Chapter 9

MATCHING AND MAPPING FOR SEMANTIC WEB PROCESSES

Tanveer Syeda-Mahmood[1], Richard Goodwin[2], Rama Akkiraju[2], Anca-Andreea Ivan[2]

[1]*IBM Almaden Research Center, 650 Harry Road, San Jose, CA 95120 -. stf@almaden.ibm.com*

[2]*IBM Watson Research Center, 19 Skyline Drive, Hawthorne, NY- rgoodwin,akkiraju,ivananca@us.ibm.com*

1. INTRODUCTION

A semantic revolution is happening in the world of enterprise information integration. This is a new and emerging field that blurs the boundaries between the traditional fields of business process integration, data warehousing and enterprise application integration. By information integration, we mean the process by which related items from disparate sources are integrated to achieve a stated purpose. For example, in data warehousing, data from two separate databases may need to be merged into a single database. This is particularly needed during mergers and acquisitions, where the respective company information from two separate databases may need to be merged into a single database. The terminology used to describe the same information in two disparate sources is hardly identical, subject to the vagaries of human use. Figure 9-1 illustrates two schemas from two databases that need to be reconciled during a data warehousing task. The two tables are called PurchaseOrder and POrder, respectively. They consist of 4 columns with names as shown. To properly merge such schemas, we need to reconcile the two terminologies and find their semantic relationships. Ordinarily, this is the job of a data warehousing specialist, who manually identifies the relationships using an application's user interface. Recent research is trying to make this process semi-automated by performing candidate matching between the names automatically, and having people verify the mappings.

POrder			
Sale Price	ItemID	ItemNumber	UnitOfMesaure

PurchaseOrder			
BrandID	Price	Qty	UoM

Figure 9-1. Illustration of schema matching in a data warehousing scenario.

Consider another scenario, now in the context of business process integration. Here a typical task may be a business flow that routes the data between suppliers and their associated applications. Typically, such flows are composed by business analysts who have limited programming skills, and work with user-interfaces that aid in the creation of business flows. They work with an abstraction of data being routed through schemas called business objects. Examples include generic business objects and application specific business objects made popular by CrossWorld (CrossWorld (2002)) a company that was later absorbed by IBM. These business objects are often encoded in XML syntax but are really structured data as illustrated in Figure 9-2. Here two business objects are depicted that come from two separate business applications, say, SAP (SAP (2005)) and Oracle e-Business Suite (Oracle (2002)) that both describe the concept 'Inventory'. The interface descriptions are shown here in the form of a tree for purpose of illustration here. In order to transform the output of one application into the next in a business flow, mapping of attributes from source to target schema is again needed. One such mapping is shown in Figure 9-2. The closely related terms shown by the arrows include some obvious cases such as terms (OrganizationID, OrgID) as well as non-obvious ones such as (InventoryType, StockType).

Our final example comes from the domain of web services. Service-oriented architecture is the latest trend in distributed computing where the need-to-know abstraction of object-oriented programming is again deployed. In service-oriented architecture, the capability of a code component anywhere on a network is described through an interface language called Web Service Definition Language (WSDL) (Chinnici, R., M. Gudgin, et al. (2003)). A WSDL describes a service as a collection of operational interfaces and their type specification, together with deployment information. Let's look at an extract of a WSDL description of an inventory checking service of an electronics company XYZ as depicted in Figure 9-3a.

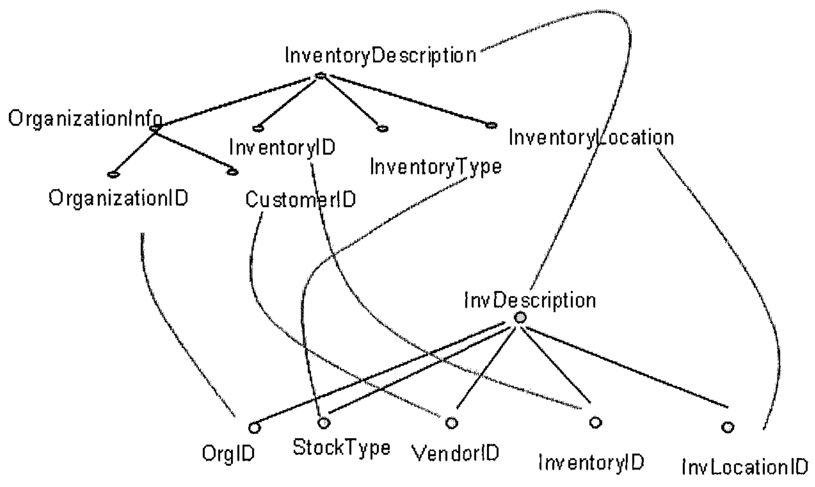


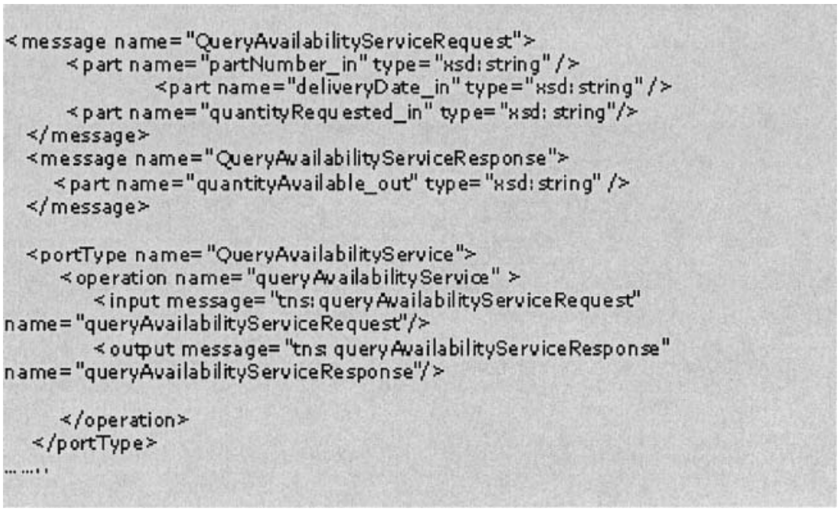
Figure 9-2. Illustration of semantic schema matching in a business process modeling scenario.

We can observe that the WSDL document follows the XML syntax. A set of operations supported by a service are encapsulated in a description using the PortType tag. The PortType in turn lists the operations supported by the service. Each operation lists the inputs and outputs the service takes in the form of messages. In this example, the actual inputs and outputs are expanded in QueryAvailabilityServiceRequest and QueryAvailabilityServiceResponse message tags. Inside each message declaration are the name and type declarations of the inputs and outputs. Here the message shows that it takes the requested item's part number, delivery date and the requested quantity as inputs, and returns the quantity available to be delivered on the requested date as output.

Despite the advancement in service abstraction, the WSDL specification does not prescribe the use of consistent terminology to express the capabilities and requirements of services. Thus two services that accomplish the same task may use different terms to describe similar operations. In some cases, the similarlity between the terms could be spotted through lexical similarity of names, while in other cases, such similarity can only be discovered through the use of domain-specific information. To illustrate this, let's consider a service related to the one depicted in Figure 9-3a. This web service is offered by ABC Inc. and also checks inventory. Its description is shown in Figure 9-3b. We notice first that ABC calls it CheckInventoryService and its inputs and outputs are different from the ones offered by XYZ company's QueryAvailabilityService. ABC's service requires a Universal Product Code instead of a manufacture's part number. The term dueDate is used rather than DeliveryDate and NumberOfItems is used rather than Quantity. Also, ABC's service just returns an ItemAvailabilityConfirmation, which is true if the requested quantity is available and false otherwise. On the other hand, XYZ's service indicates when a request can be partially filled, by returning the number of available items.

As can be seen, there are differences in the interfaces of the services. However, if the objective is to find a service that gives information about the availability of a given part, both services could be semantically similar. In order to chain a sequences of services such as the one above, or to select a similar service from a pool based on a desired interface such as the one shown in Figure 9-3a, we need to find the semantic match between the input or output descriptions present in these WSDL schemas.

This last example also illustrates that finding semantic relationship may require the use of both domain-independent and domain-specific information. A domain independent source of clues gives us a breadth of coverage for common terms, while a domain specific ontology can give a depth of coverage by providing clues based on industry and application specific terms and relationships.

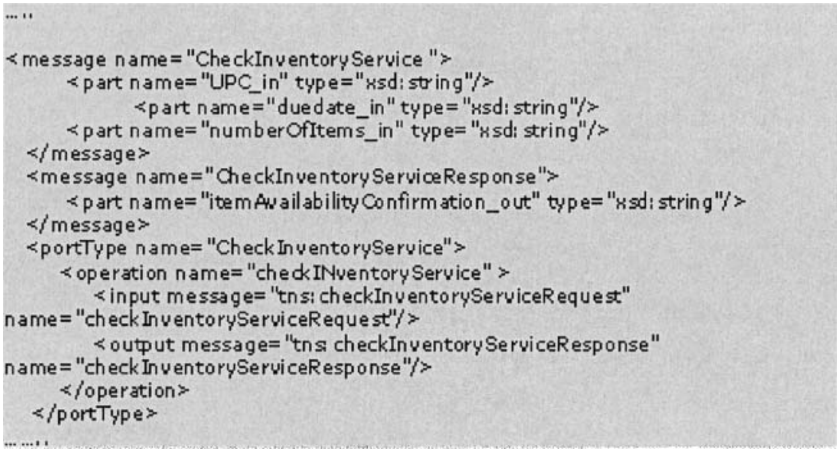


```
<message name="QueryAvailabilityServiceRequest">
      <part name="partNumber_in" type="xsd:string" />
            <part name="deliveryDate_in" type="xsd:string" />
      <part name="quantityRequested_in" type="xsd:string"/>
  </message>
  <message name="QueryAvailabilityServiceResponse">
     <part name="quantityAvailable_out" type="xsd:string" />
  </message>

  <portType name="QueryAvailabilityService">
     <operation name="queryAvailabilityService" >
         <input message="tns:queryAvailabilityServiceRequest"
name="queryAvailabilityServiceRequest"/>
         <output message="tns:queryAvailabilityServiceResponse"
name="queryAvailabilityServiceResponse"/>

     </operation>
  </portType>
.......
```

(a)



```
.....

<message name="CheckInventoryService ">
      <part name="UPC_in" type="xsd:string"/>
            <part name="duedate_in" type="xsd:string"/>
      <part name="numberOfItems_in" type="xsd:string"/>
  </message>
  <message name="CheckInventoryServiceResponse">
      <part name="itemAvailabilityConfirmation_out" type="xsd:string"/>
  </message>
  <portType name="CheckInventoryService">
     <operation name="checkINventoryService" >
         <input message="tns:checkInventoryServiceRequest"
name="checkInventoryServiceRequest"/>
         <output message="tns:checkInventoryServiceResponse"
name="checkInventoryServiceResponse"/>
     </operation>
  </portType>
.......
```

(b)

Figure 9-3. Illustration of the schema matching in a web service scenario.

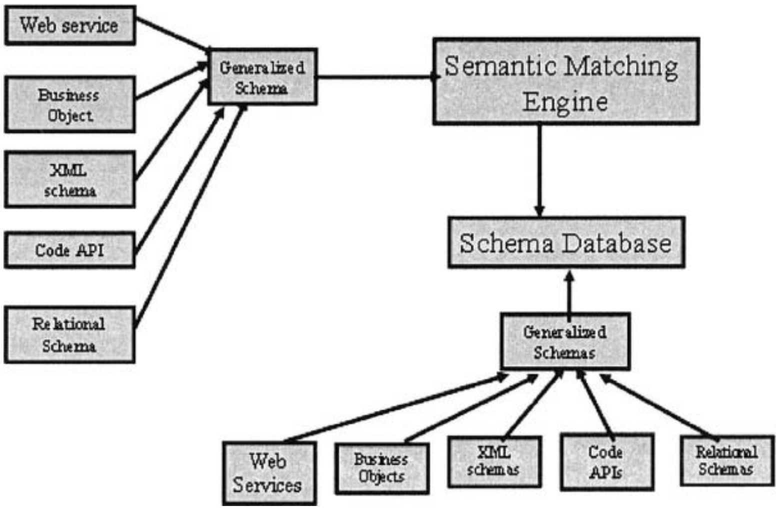


Figure 9-4. Generalized schema matching by normalizing schemas of different origin.

2. SEMANTIC MATCHING AND MAPPING

As we saw from the above scenarios, matching and mapping of schemas is a problem that is applicable in different contexts and would need to be independent of the nature of schemas used in the semantic web process. Further, we saw that there is a need for bridging the semantic gap between the descriptions in order to make true information integration feasible. The field of semantic matching and mapping has now emerged as a new and exciting field to address these problems of semantic mismatch of descriptions using automated relationship discovery techniques.

We can now define the semantic schema matching problem as follows. Given a source and a target schema defined it terms of its attributes and relationships, find a way to semantically match the schema attributes in a way that is independent of the schema origin. Since different schema origins have different nuances, the schema matching techniques would have to be agnostic to the details of the schema format, but at the same time, capture the underlying name, type and structure relationships described therein. One way to achieve this is to develop a generic schema representation that captures the essential information across different schema formats, and then use this general schema representation as the basis for matching. This approach is illustrated in Figure 9-4. Here schemas arising from different application domains are reduced to a normalized format called the generalized schema. The semantic schema matching is then performed between a pair of source and target generalized schemas.

2.1 Generalized Schema

Schemas of different origin such as code APIs, Web services, XSD (XMLSchema (2004) can be reduced to a normalized format called the Generalized Schema using the following simple grammar.

Gs->NaCtTyRsUdOjGs* (1)

Where Na stands for the name of the schema, Ct stands for its category (eg. Its origin as WSDL, XSD, etc.), Ty stands for its type (eg. A complexType or simpleType), Rs stands for any restrictions on its values (eg. Range of values supported), Ud stands for a simple user-friendly name for the schema (as exposed through user interfaces), and Oj stands for the original schema object from which the normalized schema is derived. The Generalized schema can be recursively expanded to describe the structure in its full detail. The type expansions of each of the symbols in the above grammar are given below:

```
Na->a String
Ct-> a String
Ty->primitive type|language-defined type
Rs->language-defined restrictions
Ud->User-friendly name
Oj->Language-defined object instance
Primitive type --> int|char|String|double|Boolean|Byte|Char|Short|Integer|Long|Float|Double
```

The above normalized format for schemas has been used earlier for representing code objects (D. Caragea et al. (2004)) and for web services (Syeda-Mahmood et al. (2005)). It can be shown that many abstract data types supported in schemas can be modeled by the above generalized schema. In fact, automatic conversion programs can be written to transform incoming schemas from any of the formats described in Figure 9-4 into Generalized Schema.

3. A FRAMEWORK FOR SCHEMA MATCHING

Let us now consider the problem of semantic schema matching using the generalized schema representation. As defined in Section 2, this is the problem of matching the attributes of the source and target schemas. Ideally, we would like the matching to be 'best' in some objective sense. In other words, we seek a 'best' correspondence of source and target schema attributes. A general way to model such correspondence is to treat the source and target schema attributes as two sets of nodes of a bipartite graph as shown in Figure 9-5. An edge can then be drawn between a source and target node, if the corresponding attributes are semantically similar. Finding the best set of matching attributes then reduces to the problem of finding the maximum matching in the bipartite graph, i.e. with the largest pair of nodes matching. A matching in a bipartite graph is formally defined as a subset of edges of the bipartite graph such that there is a unique assignment for the selected source and target attributes.

Thus the problem of determining an optimal correspondence between the source and target schemas can be expressed as the problem of finding a maximum matching in the bipartite graph. Figure 9-5 illustrates such a maximum matching. On the left is the original bipartite graph formed from the attributes in the pair of source and target schemas. Here we see that multiple edges emanate from source and target attributes indicating there is more than one possible match for an attribute. In the maximum matching, selected attributes are paired with unique matches. The size of the matching is 5 indicating that at most 5 attributes find a match in this arrangement.

In practice, the semantic similarity between attributes is actually reflected through a similarity score which can be treated as a weighted edge. The optimal matching desired in that case is then a matching of maximum cardinality and maximum weight as well. Well-known algorithms are available in literature to obtain such a matching using variants of the maximum flow algorithm (A. Goldberg and Kennedy (1993), J.E. Hopcroft, R.M. Karp (1973)). In these algorithms, the matching is computed by setting up a flow network, with weights such that the maximum flow corresponds to a maximum matching.

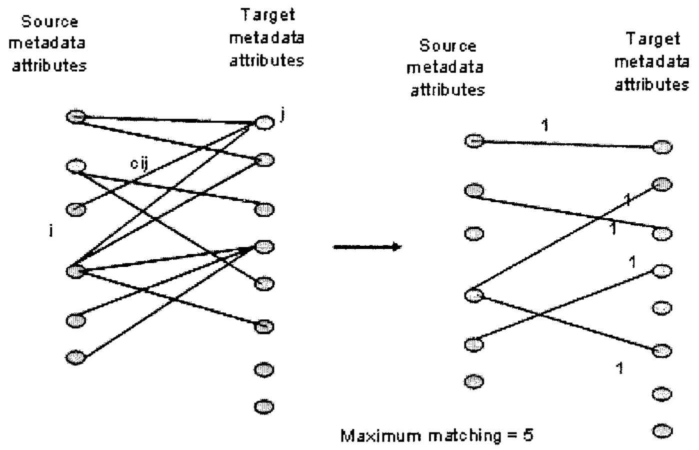

Source metadata attributes
Target metadata attributes
j
cij
i
Source metadata attributes
Target metadata attributes
1
1
1
1
1
Maximum matching = 5

Figure 9-5. Bipartite graph matching framework for schema matching.

Algorithms for finding the maximum matching involve compute-intensive operations as they solve the network flow optimization problem. Often, a good lower bound on the size of the matching can be quickly obtained using a greedy matching algorithm in which the edges are sorted in cost and picked in descending order starting with the highest scoring edge and deleting all edges emanating from the selected pair of attributes.

Notice we have not yet described how the similarity between attributes can be determined. But assuming that such a similarity score can be developed, we now have a general way of picking the best possible subset of edges, and hence a best matching of the attributes of the respective schemas using the above framework for bipartite graph matching.

4. FINDING SEMANTIC SIMILARITIES BETWEEN ATTRIBUTES

Several cues can be exploited to define the cost of edges in the above framework. In particular, we can exploit the similarity in name, type, or structure to define a semantic similarity score. In this section, we describe some of the popular approaches to capturing semantic similarity between attributes.

4.1 Lexical Comparison of Terms

The simplest one is to do a lexical comparison of their names using a variant of string matching algorithms. A popular approach is to take the longest common subsequence of the two names of attributes being considered (Cormen et al, (1994)). For example, the longest common subsequence between pair '*customer*' and '*custmr*' is '*custmr*' of length 6. A popular formula for finding the similarity between terms on a lexical basis is:

$$\mathrm{Lex}(A,B)= |\mathrm{LCS}(A,B)|/|A|+|B| \qquad (2)$$

Where LCS(A,B) is the longest common subsequence between strings A and B and the | | stands for the length of the strings. The LCS measure is good for capturing obvious similarities in name of the type above, and also when terms differ by numeric values, or are abbreviations. Examples include, (Address1, Address2), (Num, Number), etc. However, a score value has to be sufficiently high to be a meaningful similarity to avoid false positives. It is very easy for a sequence of symbols to be common without any basis of semantic similarity. Examples include (Address, Adroit), (summary, summon), etc.

4.2 Semantic Similarity of Terms

Next, we address cases where the terms are not syntactically similar but semantically related. A thesaurus is usually employed for this purpose. Among the popular ones are WordNet, a free thesaurus (G.A. Miller (1995)), and SureWord (SureWord (2005)), a commercial thesaurus software for English language.

To determine the semantic similarity of terms we have to first tokenize the multi-word term. Part-of-speech tagging and stop-word filtering has to be performed. Abbreviation expansion may have to be done for the retained words. A thesaurus can then be used to find the similarity of the tokens based on synonyms. The resulting synonyms are assembled back to determine matches to candidate multi-term word attributes, after taking into account the tags associated with the attributes. The details of these operations are described below.

4.2.1 Work Tokenization

To tokenize words, common naming conventions used by programmer analysts, DBAs and business analysts may have to be exploited. In particular, word boundaries in a multi-term word attribute can be found using changes in font, presence of delimiters, such as underscore, spaces, and numeric to alphanumeric transitions. Thus words such as CustomerPurchase can be separated into Customer and Purchase. Address_1, Address_2 would be separated into Address, 1 and Address, 2 respectively.

4.2.2 Part-of-speech tagging and filtering

Simple grammar rules can be used to detect noun phrases and adjectives. Stop-word filtering when performed using a pre-supplied list can help further pruning. Common stop words in the English language similar to those used in search engines include words such as and, or, the, etc.

4.2.3 Abbreviation expansion

The abbreviation expansion operation can exploit domain-independent as well as domain-specific vocabularies. It is possible to have multiple expansions for a candidate words. All such words and their synonyms can be retained for later processing. Thus, a word such as CustPurch can be expanded into CustomerPurchase, CustomaryPurchase, etc.

4.2.4 Synonym search

A language thesaurus such as SureWord or WordNet can be used to find matching synonyms to words. Using SureWord, it is possible to assign to each synonym, a similarity score based on the sense index, and the order of the synonym in the matches returned.

4.2.5 Semantic similarity scores

Given a pair of candidate matching multi-term attributes (A, B) from the source and destination schemas, we can generate a similarity score between the attributes by combining the match scores returned by a thesaurus for their word tokens as follows.

Let A and B have m and n valid tokens respectively, and let S_x and S_y be their expanded synonym lists based on semantic processing. We consider each token i in source attribute A to match a token j in destination attribute B where i ε S_x and j ε S_y. The semantic similarity between attributes A and B is then given by

$$Sem(A, B) = 2*Match(A,B)/(m + n) \quad (3)$$

where Match(A, B) are the matching tokens based on the definition above.

Using the similarity scoring such as above, we can determine semantically similar attributes such as (state, province) for the single token case, to (CustomerIdentification, ClientID), (CustomerClass, ClientCategory), for the multi-term attributes.

4.3 Ontological Similarity of Terms

In addition to domain-independent thesaurus, schema matching can be aided by domain-specific terminology. In fact, each organization usually has a glossary of terms compiled that are specific to their domains, such as a banking glossary, electronics parts glossary, etc. With the newly developed standards, it is now possible to represent complete ontologies in formats such as OWL (OWL, (2004)).

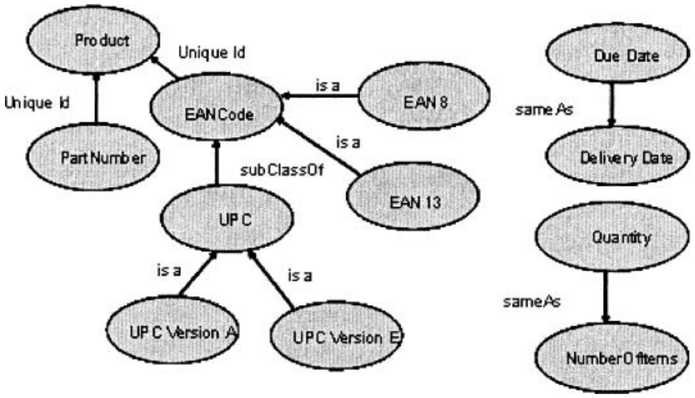


Figure 9-6. Illustration of a simple domain ontology.

To discover similarities between attributes by consulting ontologies, they would first have to be loaded into an ontology management system. An example of such a system is SNOBASE (Lee et al. (2003)), that can reason with concepts and supply similar concepts by derivation from the defined concepts in the ontology. A simple domain-specific ontology that models the relationships between electronic parts is indicated in Figure 9-6. As can be seen, four different types of relationships between two concepts A and B are modeled, namely, subClassOf, superClassOf, instanceOf, and equivalenceClass. Larger ontologies may model many more relationships.

4.3.1 Finding related terms in an ontology

Given a domain-specific ontology and a term from the source schema, how can we find a matching term in the destination schema? Using rule-based inference in the ontology, we can recover all potential similar terms that are in one of the specified relationships, such as subclass, superclass, etc. The matches returned are a set of related concepts along with distance scores representing distance between them. A simple scoring scheme to compute distance between related concepts in the ontology could be as shown in Table 9-1. The discretization of the score into three values (0, 0.5, 1.0) gives a coarse idea of semantic separation between ontological concepts. For example, in the electronics domain ontology shown in Figure 9-6, concepts DueDate and DeliveryDate have a distance of 0 while EANCode and UPC have a distance of 0.5. More refined scoring schemes are possible, but a simple choice such as the one in Table 9-1 works well in practice, without causing a deep semantic bias. Thus given a source attribute DueDate, we can retrieve ontologically matching concepts as the terms DeliveryDate, while a source term "UPC" will return as related concepts (EAC code, Part Number, EAN8, EAN13,UPCversion A, and UPC version E using inference in the ontology of Figure 9-6. In practice, we can choose a suitable threshold T so that all related concepts with distance scores above T can be ignored.

Once the related concepts are found, we can search for these terms in the destination schema and record them as matching attributes to the given attributes from the source schema. Instead of finding ontologically similar terms directly from the attributes of the source schema, it often makes sense to invoke such similarity on annotations associated with the source and destination schemas. Such annotations are usually manually attached by domain experts and are likely to be well-defined terms rather than the cryptic abbreviated multi-term phrases that technical personnel used to name attributes of database and other schemas. As for the inference itself, several rule-based engines are available for reasoning with ontologies including the ABLE (Bigus et al. (2001)) system that uses Boolean and fuzzy logic, forward chaining, backward chaining etc. Rule sets created using the ABLE Rule Language can be used by any of the provided inference engines, which range from simple if-then scripting to light-weight inference to heavy-weight AI algorithms using pattern matching and unification.

Table 9-1. Illustration of a Ont(A,B) for different relationships in the ontology.

Concept Pair	Relationship	Distance Score Ont(A,B)
(A,B)	EquivalentClass	0
(A,B)	RDFType	0
(B,A)	SubClassOf	0.5
(A,B)	SubClassOf	0.5
(A,B)	Other	1

4.4 Type and Structural Similarity of Attributes

So far, we have considered each attribute on an individual basis. However, there are inter-relationships between attributes that need to be respected such as their associated types and positions in schema structure. We now discuss how type and structural information can be taken into account during similarity computations.

4.4.1 Type similarity

For schemas that correspond to code APIs the type of attributes is a strong cue in matching. Specifically, unless the type can be properly cast, the destination component cannot be launched even if the schema matching says otherwise. One way to capture the type similarity is to take the help of the reference type hierarchy defined for the language specification such as XSD, Java, etc. If the conversion is possible but will cause a loss of data (eg. *float* to *int* conversion), then we attach a lower weight. Lossless type conversion (eg. *int* to *float*) and other equivalent subclass type inheritance and polymorphism can be given higher weights. If the similarity cannot be inferred using the reference type hierarchy, explicit user-defined data type conversion functions may exist. For example, a 2D to 1D data type conversion, such as an array to vector conversion is not allowed in the reference type hierarchy but can be achieved through an explicitly written conversion function.

A simple reference type similarity measure can be given by

$$\text{Type(A,B)} = \begin{cases} 1.0 \text{ for lossless type conversion or if type conversion function exists} \\ 0.5 \text{ for lossy type conversion} \\ 0.0 \text{ otherwise} \end{cases} \qquad (4)$$

4.4.2 Structural similarity

The structural similarity of schemas can be captured in many ways. A simple way is to consider each level in the schema as representing a grouping of related concepts. For example, all related aspects of a data structure are grouped under an abstract data type by programmers. These in turn may be composed of substructures which are suitable abstract data types formed from lower level type structures. The leaf level attributes in such cases are usually attributes with type primitives such as int, float, etc. Thus structural similarity in the attributes can be measured by the difference in the tree depth at which the attribute occurs. If we record the depth of the attribute from the root node of the schema, the structural similarity between two attributes A and B from source and destination schemas respectively can be given by

$$\text{Struct(A,B)} = 1 - \frac{(|D(A) - D(B)|)}{\max\{D(G_A), D(G_B)\}} \qquad (5)$$

where D(A) and D(B) are the depths of the attributes in their respective schema trees G_A and G_B .

4.5 Combining Similarity of Attributes

As we saw in the above sections, there are many cues that can be used to compute the similarity of attributes. To use these measures in the graph matching framework of Section 3, we need to combine them into an overall similarity measure. Here again, several choices are possible, including linear combination, probabilistic fusion (Kahler et al., (2004)), etc. Here we describe a simple weighted linear combination, where the relative contributions of each cue can be tuned based on the origin of the schemas. For example, the type cue may be more important for API schemas, while the name may be more important for business objects. The overall similarity of a pair of attributes A, B from source and destination schemas respectively can then be given by.

$$Sim(A,B) = \alpha_1 Lex(A,B) + \alpha_2 Sem(A,B) + \alpha_3 Ont(A,B) + \alpha_4 Type(A,B) + \alpha_5 Struct(A,B) \qquad (6)$$

The above similarity score can be used as the edge score in the graph matching framework and a maximum matching can be derived used network flow optimization methods as described in Section 3.

5. SUMMARY

In this chapter the matching and mapping problem for web processes has been introduced. We have seen that the matching of schemas is a general problem for schemas derived from a variety of application domains. A graph matching framework has been described for addressing the mapping and matching of semantic web process. Multiple cues for determining the similarity of attributes has been defined based on name semantics, type and structural information. The emergence of a general paradigm for accommodating the matching and mapping problem from several different domains ranging from business process modeling to schema integration, is a significant advancement in the development of semantic web processes.

6. RELATED WORK

The schema matching problem has been addressed by a number of researchers from both database and web service communities. Recently, clustering and classification techniques from machine learning are being applied to the problem of web service matching and classification at either the whole web service level (Hess et al. (2003)) or at the operation level (Dong, (2004)). In (Hess et al. (2003)) for example, all terms from portTypes, operations and messages in a WSDL document are treated as a bag of words and multi-dimensional vectors created from these bag of words are used for web service classification. The paper by Dong et al. addresses this aspect by focusing on matching of operations in web services. Specifically, it clusters parameters present in input and outputs of operations (i.e. messages) based on their co-occurrence into parameter concept clusters. This information is exploited at the parameter, the inputs and output, and operation levels to determine similarity of operations in web services. The notion of elemental and structural level schema matching has been present in the METEOR-S project (Patil et al. (2004)), where the engine can perform both element and structure level schema matching for Web services. The element level matching is based on a combination of Porter-Stemmer (Porter , (1980)) for root word selection, WordNet dictionary for synonyms (Miller (1995)), abbreviation dictionary to handle acronyms and NGram algorithm for linguistic similarity of the names of the two concepts. The schema matching examines the structural similarity between two concepts. Both element match score and schema match score are then used to determine the final match score.

The problem of automatically finding semantic relationships between schemas has also been addressed by a number of database researchers lately

(Madhavan et al., (2003)), (Rahm and Bernstein, (2001)), (Madhavan et al. (2001)). Thus algorithms are available for XML schema matching such as Clio (Miller et al. 2001), Cupid (Madhavan et al. (2001)), and similarity flooding (Melnik et al. (2002)). In the case of database schema matching both schema content (i.e. data) and names of attributes are exploited for schema matching.

The use of ontology match making engines for semantic matching has also been explored by a number of researchers. One of the earliest ontology-based semantic matchmaking engines is Sycara et al MatchMaker (Sycara, (1999)) that is available on the Web as a service. In addition to utilizing a capability-based semantic match, the engine also uses various other IR-based filters. Another related effort is Racer (Li and Horrocks, (2003)), that focuses solely on a service capability-based semantic match for application in e-commerce systems. In a recent work, both ontological and semantic similarity cues were combined to address the larger problem of semantic search which embeds semantic schema matching (Syeda-Mahmood et al, (2005)).

7. QUESTIONS FOR DISCUSSION

Beginner:
1. Name some real-world problems that have been solved by maximum matching in bipartite graphs.
2. What is the difference between schema matching and schema mapping?

Intermediate:
1. If both domain-specific and domain-independent ontologies had to be used, how would you prioritize the matches to attributes?
2. Suggest other combination schemes for cues besides the linear combination described in text.
3. Think of other cues that can be used for capturing similarity of attributes. Describe how they can be measured.

Advanced:
1. Can the service composition problem by addressed by the bipartite graph matching framework? If not, suggest modifications to the framework to model composition.
2. In practice, a combination of source attributes may map to a single target attribute (eg. A database join) and vice versa. Can such mappings be handled in the graph matching framework? If not, show how the framework can be adapted to handle such combination mappings.

Practical Exercises:

1. Go to xmlmethods.com. Hand-simulate the schema matching on a pair of web services and postulate what the mappings would be.
2. Now write a program to generate the candidate mappings for an arbitrary pair of web services selected from xmlmethods.com.

8. SUGGESTED ADDITIONAL READING

- R. Fagin and P. Kolaitis and L. Popa and W. Tan (2004), "Composing schema mappings: Second-order dependencies to the rescue", in Proc. of PODS, 2004.
- P. Bernstein et al. (2004): "Industrial-strength schema matching," in SIGMOD Record, Vol. 33, No. 4, pp.38-43, December 2004.

9. REFERENCES

CrossWorlds (2002), http://www306.ibm.com/software/info1/websphere/cw011402.jsp.

SAP (2005), http://www.sap.com.

Oracle (2005), http://www.oracle.com.

Chinnici, R., M. Gudgin, et al. (2003). Web Services Description Language (WSDL) Version 1.2, W3C Working Draft 24, http://www.w3.org/TR/2003/WD-wsdl12-20030124/.

XMLSchema (2004). XML Schema Part 2: Datatypes Second Edition, W3C Recommendation 28 October 2004.

D. Caragea and T. Syeda-Mahmood (2004), "Semantic API matching for web service composition" in Proc. ACM WWW 2004 conference, New York, NY.,pp., 436-439, June '04.

T. Syeda-Mahmood et al. (2005): Semantic search of schema repositories. IEEE Int. Conference on World-Wide Web (WWW), 1126-112.

A. Goldberg and Kennedy (1993) : An efficient cost-scaling algorithm for the assignment problem. SIAM Journal on Discrete Mathematics, 6(3):443-459, 1993.

J.E. Hopcroft, R.M. Karp (1973): An n 5=2 algorithm for maximum matching in bipartite graphs, SIAM Journal on Computing 2, 225-231, 1973.

T.H. Cormen, C.E. Lieserson, and R.L. Rivest (1990): Introduction to Algorithms. New York: McGraw Hill, Cambridge: MIT Press, 1990.

G.A. Miller (1995): Wordnet: A lexical database for English. Communications of the ACM, 38(11):39-41, 1995.

SureWord (2005): http://www.patternsoft.com/sureword.htm.

OWL (2004). OWL Web Ontology Language Reference, W3C Recommendation, World Wide Web Consortium, http://www.w3.org/TR/owl-ref/. **2004**.

Lee J., Goodwin R. T., Akkiraju R., Doshi P., Ye Y.(2003): SNoBASE: A Semantic Network-based Ontology Ontology Management. http://alphaWorks.ibm.com/tech/snobase.

Bigus J., and Schlosnagle D. 2001. Agent Building and Learning Environment Project: ABLE. http://www.research.ibm.com/able/

Olaf Kähler, Joachim Denzler, and Jochen Triesch (2004) : Hierarchical Sensor Data Fusion by Probabilistic Cue Integration for Object Tracking, Image Analysis and Interpretation, 2004. 6th IEEE Southwest Symposium on Object Tracking, pages 216–220.

UDDI Technical Committee. "Universal Description, Discovery and Integration (UDDI)". http://www.oasis-open.org/committees/uddi-spec/

X. Dong et al (2004): "Similarity search for web services," in Proc. VLDB, pp.372-283, Toronto, CA, 2004.

A.Hess and N. Kushmerick (2003): "Learning to attach metadata to web services," in Proc. Intl. Semantic web conference, 2003.

E. Rahm and P. Bernstein (2001): A survey of approaches to automatic schema matching, in VLDB Journal 10:334-350, 2001.

J. Madhavan et al (2001), "Generic schema matching with cupid," in Proc. VLDB 2001.

S. Melnik et al, "Similarity flooding: A versatile graph matching algorithm and its application to schema matching," in Proc. ICDE, 2002.

A.Patil et al. (2004): "Meteor-s web service annotation framework", in Proc. WWW conference, pp. 553-562, 2004.

Porter, M. F. (1980): "An Algorithm for Suffix Stripping." Program 14, 1980, 130-137.

S. Melnik et al. (2002): Similarity flooding: A versatile graph matching algorithm and its application to schema matching. In Proc. ICDE, 2002.

Renee J. Miller et al. (2001): The Clio project: managing heterogeneity. SIGMOD Record (ACM Special Interest Group on Management of Data), 30(1):78-83, 2001.

K. Sycara et al. (1999): "Dynamic service match making among agents in open information environments," in Jl. ACM SIGMOD Record, 1999.

L. Li and I. Horrocks, (2003): " A software framework for matchmaking based on semantic web terminology," in Proc. WWW Conference, 2003.

T. Syeda-Mahmood et al. (2005): "Searching schema repositories by combining semantic and ontological matching," in Proc. IEEE Intl. Conf. on Web Services, (ICWS), pp.13-20, 2005.

J. Madhavan et al. (2003). Corpus-based Schema Matching. In Workshop on Information Integration on the Web at IJCAI, 2003.