Ontology Maintenance with an Algebraic Methodology: a Case Study *

Jan Jannink, Gio Wiederhold
Stanford University
Stanford CA, 94305, U.S.A.
{jan, gio}@db.stanford.edu

Abstract

There is a growing need to combine information from multiple existing knowledge bases for novel uses, and to maintain such information when the underlying knowledge bases change. The autonomy of diverse knowledge sources is an obstacle to integrating all pertinent knowledge within a single knowledge base. The cost of maintaining integrated knowledge within a single knowledge base grows both with the volatility and the number of the sources from which the information originates. Establishing and maintaining application specific portions of knowledge sources are therefore major challenges to ontology management.

Rather than materializing all of the information from the sources into a single knowledge base, we are developing an algebra that enables the construction of virtual knowledge bases geared towards a specific application. This algebra consists of composable operators that transform contexts into contexts. Operators express the relevant parts of a source and the conditions for combining sources using rules. These rules define what is necessary to achieve a valid transformation from the source information to the application context. Rules which expose the relevant parts of a source determine what we call a congruity measure between the source and its target application. The rules which articulate knowledge from diverse sources establish a similarity measure between them.

We focus on one example to show how the use of our algebra is an important framework for establishing new applications using existing knowledge, and for maintaining up to date knowledge in the face of changes to the underlying sources. We have used an on-line dictionary that is autonomously maintained to develop a novel thesaurus application. With over 120,000 entries, two million words in the definitions, and semi-annual updates, this dictionary has provided us with a test bed to examine the issues of creation and maintenance of knowledge contexts using an algebraic methodology.

*This work was supported by a grant from the Air Force Office of Scientific Research (AFOSR).

Introduction

This paper presents a concrete case study of knowledge extraction from an autonomous source, for use in a real world application. In the course of developing the application, the underlying data has been updated three times, requiring a further execution of the knowledge extraction operation to bring the application up to date. We show how a principled approach based on an algebra has made it possible to create and maintain access to this information with a low overhead cost. We present the example in order to motivate what follows.

There exist proprietary dictionaries (Mirriam-Webster 1999) and encyclopedias (Encyclopedia Britannica 1999) that are accessible through the World Wide Web, but typically only one term at a time. Most freely available dictionaries are at best partial, or limited to a specific domain (Geraci 1996). Among the most extensive freely available corpora we find WordNet (Miller & *al.* 1990), but it too suffers from being hand-crafted, and can not claim to be a complete language reference. However, there does exist an on-line version of the 1913 Webster's dictionary that is available through the Gutenberg Project (PROMO.NET 1999). The original dictionary is a corpus of over 50 MB containing some 125,000 terms, and over 2,000,000 words in the definitions alone.

The source data of the dictionary was originally scanned and converted to text via character recognition software, and therefore contains thousands of errors and inconsistencies. The abundance of errors in the dictionary data makes it an ideal test bed for our research. Dealing with incorrectness in sources helps our approach to achieve the robustness needed for real-world settings. Our target application for the dictionary data is the construction of a graph of the definitions from which we can determine related terms, and automatically generate thesaurus style entries. As we run flow algorithms on the graph structure, accuracy in the data is important for meaningful results. However, misspellings and incompleteness in the terms and definitions, as well as errors in the labelling of the data resulted in over five percent of the data being incorrectly interpreted using a naive wrapper. We iteratively refined the naive wrapper using an operator from our algebra to reduce the exception rate below one percent.

The Webster's dictionary is a representative real world data set, in that it contains inconsistencies, and is autonomously updated. Our use for the information contained in the dictionary is typical of a demanding application, as it has strict tolerances for how much erroneous information it allows and it provides a legitimate service. We discuss in detail, within an algebraic framework, how we establish and maintain a context that takes the source data, and makes it available to the application, in such a manner that it meets the application's requirements. We begin by examining related work, and by covering, in the next section, the background material for our work. We continue by presenting the initial derivation of the context for our application, and show in the following section how the refinement process is repeated, albeit with less overhead, when the source undergoes change. In the final section we describe our application in more detail, its relevance to ontology management, and discuss our future directions of research.

Related Work

The starting point for this work comes from a proposal for an algebra for ontology composition (Wiederhold 1994). This proposal foresaw problems in maintaining knowledge combined from autonomous knowledge sources. In this section we examine some of the other work that is relevant to this paper.

A rule-based approach to semantic integration is presented in (Bergamaschi, Castano, & Vincini 1999). It uses a description logic to generate a shared ontology for the source information, and rules to map terms to the common format. However, the notion of maintenance and considerations of scalability are lacking from the discussion. Also, there is no view of the integration process, as an application of an algebra over the source domains.

In considering semantic reconciliation between data source and data receiver (Siegel & Madnick 1991) covers some issues relating to congruity. It does not discuss maintenance issues nor the actual algorithms used to achieve semantic reconciliation.

In (Uschold *et al.* 1998) the problem of reuse of ontologies is considered, and a handcrafted adaptation of an ontology is presented. This work applies a one time transformation of the ontology to fit the requirements of the target application. The notion of maintenance is not considered in this work, as changes to the specific source ontology are infrequent.

The work on specification morphisms (Smith 1993) develops a system to iteratively refine a program specification into a working application. Our work follows similar principles for deriving contexts from sources, and refining them for use in a target application. It extends the notion of specification to allow for inaccuracy, inconsistency and incompleteness found in real world information sources.

The WordNet system (Miller & *al.* 1990) is a corpus of nouns, verbs and adjectives that lists lexical relationships between entries in the system. It is specific about the relationships between entries, but is therefore limited to a small set of possible relationships. Also, it separates the different parts of speech into separate categories, and is complete only in limited domains.

WHIRL (Cohen 1998) uses textual similarity to find co-referent terms in distinct sources with high accuracy. The textual similarity measure it defines is one example of the multitude of initial similarity measures that approximate the true similarity relationship between sources.

The PageRank algorithm for ranking the importance of web pages is due to (Page & Brin 1998). PageRank is a flow algorithm over the graph structure of the World Wide Web that models the links followed during a random browse through the Web. It is the starting point for the algorithm we use to determine the strength of the relationship between dictionary terms.

Latent semantic indexing (Deerwester *et al.* 1990) and hypertext *authority* (Kleinberg 1998) exploit properties of eigenvectors to answer queries over a corpus of text documents or web pages. The eigenvectors are computed from the adjacency matrix of a graph representing the structure of the corpus. These methods reject stop words such as 'The' and do not seek to measure the relative importance of relationships between terms. However, the underlying mathematics of these systems are close to our dictionary application.

Background

In this section we provide an overview of the algebra and its operators. We present the background for the algebra, including our object model, and the primitive operations on objects that form a rule language underpinning the algebra. We also present a formal definition of context that we use to encapsulate ontologies. Contexts are the unit of semantic consistency in our framework, and are the operands of the algebra.

Algebraic Operators

The algebra consists of a composable set of operators which transform contexts into contexts. These contexts, defined in more detail in the following subsection, encapsulate ontologies with a guarantee of semantic consistency. The operators are listed below with a brief description of the operation they perform. Each of these is defined and described in detail in the technical report (Jannink *et al.* 1999).

- unary operators

  `Summarize` term classification
  `Glossarize` listing of terms
  `Filter` object instance reduction
  `Extract` schema simplification

- binary operators

  `Match` term corroboration and reformulation
  `Difference` schema distance measure
  `Intersect` schema discovery
  `Blend` schema extension

Unary operators reformulate source information with respect to the requirements of the target application. `Summarize` is the canonical unary operator. It is used to establish and refine a context within which the source knowledge meets the requirements of the application. We will define the `S` operator in detail in the next section, but first we define context and the semantic guarantees that a context provides.

Semantic Context

We motivate the need for context by observing that there is no global notion of consistency of information. Models of knowledge that are appropriate for one application may be useless for another. Identical terms in separate sources will invariably have differing semantics, while distinct terms, even within the same source, may have equivalent semantics. What we desire, is the ability to specify that the semantics of the objects relevant to an application are locally consistent, and free of mismatch.

We define, following (Guha 1991), contexts to be objects that encapsulate other objects. Contexts assert the validity of statements about the objects they encapsulate. In other words, given an appropriate set of statements about its objects, a context provides guarantees about their consistency. Since we use contexts to model knowledge obtained from diverse sources for application specific uses, we are concerned with two specific relationships: *congruity* and *similarity*. The former expresses the relevance of source information to the target application, the latter identifies equivalent and mergeable objects between different sources. While the two relationships resemble each other, distinguishing the two is important for maintenance and scalability. Because we assume sources are autonomous, they may change at any time. In particular, as the number of sources grows the likelihood of change at any time increases dramatically. By distinguishing congruity and similarity we are able to separate changes of a source that affect their relevancy to our application from those changes that affect their similarity to other sources with which we combine them.

In our system, contexts are an object whose value consists of a ruleset and a sequence of objects represented by the ruleset. As implied by the previous statement object values are a sequence of values, both of primitive and object types. The ruleset itself is an object, whose interpretation defines other objects. The ruleset transforms source knowledge into an object set that meets the consistency requirements of the target application. The consistency guarantee, as embodied by a congruity expression is written in the rule language given below in the next section.

Rule Language

While the algebra transforms contexts, it is also convenient to be able to express these transformations in terms of operations on the objects that compose the contexts. The set of primitives below is complete with respect to transformations of the object model defined in the next subsection. The completeness proof appears in a separate technical report (Jannink *et al.* 1999). In addition, conversion operators allow transformations from string and numerical values to objects, which enables their use in wrapping sources which are not already captured by a context. These conversion operators allow uninterpreted components of an object to become attributes of the object. The operators listed below form expressions that operate on an idealized data stream from the information source to the application. Expressions concatenated together are executed in sequential order, and require multiple passes of the data stream, while nested expressions represent a single pass of the data stream.

- constructors
  - **create object** constructor taking a sequence of values as parameters
  - **create set** constructor taking a set of values as parameters
- connectors
  - **match object** set an object to proxy source objects that match a predicate
  - **match set** set a proxy per equivalence class defined by a predicate applied to source object sets
- editors
  - **insert value** insert a value into object at a specified position
  - **edit value** edit a value within object at a specified position
  - **move value** move a value within object to a specified position
  - **delete value** delete a value from object at a specified position
- converters
  - **object/value** value object transform: string, number, set $\longleftrightarrow$ object
  - **object indirection** referentialize object label: object $\longleftrightarrow$ object - reference - object
  - **reference indirection** reify reference: reference $\longleftrightarrow$ reference - object - reference

Note that a context is itself an object and that the rule language specifies how the context is populated with values from sources. Constructors create new objects, not represented directly in sources. Connectors generate proxy objects that stand in for one or more objects from sources, which may then be modified using editors and converters. In the following section we list the requirements for the object model to support the entire framework presented above.

Object Model

Rather than invent yet another model to represent the objects of our ontologies, we simply allow any model that satisfies the following semantics. These semantics subsume a number of existing models, and are powerful enough to simulate others. Any adequate object model must provide for the following abstractions:

reference object identity

value object is a possibly nested sequence of values

attribute set labeled set of atoms

atom object references, strings, numbers, sets as primitive values

Note that only objects have an identity to which others can refer. All other components of the model are atomic values, which can not be shared. Objects have a value, which is defined as the sequence of values that compose it. This model corresponds to XML supplemented with object identity, where a bracketed `<obj>...</obj>` XML object corresponds to an object in our model. The object model allows us to represent, without modification, HTML and XML documents, data in Stanford's OEM format, plain text and database relations. Furthermore, it is rich enough to model more complex relationships such as inheritance, and typing. This expressiveness allows it to simulate UML (Fowler 1998) and frame (Karp 1992) models.

In this section we have presented the foundations necessary in order to present the use of the algebra in the creation and maintenance of an application specific ontology. In the next section, we describe the dictionary ontology, as well as how to create and refine it. We focus on one algebra operator in particular, the `Summarize` or S operator. The S operator is the primary tool for refining and maintaining a context between a knowledge source and an application.

Context Creation

In the previous section we have presented the foundations of the algebra we use for ontology management. Here we present a definition for the `Summarize` operator and show its use in creating and refining a dictionary ontology.

`Summarize` Operator

The S operator is a unary operator that transforms source data based on a predicate which corresponds to, or augments a congruity expression. The predicate is a connector, a matching rule as defined above in the rule language. S creates an object that encapsulates the information of the source, and populates the object with results of an aggregation operation over the source information. The application that motivates the existence of the S operator is data classification. The aggregation over the source data effectively groups the source into equivalence classes. Given contexts c_1, c_2, a matching rule e, The syntax of the operator is as follows:

$$c_2 = \mathsf{S}_e(c_1)$$

Formally, the matching predicate e partitions the objects of the initial context c_1 into n equivalence classes. The constructed result context c_2, is an object consisting of $n+1$ values: the first is e, and the following n values are sets $s_1 \ldots s_n$ of references to each of the objects of c_1. One of the equivalence classes of the result context is an exception class, for objects that can not match e. Since it is difficult to grasp the capabilities of the S operator from a description alone, we present an extended example from our research. Following the example we describe the algorithm for establishing a congruity measure between a source data set and its new application.

Webster's Dictionary

We have been using data from the Webster's dictionary to research the automatic extraction of thesaurus style entries, such as for the term **Egoism** in Figure 1. The relationships between terms are expressed using the implicit structure contained in the dictionary, rather than explicitly marked ones such as synonyms. The purpose of the application is to serve as a tool in reducing the occurrence of lexical mismatch when merging diverse ontologies.

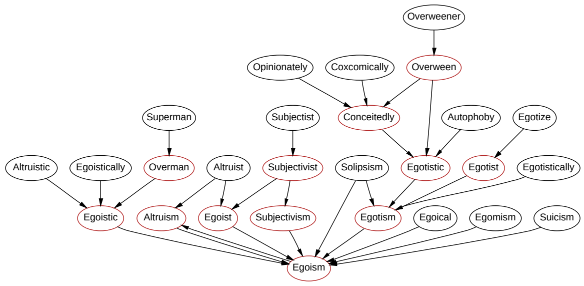

Figure 1: Automatic Thesaurus Extraction from Dictionary

This edition of Webster's dictionary was originally published in 1913, and was recently (1996) converted to text format from scanned images. The resulting text is tagged to mark the parts of the definitions. Definitions from the dictionary data for **Egoism** are shown below:

```
<p><hw>E"go*ism</hw> <pr>(?)</pr>,
<pos>n.</pos> <ety>[F.
<ets>\'82go\'8bsme</ets>,
fr. L. <ets>-ego</ets>
I.  See <er>I</er>, and
cf. <er>Egotism</er>.]</ety>
<sn>1.</sn> <fld>(Philos.)</fld>
<def>The doctrine of certain extreme
adherents or disciples of Descartes and
Johann Gottlieb Fichte, which finds all
the elements of knowledge in the
<xex>ego</xex> and the relations which
it implies or provides for.</def></p>

<p><sn>2.</sn>
<def>Excessive love and thought of self;
the habit of regarding one's self as the
center of every interest; selfishness; --
opposed to <xex>altruism</xex>.</def></p>
```

For the purpose of our application we were interested in the head words <hw>...</hw> and definitions <def>...</def>. Head words and definitions are in a many to many relationship, as each definition refers to different senses of a term, and each head word has alternate spellings. We constructed a directed graph from the definitions as follows:

1. each head word and definition grouping is a node
2. each word in a definition node is an arc to the node having that head word

We assumed that such a graph would be very simple to construct, but immediately ran into problems:

- syllable and accent markers in head words
- misspelled head words
- mis-tagged fields
- stemming and irregular verbs (Hopelessness)
- common abbreviations in definitions (etc.)
- undefined words with common prefixes (un-)
- multi-word head words (Water Buffalo)
- undefined hyphenated and compound words

Even after accounting for accented characters, a naive script mis-assigns over five percent of the words, because of these differences between the actual data in the dictionary and its assumed structure. Any errors in the computation of the graph would affect any subsequent computation of related terms for the thesaurus application. Therefore, we set a goal of 99% accuracy in the conversion of the dictionary data to a graph structure.

Constructing the Congruity Expression

The initial script for the congruity measure for the thesaurus graph application over the dictionary consists roughly of the following operations (regular expressions are partially elided for conciseness of presentation):

```
match object dictionary(.*) {
      // matches entire dictionary
  match set group(<hw>\(...\)*<hw>) {
      // matches definition groupings
    match set hw(<hw>[^>]*</hw>) {
      // matches single head words
      convert ('["'*]', '')
      // removes syllable & accent markers
    }
    match set def(<def>\(...\)*</def>) {
      // matches single definitions
    }
  }
}
```

This script creates an object that represents the entire source, which is then subdivided into chunks containing at least one head word and one definition, which are then extracted into separate sets within the chunk. Each script fragment such as the one above represents operations in the course of a single pass through the source data, the output of which may be passed to another script. This initial script very approximately expresses the congruity relationship between the dictionary and the thesaurus application. For lack of space, the final script, containing over 400 conversion operations, that perform over 300,000 transformations, is not presented here. Instead, in the following section, we show how the S operator allows us to capture some of the classes of conversions which enhanced the above initial script.

Ontology Maintenance

In this section we see how we use the algebra to iteratively refine the dictionary context, by discovering anomalies in the source data, and incorporating resulting fixes into the consistency guarantees of the context. We show how this process is equivalent to the maintenance process when the underlying source data changes.

Context Refinement

The S operator provides a simple method for assessing the contents of a context. For example, $\mathsf{S}_{\texttt{len(hw)div20}}(\texttt{dictionary})$ returns the entries of the dictionary, grouped by length of the head words. Applying this operation on the actual data revealed terms with missing end tags in the data (implying long head word length). Once the errors were identified, the rules to convert terms with missing end tags are added to the definition of the set "`hw`" above. Using S we were also able to determine that other tags were equally valuable as head words, that we needed to remove accentuation from foreign words, and discover spelling errors in the head words by analyzing the frequency of words found in definitions, but not as head words. The context refinement algorithm is as follows:

1. For i in $\{1, \cdots, n\}$
2. $c_{i'} = \mathsf{S}_{e_i}(c_i)$
3. Generate rule(s) r_i to handle exception objects
4. Insert r_i into ruleset for c_i creating c_{i+1}
5. Generate e_{i+1}

Maintaining the Ontology

In the course of developing the wrapper to the Webster's dictionary a major revision of the source data occurred, affecting 10%-25% of all of the entries. These changes are part of an ongoing effort to correct and extend the dictionary, and they included corrections in the tagging of the entries, spelling corrections, reformatting of the text, addition of notes and comments, etc. By maintaining statistics with the S operator on the process of extracting the relevant parts of the dictionary, we were able to note which rules were no longer needed because the exception they handled had been updated. A comparison of the terms that we could not classify in the old and updated sources, revealed new errors that had been introduced in the data. As it turns out there was relatively little within the wrapper that required correction when the source changed. Having the congruity measure within the algebraic framework significantly simplified the process of identifying and handling the changes.

Future Work

In the previous sections we have described how we have used the ontology algebra to iteratively create and refine a context that establishes a dictionary ontology, and then maintain it as the underlying source changes. Here we discuss an application of the dictionary ontology, how they relate to the ontology algebra, and current directions of our research.

Using the Dictionary Ontology

The semantics of the relationship between terms in the dictionary ontology is that the second term contributes to the meaning of the first. We have created an ArcRank algorithm based on PageRank (Page & Brin 1998) that iteratively determines the most important arcs between terms in the ontology, using the results to generate hierarchies of related terms as shown in the graph for **Egoism** above. We are preparing a web based interface to display the results of this work, which will be linked to `http://www-db.stanford.edu/SKC/` our research group's website. To our knowledge this is the first algorithm to evaluate the strength of relationships between terms in a corpus of this magnitude, without any special preprocessing.

Conclusion

In this paper we have presented a case study demonstrating advantages of the algebraic approach to ontology management. An on-line Webster's dictionary represents an ideal test bed for the use of our ontology algebra on real world problems. We used a `Summarize` operation to define and refine a context that prepares the dictionary data for use by a thesaurus service. We showed how the consistency guarantees established for the context were for the most part preserved in the face of substantial changes to the source data. We have looked at the future directions of research on the dictionary data and its relevance as a tool to support the ontology algebra.

Acknowledgements

Thanks are due to Prasenjit Mitra who carefully reviewed drafts of the paper, and to Erich Neuhold and Rudi Studer whose comments and critiques guided the development of this work.

References

Bergamaschi, S.; Castano, S.; and Vincini, M. 1999. Semantic integration of semistructured and structured data sources. *SIGMOD Record* 28(1):54–59.

Cohen, W. W. 1998. Integration of heterogeneous databases without common domains using queries based on textual similarity. In *ACM SIGMOD '98*, 201–212. ACM.

Deerwester, S.; Dumais, S. T.; Landauer, T. K.; Furnas, G. W.; and Harshman, R. A. 1990. Indexing by latent semantic analysis. *Journal of the Society for Information Science* 41(6):391–407.

Encyclopedia Britannica. 1999. Encyclopedia britannica online. http://www.eb.com/.

Fowler, M. 1998. *UML Distilled*. Reading, Massachusetts: Addison-Wesley.

Geraci, A. 1996. The computer dictionary project: An update. *Computer* 29(7):95.

Guha, R. V. 1991. *Contexts: A Formalization and Some Applications*. Ph.D. Dissertation, Stanford University.

Jannink, J.; Mitra, P.; Neuhold, E.; Pichai, S.; Studer, R.; and Wiederhold, G. 1999. An algebra for semantic interoperation of semistructured data (extended version). Technical report, Stanford University.

Karp, P. D. 1992. The design space of frame knowledge representation systems. Technical report, SRI International Artificial Intelligence Center.

Kleinberg, J. 1998. Authoritative sources in a hyperlinked environment. In *Proceedings of the 9th ACM-SIAM Symposium on Discrete Algorithms*. http://simon.cs.cornell.edu/home/kleinber/auth.ps.

Miller, G. A., and *al.* 1990. Five papers on WordNet. Technical Report 43, Cognitive Science Laboratory, Princeton University.

Mirriam-Webster. 1999. Wwwebster dictionary. http://www.m-w.com/.

Page, L., and Brin, S. 1998. The anatomy of a large-scale hypertextual web search engine. *Proceedings of the 7th Annual World Wide Web Conference*.

PROMO.NET. 1999. Project Gutenberg. http://www.gutenberg.net/.

Siegel, M., and Madnick, S. E. 1991. A metadata approach to resolving semantic conflicts. In *Proceedings of the 17th International Conference on Very Large Data Bases*, 133–145.

Smith, D. R. 1993. Constructing specification morphisms. *Journal of Symbolic Computation* 15:571–606.

Uschold, M.; Healy, M.; Williamson, K.; Clark, P.; and Woods, S. 1998. Ontology reuse and application. In *Formal Ontology in Information Systems, FOIS'98*.

Wiederhold, G. 1994. An algebra for ontology composition. In *Proceedings of 1994 Monterey Workshop on Formal Methods*, 56–61. U.S. Naval Postgraduate School.