ACM Transactions on Office Information Systems*, **5**(1), 3–26, (1987).
- [2] T. R. Gruber, 'A translation approach to portable ontologies', *Knowledge Acquisition*, **5**(2), (1993).
- [3] K. Knight and S. K. Luk, 'Building a large-scale knowledge base for machine translation', in *Proceedings of the Twelfth National Conference on Artificial Intelligence (AAAI-94)*, (1994).
- [4] D. B. Lenat, 'Cyc: A large-scale investment in knowledge infrastructure', *Communications of ACM*, **38**(11), 33–38, (1995).
- [5] J. McCarthy, 'Notes on formalizing context', in *Proceedings of the Thirteenth International Joint Conference on Artificial Intelligence*, (1993).
- [6] N. F. Noy and M. Klein, 'Ontology evolution: Not the same as schema evolution', *Knowledge and Information Systems*, **6**(4), 428–440, (2004).
- [7] N. F. Noy and M. A. Musen, 'An algorithm for merging and aligning ontologies: Automation and tool support', in *Proceedings Sixteenth National Conference on Artificial Intelligence (AAAI-99) Workshop on Ontology Management*, (1999).
- [8] N. F. Noy and M. A. Musen, 'The prompt suite: Interactive tools for ontology merging and mapping', *International Journal of Human-Computer Studies*, **59**(6), 983–1024, (2003).
- [9] A. Segev, 'Identifying the multiple contexts of a situation', in *Proceedings of IJCAI-Workshop Modeling and Retrieval of Context (MRC2005)*, (2005).
- [10] A. Segev and A. Gal, 'Putting things in context: A topological approach to mapping contexts and ontologies', in *Proceedings of AAAI-Workshop Workshop on Contexts and Ontologies: Theory, Practice and Applications*, (2005).
- [11] E. Toch, A. Gal, and D. Dori, 'Automatically grounding semantically-enriched conceptual models to concrete web services', in *ER*, eds., L.M.L. Delcambre, C. Kop, H.C. Mayr, J. Mylopoulos, and O. Pastor, volume 3716 of *Lecture Notes in Computer Science*, pp. 304–319. Springer, (2005).
- [12] G. Tsatsaronis, R. Pitkanen, and M. Vazirgiannis, 'Clustering for ontology evolution', in *Proceedings of the 29th Annual Conference of the German Classification Society (GfKI 2005)*, (2005).
- [13] M. Zurawski, 'Reasoning about multi-contextual ontology evolution', in *Proceedings of the First International Workshop on Context and Ontologies: Theories, Practice and Applications, The Twentieth National Conference on Artificial Intelligence (AAAI-05)*, (2005).