Mining Transformation Rules for Semantic Matching

Peter Yeh, Bruce Porter, and Ken Barker

University of Texas at Austin, Department of Computer Sciences
Austin, TX 78712 USA
e-mail: {pzyeh,porter,kbarker}@cs.utexas.edu

Abstract. Semantic matching is finding a mapping between two knowledge representations encoded using the same ontology. Solving this matching problem is hard because the syntactic form of two knowledge representations rarely matches exactly. Previous research has shown transformation rules can be used to improve matching, but acquiring transformations is difficult. In this paper, we present an algorithm for mining transformation rules for semantic matching. This algorithm was evaluated in two domains – battle space planning and chemistry. In both cases, the resulting transformations helped to improve matching significantly compared to using only taxonomic knowledge.

1 Introduction

Many AI tasks require matching two structured knowledge representations encoded using the same ontology to find a mapping between them. For example, rule-based classification requires matching rule antecedents with working memory; information retrieval requires matching queries with documents; and some knowledge acquisition tasks require matching new information with already encoded knowledge to expand upon and debug both of them. We call this problem semantic matching.

Because structured representations can be encoded as graphs, the semantic matching problem is a graph matching problem that finds a mapping between the nodes of two graphs. Solving this matching problem, however, is hard because many valid mappings between two graphs often cannot be established due to syntactic differences. Previous research [16, 17] showed that an imperfect match between two representations can be significantly improved by using transformation rules to transform a mismatched portion in one graph to match exactly a corresponding portion in another graph. These transformations were enumerated (by hand) from a domain-neutral upper ontology. The resulting transformations are domain-neutral in the sense that the contents of these transformations are encoded using generic concepts not specific to any particular domain. Previous research [16, 17] also showed matching can be further improved by using additional domain-specific transformations whose contents are encoded using concepts specific to the domain of the application.

Although transformations can improve matching significantly, it is difficult to acquire these transformations. No algorithms exist for acquiring transformations for semantic matching, and enumerating transformations by hand has two obvious problems. First, we cannot enumerate all possible transformations because many domains have mismatches specific to the domain. These domain-specific mismatches are idiosyncratic

and result from modeling decisions or inherent properties specific to that domain. Second, enumerating transformations by hand is costly and time-consuming. Therefore, having to enumerate domain-specific transformations for every new domain makes using transformations impractical. Most existing semantic matching approaches, therefore, use only taxonomic knowledge [8, 11, 13, 18], which is relatively easy to acquire for both general concepts and domain-specific ones.

Our work extends [16, 17] by automating the task of discovering transformations which are effective at resolving mismatches – especially domain-specific mismatches. Our solution is inspired by previous research in rule mining [1, 2, 9, 10, 12, 14] where the basic idea is to discover associations (and hence association rules) from recurring features within a data set. This approach has successfully discovered association rules in data sets ranging from transactions to text. We believe this basic idea can also be applied to our problem of discovering transformations from structured representations.

Although our approach is inspired by this body of research, it differs in one fundamental way. We are not trying to discover association rules. Rather, we are trying to discover transformation rules to help improve semantic matching. Hence, we are interested in features *not* shared between items in a data set because transformations are intended to resolve mismatches.

2 Background

In our framework, knowledge is represented using simple conceptual graphs which are finite connected bipartite graphs without any nesting or context [15]. The two types of nodes in a simple conceptual graph are concepts and relations, and an ontology provides the vocabulary used to label these nodes.

A **transformation** is a rule of the form $lhs \Rightarrow rhs$ where the lhs and rhs are also simple conceptual graphs encoded using the same ontology as the representations to which they are applied. Applying a transformation transforms the local portion of a graph that can be projected[1] onto by the lhs into an alternative form encoded by the rhs (see [16, 17] for a discussion of transformations and their use for semantic matching).

A **semantic matcher** [8, 11, 13, 16–18] takes two representations (i.e. graphs) and finds a mapping between the concepts and relations[2] of these two graphs based on a match criterion such as graph isomorphism, subgraph isomorphism, Maximal Common Subgraph (MCS), *etc*. We use the MCS criterion [5] to find the largest connected subgraph in one representation that can be projected [6] (i.e. mapped) onto the other representation being matched. The **matched parts** between these two representations are the subgraphs which project onto each other. Subgraphs not included in this projection are the **unmatched parts** (i.e. mismatches). In this paper, we will use $\boldsymbol{g_i}$ and $\boldsymbol{g_i'}$ to refer to the matched and unmatched parts, respectively, of a representation G_i.

A **mismatch point** is a pair of matched concepts connected to unmatched relations. Formally, a mismatch point between two graphs, G_1 and G_2, is a pair of concepts, (c_1, c_2), that satisfies the following properties. First, c_1 is in g_1 and is connected to a

[1] We require this projection to be 1-to-1 and onto – i.e. an iso-projection [6].

[2] A mapping between two relation nodes can be established only if mappings can also be established for the concept nodes they are directly connected to.

relation in g_1'. Second, c_2 is in g_2 and is connected to a relation in g_2'. Finally, c_1 maps to c_2 as defined by the mappings returned from matching G_1 and G_2 (see Figure 1 for an example).

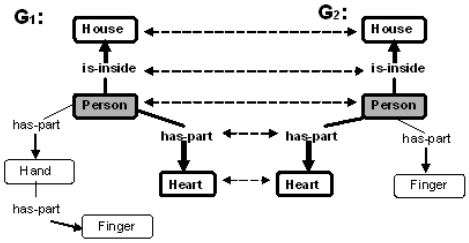

Fig. 1. The matched parts between G_1 and G_2 are highlighted in **bold**. The dotted lines show the mappings between G_1 and G_2. Concepts that are mismatch points are shown in gray.

The degree of match, which we call **score**, computes the similarity between two representations based on their mappings. We require this function to range from 0 (i.e. no match) to 1 (i.e. every node was matched), but leave its definition up to the application.

3 Mining Transformations

Our approach finds instances of mismatches between two knowledge representations that encode sufficiently similar content. These instances are then generalized into transformation rules for use in semantic matching.

3.1 Search Biases

The space of mismatches through which we search for transformations is very large. To search this space efficiently, we exploit two properties of transformations to use as search biases.

First, we observed in [16, 17] that transformations take relatively few syntactic forms. Hence, we only search for mismatches that obey these syntactic forms, which are generalized rules of the form $lhs \Rightarrow rhs$. The lhs and rhs are conceptual graphs with variables in place of concepts and relations (we use upper-case letters for concept variables and lower-case letters for relation variables). For example,

$$X - r \rightarrow Y - r \rightarrow Z \Rightarrow X - r \rightarrow Z.$$

is a syntactic form describing a transitivity mismatch.

A mismatch (i.e. the unmatched parts of G_1 and G_2) obeys a syntactic form F – i.e. $obey(F, g_1', g_2')$ – if four conditions are satisfied. 1) g_1' is isomorphic to lhs. 2) g_2' is isomorphic to rhs. 3) For every concept variable X, either an $is\text{-}a$ or $instance\text{-}of$

relationship must exist between the labels of all the concepts in g'_1 and g'_2 bounded to X. 4) For every relation variable r, the labels of all the relations in g'_1 and g'_2 bounded to r must be the same.

Second, there is an intersection between the lhs and rhs of every transformation because the lhs is transformed into an alternative form encoded by the rhs. We use this fact to reduce the search space by limiting the search to regions of the mismatch space around a mismatch point.

3.2 Algorithm for Mining Transformations

The outline of our algorithm for *Min*ing *T*ransformations, which we call MinT, is shown in Figure 2. MinT takes as input a semantic matcher (we'll call M), two lists of graphs (we'll call D_1 and D_2), a similarity measure (we'll call $score$), and three parameters (we'll call $minscore$, $minsupp$, and $mincert$ for the minimum match score, support, and certainty respectively). It returns as output a list of transformation rules.

GIVEN: $M(Gi, Gj)$, $D_1 = \{G_1, ..., G_n\}$, $D_2 = \{G_1, ..., G_m\}$, and $score(mappings)$
parameters: minscore, minsupp, mincert
FIND: A list of transformations.
PHASE1: Generate candidate transformations.
FOR all pairs (Gi, Gj) where $G_i \in D_1$ and $G_j \in D_2$ DO
LET $mappings = M(Gi, Gj)$
IF $score(mappings) \geq minscore$ THEN
FOR each mismatch point (c_i, c_j) between G_i and G_j DO
FOR each F_k in $\{F_1, ..., F_N\}$ DO
Search g'_i and g'_j starting at c_i and c_j, respectively, for subgraphs sg'_i in g'_i and sg'_j in g'_j that obey F_k.
IF $obey(F_k, sg'_i, sg'_j)$ THEN Add $sg'_i \Rightarrow sg'_j$ to T.
ELSE IF $obey(F_k, sg'_j, sg'_i)$ THEN Add $sg'_j \Rightarrow sg'_i$ to T.
PHASE2: Generalize candidate transformations.
LET $L = \{\}$
FOR each T_i in T DO
FOR each (cg_j, Rj) in L DO
LET cgenl = minimum common generalization of T_i and cg_j.
IF cgenl exists THEN
Change (cg_j, Rj) to $(cgenl, Rj \cup \{T_i\})$ and break from inner FOR loop.
IF no common generalization found for T_i THEN
Add $(T_i, \{T_i\})$ to L.
PHASE3: Filter transformations.
LET $RESULT = \{\}$.
FOR each (cg_j, Rj) in L DO
IF $Support(cg_j) \geq minsupp$ and $\delta(Null(cg_j), cg_j) > Certainty(cg_j, Rj, mincert)$
THEN Add cg_j to $RESULT$.
RETURN $RESULT$

Fig. 2. Outline of our algorithm, MinT.

In the first phase, MinT generates a list, T, of candidate transformations by matching each graph G_i in D_1 with each graph G_j in D_2. If G_i and G_j are sufficiently similar (i.e. the score between G_i and G_j is greater than or equal to $minscore$), then MinT will search the unmatched parts of G_i and G_j around each mismatch point for all subgraphs that obey one of the syntactic forms in $\{F_1, ..., F_N\}$. This search is a breadth-first search rooted at the mismatch point and biased by the syntactic forms. Those subgraphs found through this process are returned as candidate transformations.

In the second phase, MinT groups together those candidate transformations in T that have a common generalization. Two transformations, T_i and T_j, have a common generalization if: 1) T_i and T_j obey the same syntactic form, 2) the labels of all the corresponding concepts (i.e. concepts bound to the same concept variable) between T_i and T_j have a common ancestor, and 3) the labels of all the corresponding relations (i.e. relations bound to the same relation variable) between T_i and T_j are the same. This generalization is minimum if the common ancestor for the labels of all corresponding concepts is the minimum common ancestor. The result is a list of the form $L = \{(cg_1, R1), ..., (cg_n, Rn)\}$ where each element of L consists of a subset Rj of the candidate transformations in T and the minimum common generalization cg_j (which is also a transformation) for this subset.

In the final phase, MinT removes those generalized transformations cg_j from L that do not satisfy the minimum support and certainty requirements. Support is a metric we borrowed from the rule mining literature [1, 2, 9, 10, 12, 14] to measure how frequently a generalized transformation occurs. We define this metric $Support(cg_j)$ as the fraction of sufficiently similar matches between D_1 and D_2 that contains a candidate transformation that cg_j is a generalization of.

Certainty is a metric we introduce to measure a generalized transformation's strength. We argue that overly general transformations have little strength because they can align almost anything. Hence, this metric corresponds to how certain are we that a generalized transformation is not overly general. We measure our certainty in a generalized transformation, cg_j, by first computing a range around cg_j using

$$Certainty(cg_j, Rj, cert) = \tau_{(df=|Rj|-1,cert)} \frac{\sqrt{\sum_{T_i \in Rj} \delta(T_i, cg_j)^2}}{|Rj|} \quad (1)$$

where $\tau_{(df=|Rj|-1,cert)}$ is the $t\text{-}score$ for $df = |Rj| - 1$ at the specified certainty level, Rj are the candidate transformations cg_j was generalized from, and $\delta(T_i, cg_j)$ is the distance between T_i and cg_j. We compute $\delta(T_i, cg_j)$ using

$$\delta(T_i, cg_j) = \sum_{c \in T_i} taxdist(c, corresp(c, cg_j)) \quad (2)$$

where c is a concept in T_i, $corresp(c, cg_j)$ is the corresponding concept of c in cg_j, and $taxdist$ is the taxonomic distance between the labels of these two concepts. We compute $taxdist$ as the minimum number of $is\text{-}a$ and $instance\text{-}of$ links (as defined by the ontology) between the labels of c and $corresp(c, cg_j)$. Next, we say cg_j satisfies the minimum certainty requirement if the *Null* generalization lies outside this range – i.e. $\delta(Null(cg_j), cg_j) > Certainty(cg_j, Rj, mincert)$. We compute $\delta(Null(cg_j), cg_j)$

using Equation 2 and the *Null* generalization $Null(cg_j)$ by replacing the labels of all the concepts in cg_j with $\top$ – the root concept in the ontology.

4 Evaluation

We evaluate our algorithm by using it to mine for transformations to use for two tasks – Course of Action (COA) critiquing and finding examples to illustrate chemistry encoding mistakes.

4.1 COA Critiquing

The COA critiquing task is one of the challenge problems in DARPA's Rapid Knowledge Formation project. COAs are large, detailed battle plans intended to meet a specific military objective. Because of their complexity, military analysts have difficulty evaluating them quickly and accurately. Thus, the task of COA critiquing is to assess a COA's strengths and weaknesses.

The solution to this problem has two parts. First, military analysts build a knowledge base of "critiquing patterns" using an ontology in the domain of Battle Space Planning (BSP). [3] Each pattern, encoded as a simple conceptual graph, describes a situation that might arise in a COA, for example "blue aviation units attack red artillery units before the main attack" or "blue holds an artillery unit in reserve during the main attack". This library of patterns is compiled based on military doctrine and experience. Second, to prepare for a battle, military commanders design a COA to achieve stated objectives. It, too, is a simple conceptual graph and is encoded using the BSP ontology. These COAs are evaluated by matching the library of critiquing patterns to them to assess their strengths and weaknesses.

The knowledge base of critiquing patterns was built by two Subject Matter Experts (SMEs) using the SHAKEN system [3]. The COAs were built by the same people using a COA-authoring system called NuSketch [7]. The two SMEs produced a total of 44 patterns and three COAs. The patterns and COAs were both encoded as graphs but authored using different knowledge-authoring tools. COA critiquing is therefore a graph matching problem with many opportunities for mismatches.

Experimental Methodology We used the knowledge base of critiquing patterns and COAs authored by the two SMEs as our data. We say a pattern matches a COA if the match score meets or exceeds a pre-specified threshold (*TH*). This score is computed based on the fraction of nodes in the pattern matched with a COA. Because of the size and complexity of the COAs (each averaging several thousand nodes), a pattern can match a COA multiple times, in different ways.

To match the patterns with the COAs, we used four approaches that differ only in the types of transformations used. First, we used a semantic matcher that uses only taxonomic knowledge as our baseline (i.e. no transformations were used). We call this matcher "TaxonMatch", and we implement it using the algorithm described in [16, 17].

[3] This ontology can be browsed and downloaded at http://www.cs.utexas.edu/users/mfkb/RKF/tree

Second, we constructed a matcher called "TaxonMatch+Mined" by augmenting TaxonMatch with the transformations mined by MinT. To mine for these transformations, we split our data into three sets where each set consists of one COA and a set of applicable patterns. We chose one set to serve as the test set, and the other two for mining transformations. MinT's inputs and parameters were set as follows. We set the semantic matcher to TaxonMatch and the two lists of graphs to the patterns and COAs in the training set. We define the similarity measure as the fraction of nodes in a pattern matched with a COA, and we arbitrarily set the minimum match score, support, and certainty to 0.6, 0.05, and 0.95 respectively. The syntactic forms used for this experiment are shown in Table 1.

	lhs	**rhs**
F_1	$X - r \rightarrow Y - r \rightarrow Z$	$X - r \rightarrow Z$
F_2	$X - r \rightarrow Y - s \rightarrow Z$	$X - r \rightarrow Z$
F_3	$X - r \rightarrow Y$	$X - s \rightarrow Y$
F_4	$X - r \rightarrow Y - s \rightarrow Z - t \rightarrow W$	$X - s \rightarrow W$

Table 1. The syntactic forms used for both this experiment and the chemistry experiment described in Section 4.2.

The resulting transformations (see Figure 3 for examples) were used by TaxonMatch+Mined to match the patterns with the COA in the test set. This process was repeated three times (each time a different training and test set were selected), and we totalled all the matches.

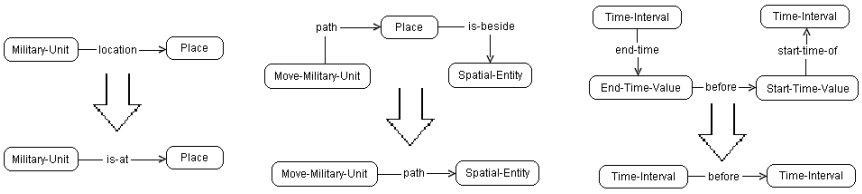

Fig. 3. Selected examples of transformations mined for the battle space planning domain.

Third, we constructed a matcher called "TaxonMatch+DN" by augmenting TaxonMatch with the human authored domain-neutral transformations reported in [16, 17].

Finally, we constructed a matcher called "TaxonMatch+Both" by augmenting TaxonMatch with both mined and human authored transformations.

To evaluate the matches returned by each approach, and hence their performance, we note that applying patterns to COAs can be viewed as a form of information retrieval where information of interest is retrieved from a COA based on a pattern that is acting as a query. Therefore, we use metrics of precision and recall. To calculate these metrics,

we compared the match results for each approach against those of a human "oracle", who determined that there were a total of 927 correct matches between the patterns and the COAs. Table 2 shows the number of correct matches over the total number of matches given by each approach along with the precision and recall results for match thresholds between 0.5 and 1.0. We used the $\mathcal{X}^2$ test to test for significant differences.

	TaxonMatch			**TaxonMatch+Mined**			**TaxonMatch+DN**			**TaxonMatch+Both**		
TH	raw	prec	recall	raw	prec	recall	raw	prec	recall	raw	prec	recall
0.5	508/1395	36.4	54.8	589/1535	38.4	63.5	763/1714	44.5	82.3	763/1732	44.1	82.3
0.6	508/1239	41.0	54.8	571/1431	39.9	61.6	745/1539	48.4	80.4	745/1680	44.4	80.4
0.7	459/686	66.9	49.5	571/840	68.0	61.6	696/965	72.1	75.1	745/1014	73.5	80.4
0.8	459/667	68.8	49.5	522/772	67.6	56.3	696/946	73.6	75.1	696/946	73.6	75.1
0.9	103/103	100.0	11.1	516/652	79.1	55.7	553/689	80.3	59.7	679/815	83.3	73.2
1.0	35/35	100.0	3.8	501/501	100.0	54.1	424/424	100.0	45.7	654/654	100.0	70.6

Table 2. This table shows the raw data (i.e. *raw*), precision (i.e. *prec*), and recall results given by each approach. Precision is computed by dividing the number of correct matches given by the total number of matches given. Recall is computed by dividing the number of correct matches given by the number of correct matches given by the "oracle".

Discussion With respect to precision (see Table 2), the four approaches were roughly comparable. The 0.9 threshold, however, was an interesting case. At this high threshold, TaxonMatch was able to match only those patterns that aligned exactly (or almost exactly) with the COAs. Thus, there were no false-positives and precision was high (this difference was significant at the 0.01 level for the $\mathcal{X}^2$ test, i.e. $p < 0.01$). Using transformations, the other three approaches were able to establish additional mappings that TaxonMatch was unable to find. This resulted in more patterns being matched, but there was still room for false-positives at the 0.9 threshold level. This difference disappeared at the 1.0 threshold level.

With respect to recall (see Table 2), the four approaches differed significantly. TaxonMatch's performance on recall was the worst, dropping significantly as the threshold level was raised. TaxonMatch+Mined performed significantly better than TaxonMatch at all threshold levels ($p < 0.01$ for all points). Because TaxonMatch and TaxonMatch+Mined differ only in the use of transformations, this feature alone must account for the observed difference in recall. Therefore, this difference shows transformations mined by MinT helped to improve matching significantly compared to using taxonomic knowledge alone.

TaxonMatch+Mined performed significantly better than TaxonMatch+DN at the 1.0 threshold ($p < 0.01$). This difference, however, was reversed between the 0.5 and 0.8 threshold levels ($p < 0.01$ at each point) because MinT was unable to discover some of the human authored transformations used by TaxonMatch+DN. Some useful candidate transformations were excluded because either they did not occur frequently enough or their common generalization failed to satisfy the minimum certainty requirement.

By using both human authored and machine discovered transformations, TaxonMatch+Both performed significantly better than both TaxonMatch+Mined ($p < 0.01$ for all threshold levels) and TaxonMatch+DN ($p < 0.01$ for the 0.7, 0.9, and 1.0 threshold levels). Interestingly, the difference between TaxonMatch+Both and TaxonMatch+DN was most noticeable at the 0.9 and 1.0 thresholds. At these high thresholds, only those patterns that aligned exactly (or almost exactly) with the COAs would count as matches. Because TaxonMatch+DN used only domain-neutral transformations, it was not able to resolve many domain-specific mismatches. As a result, TaxonMatch+DN could not align many patterns with the COAs. MinT, however, discovered transformations that resolved those mismatches that were idiosyncratic to the COA critiquing domain. These results show domain-neutral transformations are less effective at resolving domain-specific mismatches, and domain-specific transformations are needed to further improve matching.

4.2 Finding Examples to Illustrate Chemistry Encoding Mistakes

The goal of the HALO project is to build a Knowledge Base[4] (KB) that can answer chemistry questions from an Advanced Placement[5] (AP) test. In the initial phase of this project [4], Knowledge Engineers (KEs) encoded the questions to be answered. Because KEs are familiar with the KB, their encodings do not include any mistakes (i.e. discrepancies with the KB). The second phase of this project, however, requires SMEs to encode these questions. Because SMEs are not familiar with the KB, mistakes are frequent in their encodings (see Figure 4). Thus, we need to find examples of correct encodings to illustrate to SMEs their mistakes.

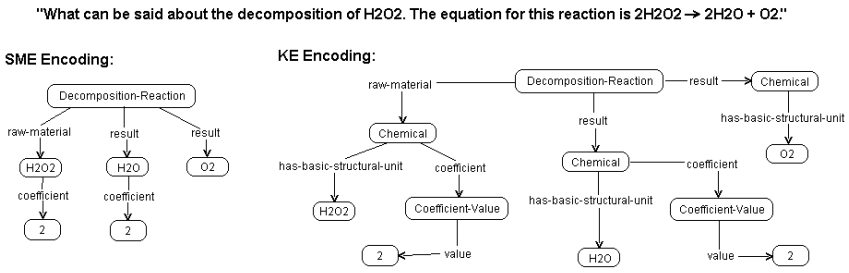

Fig. 4. This example shows two encodings of the same chemistry question – one by a SME and the other by a KE.

Our solution to this problem has two parts. First, KEs will encode a library of questions (we'll refer to the encodings of these questions as cases). Each case is a simple conceptual graph encoded using an ontology in the domain of chemistry. [6] Second,

[4] By knowledge base, we mean a set of axioms.

[5] The AP test is an examination taken by U.S. high school students to earn college credits.

[6] This ontology can be browsed and downloaded at http://www.cs.utexas.edu/users/mfkb/RKF/tree

questions encoded by SMEs (which are also simple conceptual graphs encoded using the same chemistry ontology) would be matched against this library to find the most similar case, which will serve as an example to point out mistakes made.

The library of cases consists of correct encodings for 50 questions. Three SMEs were each asked to encode these same 50 questions, and they encoded a total of 133 questions (SMEs were allowed to skip questions they did not know how to encode). Because SME questions and cases are encoded as graphs and done so independently, the task of finding examples to illustrate encoding mistakes is a graph matching problem with many opportunities for mismatches. Hence, transformations are needed.

Experimental Methodology We use the SME questions and cases as our data. Each SME question is matched against all cases, and the case it most strongly matches is selected as the example to use. The strength of the match (i.e. score) is based on the fraction of nodes in a SME question matched with a case.

To match the SME questions with the cases, we used the four approaches from the COA Critiquing task (see Section 4.1). With the exception of TaxonMatch+Mined, the other three approaches were constructed the same way.

To mine for transformations used by TaxonMatch+Mined, we split our data into three sets, where each set consists of all the questions encoded by one of the SMEs. We chose one set to serve as the test set, and the other two for mining transformations. MinT's inputs and parameters were set as follows. We set the semantic matcher to TaxonMatch. We set the two lists of graphs to the SME questions in the training set and the cases encoded by KEs. We define the similarity measure as the fraction of nodes in a SME question matched with a case, and we arbitrarily set the minimum match score, support, and certainty to 0.4, 0.05, and 0.95 respectively. We used the same syntactic forms from the COA Critiquing task (see Table 1). The resulting transformations (see Figure 5) were used by TaxonMatch+Mined to match the SME questions in the test set with the cases. This process was repeated three times (each time different training and test sets were selected), and we totalled all the matches.

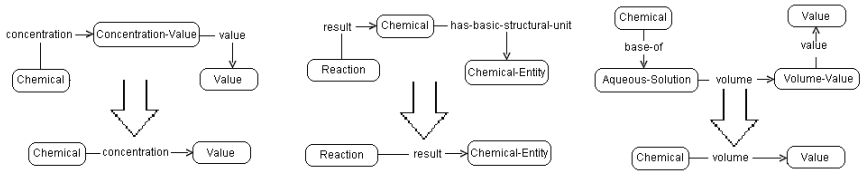

Fig. 5. Selected examples of transformations mined for the chemistry domain.

To evaluate the performance of each approach, we measure the number of SME questions for which a correct case was selected. We say the case selected for a SME question is correct, if they are both encodings of the same chemistry question. Table 3 shows the number of correct matches over the total number of matches given by each

approach along with the precision and recall results for match thresholds between 0.0 (exclusive) and 1.0 (inclusive). We used the $\mathcal{X}^2$ test to test for significant differences.

	TaxonMatch			**TaxonMatch+Mined**			**TaxonMatch+DN**			**TaxonMatch+Both**		
TH	raw	prec	recall	raw	prec	recall	raw	prec	recall	raw	prec	recall
0.0	128/1124	11.4	96.2	128/203	63.1	96.2	128/307	41.7	96.2	128/203	63.1	96.2
0.2	94/734	12.8	70.7	128/200	64.0	96.2	127/276	46.0	95.5	128/200	64.0	96.2
0.4	58/208	27.9	43.6	127/176	72.2	95.5	127/256	49.6	95.5	127/176	72.2	95.5
0.6	36/37	97.3	27.1	127/176	72.2	95.5	108/128	84.4	81.2	127/176	72.2	95.5
0.8	35/36	97.2	26.3	77/78	98.7	57.9	61/62	98.4	45.9	77/78	98.7	57.9
0.9	32/33	96.9	24.1	68/69	98.6	51.1	55/56	98.2	41.4	68/69	98.6	51.1
1.0	32/32	100.0	24.1	63/63	100.0	47.4	51/51	100.0	38.3	63/63	100.0	47.4

Table 3. This table shows the raw data (i.e. *raw*), precision (i.e. *prec*), and recall results given by each approach.

Discussion With respect to precision, the four approaches were comparable starting from the 0.8 threshold level. TaxonMatch+Mined, however, performed significantly better than TaxonMatch and TaxonMatch+DN from the 0.0 to 0.4 threshold levels ($p < 0.01$ at each point).

With respect to recall, TaxonMatch+Mined performed significantly better than TaxonMatch starting from the 0.2 threshold level ($p < 0.01$ at each point). Because TaxonMatch and TaxonMatch+Mined differ only in the use of transformations, this factor alone must account for the observed difference in accuracy. This difference further shows transformations mined by MinT can help to improve matching significantly compared to using taxonomic knowledge alone.

Interestingly, TaxonMatch+Mined performed significantly better than TaxonMatch+DN at the 0.6 ($p < 0.01$) and 0.8 threshold levels ($p < 0.05$). Although the domain-neutral transformations used by TaxonMatch+DN helped to improve recall significantly compared to TaxonMatch ($p < 0.01$ starting from the 0.2 threshold level), there were several mismatches specific to the domain of chemistry that TaxonMatch+DN could not resolve – more so than the battle space domain. MinT, however, was able to discover transformations that can resolve these mismatches. These results show domain-specific transformations can further improve matching.

Finally, TaxonMatch+Mined and TaxonMatch+Both achieved the same precision and recall (recall TaxonMatch+Both is constructed by augmenting the human authored domain-neutral transformations with those discovered by MinT). Our analysis revealed that in addition to discovering additional domain-specific transformations, MinT was able to discover all the domain-neutral transformations useful for this domain. This explains why TaxonMatch+Mined and TaxonMatch+Both had the same performance.

5 Conclusion

The problem in semantic matching is to find a mapping between two structured knowledge representations encoded using the same ontology. Solving this matching problem is hard because many valid mappings between two representations often cannot be established. Previous research has shown an imperfect match between two representations can be improved significantly by using transformation rules, but acquiring these transformations is problematic. As a result, most existing semantic matching approaches use only taxonomic knowledge.

To address this problem, we presented an algorithm, called MinT, for mining transformations. We evaluated MinT by using it to mine for transformations in two domains – Battle Space Planning (BSP) and chemistry. The results from both evaluations were encouraging. Transformations mined by MinT helped to improve matching significantly compared to using taxonomic knowledge alone. For the BSP domain, human authored transformations performed better than those mined by MinT, but we showed MinT's transformations can augment those authored by hand to outperform either one used alone. Interestingly, this was not the case for the chemistry domain – transformations mined by MinT outperformed those authored by hand. This is because most of the mismatches were specific to the domain of chemistry and hence domain-specific transformations are needed to resolve them.

Acknowledgements

Support for this research was provided by a contract from SRI international as part of DARPA's Rapid Knowledge Formation project.

References

1. R. Agrawal, T. Imielinski, and A. Swami. Mining association rules between sets of items in large databases. In *ACM SIGMOD*, 1993.
2. R. Agrawal and R. Srikant. Fast algorithms for mining association rules. In *VLDB*, 1994.
3. K. Barker, J. Blythe, G. Borchardt, V. Chaudhri, P. Clark, P. Cohen, J. Fitzgerald, K. Forbus, Y. Gil, B. Katz, J. Kim, G. King, S. Mishra, C. Morrison, K. Murray, C. Otstott, B. Porter, R. Schrag, T. Uribe, J. Usher, and P. Yeh. A knowledge acquisition tool for course of action analysis. In *IAAI*, 2003.
4. K. Barker, S. Chaw, J. Fan, B. Porter, D. Tecuci, P. Yeh, V. Chaudhri, D. Israel, S. Mishra, P. Romero, and P. Clark. A question-answering system for ap chemistry: Assessing kr&r technologies. In *KR*, 2004.
5. H. Bunke and K. Shearer. A graph distance metric based on the maximal common subgraph. *Pattern Recognition Letters*, 19, 1998.
6. M. Chein and M.L. Mugnier. Conceptual graphs: Fundamental notions. *Revue d'intelligence artificielle*, 6(4):365–406, 1992.
7. K. Forbus and J. Usher. Sketching for knowledge capture: A progress report. In *IUI*, 2002.
8. D. Genest and M. Chein. An experiment in document retrieval using conceptual graphs. In *ICCS*, 1997.
9. M. Gomez, A. Gelbukh, and A. Lopez. Discovering association rules in semi-structured data sets. In *IJCAI, Workshop on knowledge discovery*, 2001.

10. M. Gomez, A. Gelbukh, and A. Lopez. Text mining at detail level using conceptual graphs. In *ICCS*, 2002.
11. N. Guarino, C. Masolo, and G. Vetere. Ontoseek: Content-based access to the web. *IEEE Intelligent Systems*, 14(3):70–80, 1999.
12. A. Inokuchi, T. Washio, and H. Motoda. An apriori-based algorithm for mining frequent substructures from graph data. In *PKDD*, 2000.
13. P. Mulhem, W.K. Leow, and Y.K. Lee. Fuzzy conceptual graphs for matching images of natural scenes. In *IJCAI*, 2001.
14. U.Y. Nahm and R. Mooney. Mining soft-matching rules from textual data. In *IJCAI*, 2001.
15. J.F. Sowa. *Conceptual Structures: Information Processing in Mind and Machine*. Addison-Wesley Publishing Company, 1984.
16. P. Yeh, B. Porter, and K. Barker. Transformation rules for knowledge-based pattern matching. Technical Report UT-AI-TR-03-299, U.T. Austin, 2003.
17. P. Yeh, B. Porter, and K. Barker. Using transformations to improve semantic matching. In *KCAP*, 2003.
18. J. Zhong, H. Zhu, J. Li, and Y. Yu. Conceptual graph matching for semantic search. In *ICCS*, 2002.