Matching Utterances to Rich Knowledge Structures to Acquire a Model of the Speaker's Goal

Peter Z. Yeh pzyeh@cs.utexas.edu
Bruce Porter porter@cs.utexas.edu
Ken Barker kbarker@cs.utexas.edu

Department of Computer Sciences
University of Texas, Austin
Austin, TX 78712 USA

ABSTRACT

An ultimate goal of AI is to build end-to-end systems that interpret natural language, reason over the resulting logical forms, and perform actions based on that reasoning. This requires systems from separate fields be brought together, but often this exposes representational gaps between them. The logical forms from a language interpreter may mirror the surface forms of utterances too closely to be usable as-is, given a reasoner's requirements for knowledge representations. What is needed is a system that can match logical forms to background knowledge flexibly to acquire a rich semantic model of the speaker's goal. In this paper, we present such a "matcher" that uses semantic transformations to overcome structural differences between the two representations. We evaluate this matcher in a MUC-like template-filling task and compare its performance to that of two similar systems.

Categories and Subject Descriptors

I.2.4 [**Artificial Intelligence**]: Knowledge Representation Formalisms and Methods

General Terms

Algorithms

Keywords

semantic matching, ontology, knowledge-based systems, discourse, natural language understanding

1. INTRODUCTION

Calo is a system being developed as an intelligent personal office assistant. It is intended to be a platform for the integration of system components from various fields of AI such as: speech understanding, dialog interpretation, knowledge representation and reasoning, machine learning, planning and action, physical awareness, and cyber awareness. One of the evaluation tasks of *Calo* is to engage its user in a speech-based dialog to elicit the user's desire for *Calo* to assist in performing an action. An example is for *Calo* to help its user buy a computer. This problem requires, among many other things, a tight integration of a dialog understanding system with a knowledge-based reasoner. The following steps illustrate where *Calo's* subsystems need to collaborate to achieve this goal. This particular list of steps is for dialogs expressing a user's desire for *Calo* to perform an action (as opposed to a user stating a belief or intention).

1. The user makes an utterance
2. The dialog understanding system produces a logical form of the utterance
3. The dialog system engages the knowledge-based reasoner to help interpret the utterance as a desire for system action
4. The KB reasoner establishes a match between the logical form and existing structures in the KB, establishing a model of the utterance as expressing a new desire
5. Interpretations of subsequent utterances (within the same topic) are given to the KB reasoner to be related to the current model of the user's desire (so as to elaborate this model)
6. If there is a shift in topic or user pause, the dialog system takes initiative to elicit from the user any extra information (as identified by the KB reasoner's knowledge of the model) required to perform the desired action
7. Once the dialog manager and KB reasoner determine that the model of the user's desire is complete, a request is passed to a task manager to perform the desired action

K-CAP'05, October 2–5, 2005, Banff, Alberta, Canada.

In this paper, we focus on the issue of matching logical forms to existing representations in the KB – required in steps 4 and 5 – to acquire a model of the user's desire. This match process is necessary because the logical form produced by a natural language interpreter is often underspecified from the point of view of a reasoner attempting to reason over it. This underspecification occurs because:

- implicit knowledge in an utterance (needed for reasoning) is often missing from the logical form.
- the user's desire is split across multiple utterances, so the links (both direct and indirect) between these utterances are missing from the logical forms.

These problems are not due to weaknesses of natural language interpreters because often the logical form produced by these systems captures all of the information in an utterance. Instead, these problems are due to the conciseness of human communication, which assumes common background knowledge between speaker and listener. For example, a user may say:

> *I want to buy a new laptop. I need 512 MB of memory.*

The dialog system produces a logical form for the first utterance, but this logical form only captures semantics at the level of case roles, etc. To determine whether this utterance is a statement of a new user belief or desire, the dialog system must consult the KB reasoner. The KB reasoner, in this case, can match the logical form to the goal model of *purchase* in the KB, supporting the dialog system's preferred interpretation of the utterance as a desire for action (i.e. to purchase a computer).

For the second utterance, the logical form only captures the need for memory. It does not capture the link between this statement and the previous one – i.e. the memory is part of the new laptop. The KB reasoner can establish this link by matching this logical form to structures in the KB relevant to the established model of *purchase*. This supports the dialog system's preferred interpretation of elaborating the established model (as opposed to signalling a shift in topic).

There has been significant previous work on matching logical forms to an existing KB [1, 2, 6, 9, 8, 12, 13] to establish links between utterances and detect additional knowledge that does not appear in the discourse. These approaches can establish the indirect link between the first and second utterances above by matching their logical forms with knowledge that laptops have memory as a part.

Previous systems, however, require an exact match between logical forms and the KB (i.e. an ontology). This creates a matching problem when logical forms produced by a language interpreter differ structurally from representations in the KB, as is often the case due to the conciseness of natural language and the close mirroring of a logical form to the surface form of its utterance.

In this paper, we describe a flexible matcher for matching logical forms to representations in a rich KB to build a model of the user's goal. Our matcher achieves this flexibility by using semantic transformation rules to overcome structural differences between two representations. Our matcher has been used successfully in the greater *Calo* architecture, tightly coupled to the system's automatic dialog interpreter. To focus evaluation on our matcher, we present an experimental setup in which its performance is isolated from the greater system by providing it with a "perfect" (hand-built) corpus of logical forms. We also compare this performance to that of two similar systems: one from the MUC template-filling task and a matcher that does not use semantic transformations.

2. SEMANTIC MATCHING

A semantic matcher takes two representations and returns the best match between them based on knowledge about the concepts and relations referenced in the representations. Most semantic matchers [7, 10, 11, 15, 20] use only taxonomic knowledge – i.e. the subsumption relationships among concepts and relations. The top half of Figure 1 shows the result of matching two representations (here drawn as conceptual graphs [16]) by using the knowledge that *Person* subsumes **Bob* and *Get* subsumes *Buy*. Unfortunately, the other parts of the representations are left unmatched.

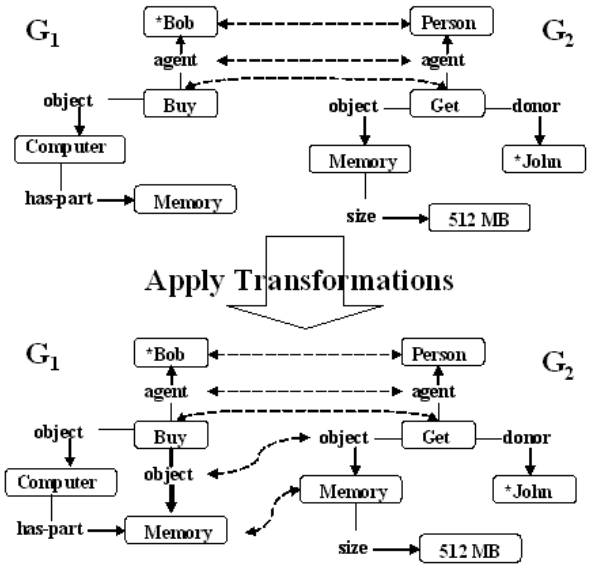

Figure 1: Top: Two conceptual graphs G_1 and G_2. The dashed lines show those nodes that can be matched using taxonomic knowledge. Bottom: A part descension transformation rule augments G_1 by adding an $object$ relation between Buy and $Memory$. This allows additional nodes to be matched.

Our semantic matcher is designed to address this shortcoming. It first uses taxonomic knowledge to find the largest connected subgraph in one representation that matches a subgraph in the other (it ignores degenerate – i.e. single node – subgraphs). Our matcher then uses a library of transformation rules to shift the representations to improve the match. This improvement might enable other (non-degenerate) subgraphs to match taxonomically, which in turn might enable more transformation rules, and so on until the match improves no further.

The bottom half of Figure 1 shows the result of applying a transformation rule to G_1. This rule augments the representation by adding an *object* relation between *Buy* and *Memory*, thereby enabling the matcher to align the *buy of memory* in G_1 with *the get of memory* in G_2. The intuition behind this rule is that the purchase of a computer includes the purchase of its memory. This, of course, is a very domain specific rule and achieving broad coverage with such rules is infeasible.

In contrast, our library of transformation rules is based on a domain-independent upper ontology [3] consisting of about 500 generic entities, events, and roles, and about 75 relations – each with a well defined semantics. We generated this library of rules in a systematic manner. First, we identified a recurring pattern of rules that are useful for semantic matching called "transfers through" [14]. The general form of this pattern is:

$$C_1 \xrightarrow{r_1} C_2 \xrightarrow{r_2} C_3 \Rightarrow C_1 \xrightarrow{r_1} C_3$$

where C_i is a concept and r_j is a relation. Then, we exhaustively enumerated all semantically valid instantiations of this pattern using concepts and relations from our upper ontology. The result was a comprehensive library of about 200 rules such as:

$$\begin{aligned} &Event_1 \xrightarrow{object} Entity_2 \xrightarrow{haspart} Entity_3 \\ &\Longrightarrow Event_1 \xrightarrow{object} Entity_3 \end{aligned} \quad (1)$$

$$\begin{aligned} &Event_1 \xrightarrow{causes} Event_2 \xrightarrow{causes} Event_3 \\ &\Longrightarrow Event_1 \xrightarrow{causes} Event_3 \end{aligned} \quad (2)$$

The first rule encodes *part descension* (i.e. acting on a whole also acts on its parts), and is the general form of the transformation rule used in Figure 1. The second rule encodes the transitivity of causality.

This library of transformations has been used previously to significantly improve matching in the domains of battle space planning [18] and chemistry [19]. For a complete list of these rules, we refer the reader to [17].

In our semantic matcher, a rule is applicable to a representation if its antecedent subsumes a subgraph of the representation (i.e. it either matches exactly or matches using taxonomic knowledge). Our matcher applies a transformation rule by *joining* [16] the rule's consequent with the subgraph that matched the rule's antecedent. See [18] for a complete description of our semantic matcher.

3. USING SEMANTIC MATCHING TO BUILD A MODEL OF THE USER'S GOAL

Calo's dialog system tracks topic shifts in the dialog and builds a semantic model of the user's goal. Its KB reasoner supports this task with two functions that use the semantic matcher described in Section 2. *EstablishModel* is invoked when a new topic is introduced in the dialog. *RelateUtterance* is invoked to augment an existing goal model with the logical form of a subsequent utterance. The logical forms and model of the user's goal are conceptual graphs.

3.1 Establish Model

$EstablishModel$ (see Figure 2) builds an initial model of the user's goal by matching the logical form of an utterance introducing a new topic with representations in an ontology. We assume this ontology encodes the following information for concepts and relations – 1) their name Nm (which is also their type), 2) their immediate superclasses Sp, 3) their immediate subclasses Sb, and 4) a representation R (which is a conceptual graph) of each concept (see middle of Figure 3 for a fragment of this ontology rooted at $Event$).

GIVEN: a logical form L and an ontology O.
RETURN: An initial model of the user's goal.
LET $Max = 0$ and $M = NIL$
FOR each $\langle Nm, Sp, Sb, R\rangle$ in O where
  Nm is a concept DO
  IF $Nm \in_{Sb} concepts(L)$ THEN
    LET $match = SemMatch(L, R, Transforms, O)$
    LET $score = \frac{TSum(match)}{|L|}$
    IF $score > Max$ THEN
      LET $Max = score$ and $M = R$
RETURN $Join(L, M)$

Figure 2: The algorithm for EstablishModel. Builds an initial model by matching a logical form introducing a new topic with an ontology.

To reduce the search space, $EstablishModel$ considers only those concepts that are either referenced in the logical form or have an immediate subclass that is referenced in the logical form. It then computes a score for each match based on the sum of the taxonomic distance between the matched nodes over the total number of nodes in the logical form:

$$\frac{TSum(match)}{|L|}$$

where $|L|$ is the number of nodes in the logical form and $TSum(match)$ is defined as:

$$TSum(match) = \sum_{(n_i,n_j)\in match} \frac{1}{taxdist(n_i, n_j) + 1}$$

where $taxdist$ is the minimum number of steps between two concepts (or relations) in our ontology. The best match is *joined* [16] – i.e. merged – with the logical form to obtain an initial model of the user's goal.

We illustrate this function with the following utterances, where the speaker is Bob:

> *I want to buy a computer with monitor. I want to get a large monitor. Order it today.*

To establish an initial model of Bob's goal, $EstablishModel$ matches the logical form of the first utterance (we'll call L_1) with an ontology (see Figure 3).

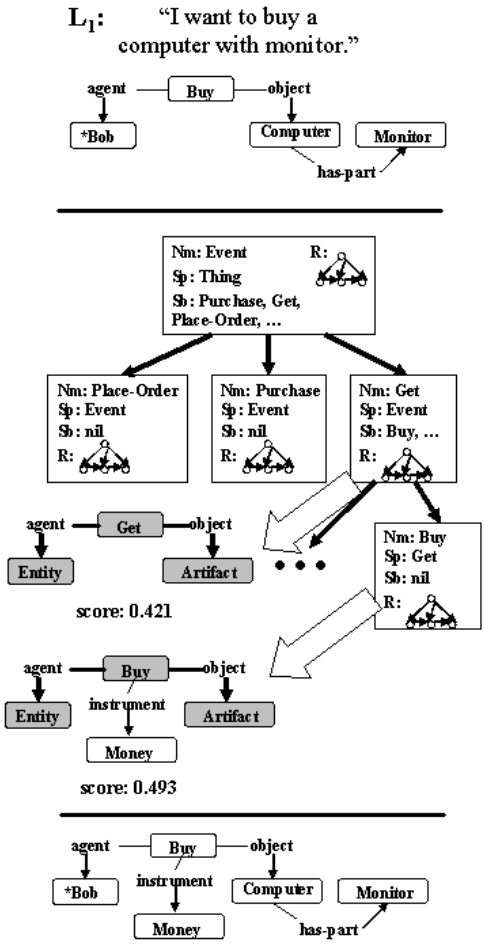

Figure 3: Top: The logical form of the first utterance. Middle: A fragment of the ontology rooted at $Event$. This ontology also shows the representations of Buy and Get, both of which match L_1. Subgraphs that match L_1 are shown in bold along with the match score. Bottom: An initial model of Bob's goal obtained by joining L_1 with Buy.

To reduce the search space, *Purchase* and *Place-Order* are not considered because neither they nor any of their immediate subclasses are referenced in L_1. This leaves *Get* and *Buy* which are matched with L_1. *Buy* matches L_1 with a score of 0.493 because $|L_1|$ is 7, and in our ontology, $taxdist(Artifact, Computer)$ = 3, $taxdist(Entity, Bob)$ = 4, and $taxdist$ is 0 for the other 3 matched nodes. *Get* matches L_1 also, but its match score is 0.421. This is because $taxdist(Get, Buy)$ = 1 and the other matched nodes have the same taxonomic distance as before. Since *Buy* is the better match, it is joined with L_1 to obtain an initial model of Bob's goal to buy a computer (as shown in the bottom of Figure 3). This match also detects additional information not in the utterance – i.e. *Money* is the instrument of a *Buy*.

3.2 Relate Utterance

$RelateUtterance$ (see Figure 4) elaborates the current goal model by matching it with the logical form of a subsequent utterance. If a match does not exist, then there is an indirect link. To establish this link, $RelateUtterance$ elaborates the current model with a new concept, drawn from the ontology, that also has features in common with the logical form. We call this new concept the bridging concept, and it is selected based on how well it matches both the current model and the logical form:

$$\frac{TSum(match1) \times TSum(match2)}{|R|}$$

where $|R|$ is the number of nodes in the representation of the bridging concept, $match1$ is the set of matched nodes between the logical form and R, and $match2$ is the set of matched nodes between R and the current model.

GIVEN: a logical form L, the current model of the user's goal M and an ontology O.
RETURN: An elaborated model of the user's goal.
LET $match = SemMatch(L, M, Transforms, O)$
IF $|match| > 0$ THEN
  RETURN $Join(L, M)$
ELSE
  LET $Max = 0$ and $Bridge = NIL$
  FOR each $\langle Nm, Sp, Sb, R\rangle$ in O where
    Nm is a concept DO
    LET $match1 = SemMatch(L, R, Transforms, O)$
    LET $match2 = SemMatch(R, M, Transforms, O)$
    LET $score = \frac{TSum(match1) \times TSum(match2)}{|R|}$
    IF $score > Max$ THEN
      LET $Max = score$ and $Bridge = R$
  RETURN $Join(L, Join(Bridge, M))$

Figure 4: The algorithm for RelateUtterance. Elaborates the current model by matching it with the logical form of a subsequent utterance.

Selecting the bridging concept can become intractable for large ontologies, but several methods can be used to reduce the search space. One method is to only consider a representation as a possible bridge if it includes concepts also referenced in both the logical form and the current model. Another method is to set a minimum match threshold for the bridging concept, and terminate the search when a representation meeting this threshold is found. Both methods work well in practice. We use the first one in our implementation.

We continue our example from section 3.1. Figure 5 shows the logical form of the second utterance (we'll call L_2) and the current model of Bob's goal to purchase a computer (we'll call M). $RelateUtterance$ matches L_2 with M, and there is a match – *Get* in L_2 matches *Buy* in M and the agent of *Get* (i.e. *Entity*) matches the agent of *Buy* (i.e. **Bob*). Since there are unmatched nodes, transformations are used to improve this match. A part descension transformation can be applied to M to add an object relation connecting *Buy* to *Monitor*. This allows the *Monitor* in L_2 to be matched with the *Monitor* in M. These are the only nodes that can be matched using transformations, so M is elaborated to reflect that: *the large monitor that Bob wants to get is part of the computer he wants to buy* (see bottom of Figure 5).

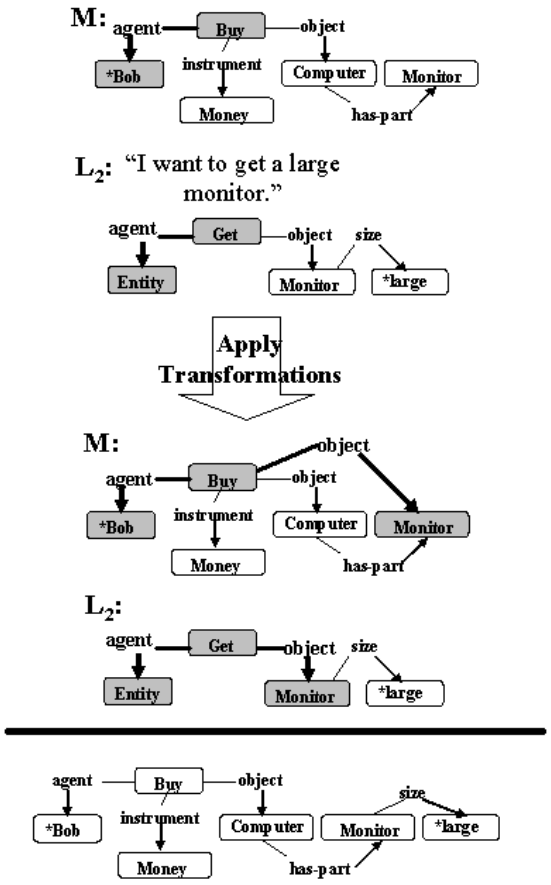

Figure 5: Top: The logical form of the second utterance (i.e. L_2), and the current model of Bob's goal to buy a computer (i.e. M). M is transformed to better match L_2 (matches are in bold). Bottom: The result of elaborating M with L_2.

The logical form of the last utterance (we'll call L_3) does not match the model resulting from the previous elaboration (also called M), so $RelateUtterance$ must establish the indirect link between them using a new concept drawn from the ontology. It cannot use *Buy* or *Get* (from Figure 3) because their representations do not match L_3. The representation of *Purchase*, however, matches both L_3 and M (see top of Figure 6), so it is selected as the bridging concept. This match establishes: 1) the indirect link between *Buy* and *Place-Order* (i.e. *Buy* is a *next-event* of *Place-Order*), and 2) the *Computer* is the object to be ordered today. More importantly, a coherent model of Bob's goal to purchase a computer with a large monitor is built through this matching process (see bottom of Figure 6).

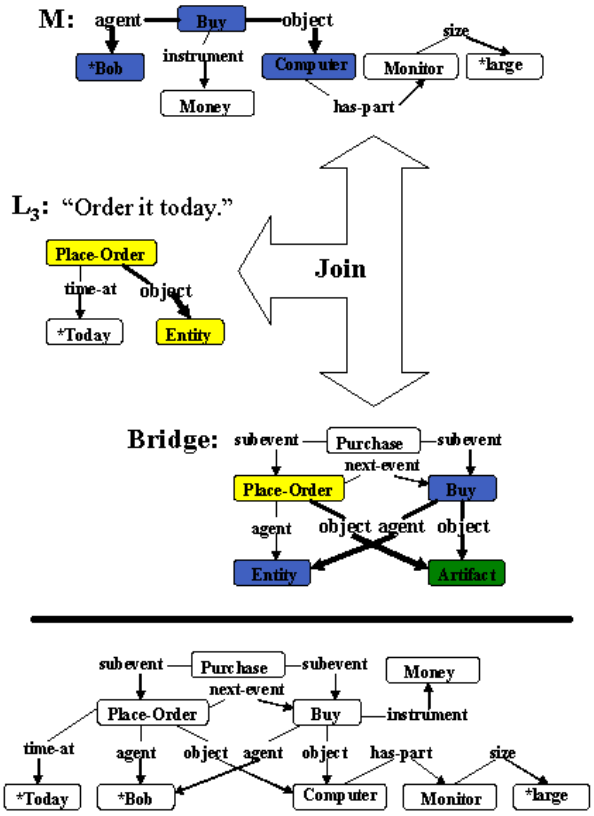

Figure 6: Top: The logical form of the third utterance L_3, the current model of Bob's goal M, and the bridging concept $Purchase$. Matches between L_3, M, and the bridging concept are shown in Bold. Bottom: The result of using $Purchase$ to establish the indirect link between L_3 and M.

4. EVALUATION

We evaluate our system on a MUC-like template-filling task using a corpus of purchase requests, and we compare its performance to that of two similar systems.

4.1 Experimental Setup

In order to evaluate the performance of our system (and those being compared), we need to isolate it from the other components of the greater *Calo* system. Hence, we constructed the following corpus with this goal in mind.

We selected, as our corpus, a log of Purchase REQuisitions (PREQs) kept by the department of Computer Sciences at the University of Texas at Austin. Each PREQ is an e-mail message from faculty or staff specifying items they want to purchase such as books, supplies, computers, etc. We chose this corpus because each e-mail is a discourse on a single topic, so the topic tracking component of the *Calo* system is not involved.

This corpus contains over 13,000 PREQs. From this large set, we selected only those PREQs concerning the purchase of computer equipment during the past year. This gave us 193 PREQs, and we randomly selected 75 for this evaluation. These 75 PREQs contain a total of 468 utterances and 4252 words. The average length of each PREQ is 6.24 utterances, and the average length of each utterance is 9.09 words.

To further isolate our system from the greater *Calo* system, we provide it with logical forms of utterances which were hand-built by three human subjects using an ontology in domain of office space (which we'll call the "office ontology"). This office ontology was built by extending the same domain-independent upper ontology [3] from which our transformations were derived. The three subjects, however, were unaware of this library of transformations, the existing structures in the office ontology, or the intended use of these logical forms. They knew only of the vocabulary in the office ontology, so this encoding task was a literal translation from English to logical forms. This setup mirrors the behavior of *Calo's* dialog system and how it interacts with our matcher.

We then asked an independent third party to review the fidelity of these encodings to ensure they mirror exactly the surface forms of their utterances. Those encodings with fidelity problems were either corrected by this third party or discarded.

4.2 Experimental Methodology

We compare our system to two similar systems on the same task. We chose the Discourse Interpreter (DI) in LaSIE-II [13] because it performed well in the MUC-7 competition. This system adds the concepts in each logical form to the existing discourse model. It then "expands" each concept to bring in additional information from the ontology and merges those with high similarity, based on the number of common attributes (i.e. shared nodes) and the taxonomic distance between the concepts being merged. We chose Overlay [1, 2] because it is just like our approach except: 1) it does not use transformations (it uses only taxonomic knowledge), and 2) it uses a different function to compute the match score.

The three systems used the same office ontology as their sole source of background knowledge, and all three systems were evaluated on the PREQ data set.

We model our evaluation after the MUC template-filling task [4, 5]. For each PREQ, we had the human subject answer as many questions as he could based on his background knowledge and understanding of the PREQ. This was done using an interactive program which begins with generic questions about purchase – e.g. "what is the object of the purchase?", "what is the objective of the purchase?", etc. Answers are given as concepts (e.g. $Laptop\text{-}Computer$), and multiple answers are allowed for each question. The program then asks follow-on questions for each answer. For example, if the subject entered $Laptop\text{-}Computer$ as the object of the purchase, then the program asks: "what is the cost of the Laptop-Computer?", "what are the parts of the Laptop-Computer?", etc. This interaction continues until no more answers are given. We use these answers as our *Gold Standard* (GS).

We then measured how well each system performed on this same task. First, the systems built a model of each PREQ from its logical forms. Then these models were used to answer questions about the purchase. Each question was answered by using its internal encoding to extract the answers from the models. Each question was encoded as a 3-tuple – i.e. $\langle Ref, rel, Filler \rangle$ where Ref is the concept in question, rel is the relation the question maps to, and $Filler$ is the answer. For example, "what is the object of the purchase?" is encoded as $\langle Purchase, object, ? \rangle$, and this is used to extract concepts related to $Purchase$ via an $object$ relation.

As our baseline, we used each question's internal encoding to extract the answers directly from the logical forms of each PREQ.

Each system's answers along with the baseline answers were compared with the gold standard and scored using the following MUC metrics [4]. A system's answer was *correct* if the GS gave the same answer, *partial*ly correct if the GS gave an answer that is a super/subclass of it (e.g. the GS said $Laptop\text{-}Computer$ and the system said $Computer$), and *incorrect* if the GS gave a different answer. If the system answered a question that the GS left blank, then this answer was counted as *spurious*. Questions answered by the GS but left blank by the system were counted as *missing*.

These metrics were then used to calculate Precision, Recall, and Overgeneration [4]:

$$\begin{aligned} Precision &= \frac{\#Correct + (\#Partial \times 0.5)}{SysAnswers} \\ Recall &= \frac{\#Correct + (\#Partial \times 0.5)}{GSAnswers} \\ Overgeneration &= \frac{\#Spurious}{SysAnswers} \end{aligned} \quad (3)$$

where *GS Answers* and *Sys Answers* are the total number of answers given by the gold standard and system respectively.

4.3 Results and Discussion

Table 1 shows each system's performance along with the baseline on the MUC metrics, and Figure 7 shows their performance on precision, recall, and overgeneration. Our approach outperformed the baseline, LaSIE-II's DI, and Overlay on precision, recall, and overgeneration. This difference was significant at the 0.01 level for the 2-tailed t-test in each case ($df = 148$ in each case).

Table 1: How each system performed on the MUC metrics.

Criteria	Our Approach	Overlay	LaSIE II DI	Baseline
Correct	2421	1453	1330	941
Partial	102	329	99	104
Incorrect	65	292	300	116
Spurious	675	894	2686	586
Missing	569	1083	1428	1996
GS Answers	3157	3157	3157	3157
Sys Answers	3263	2968	4415	1747

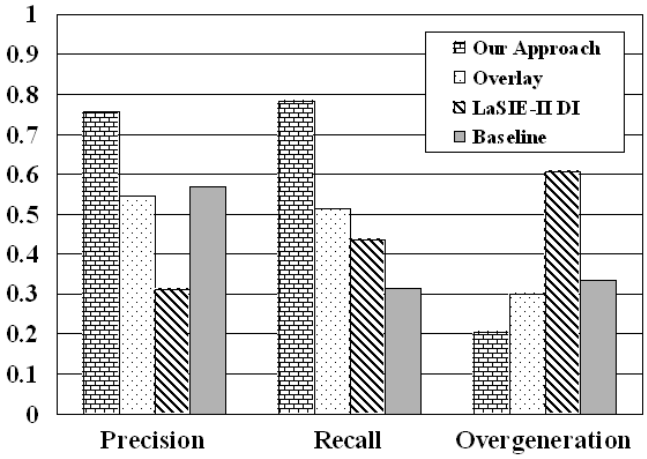

Figure 7: Each system's performance on precision, recall, and overgeneration averaged over all 75 PREQs.

Our system performed significantly better than the baseline on precision, recall, and overgeneration. As expected, the baseline had the worst recall because it does not attempt to recover any missing information. As a result, only questions whose answers were explicit in the logical form of an utterance could be answered. This shows that logical forms (even "perfect" hand-built ones) are insufficient by themselves.

Our system outperformed LaSIE-II's DI across all dimensions. We found, to our surprise, that the poor performance of LaSIE-II's DI was due to the size of the ontology used in this evaluation – the office ontology contains over a thousand concepts (and relations) and several thousand axioms. The ontology used by Gaizauskas et al. in the MUC-6 competition, for example, was significantly smaller – it contained 79 concepts and 55 relations [9]. Because of the size of the office ontology, different concepts often have many attributes in common. This causes a serious problem for LaSIE-II's DI when it tries to merge new concepts (introduced by an utterance) with previous ones based on how many attributes they share. LaSIE-II's DI eagerly expands new concepts, so much irrelevant information is brought in from the ontology causing many concepts to be merged incorrectly.

Our approach is not affected by this problem for two reasons. First, our approach brings in additional knowledge from the ontology only when there is no match between the logical form of an utterance and an existing model of the user's goal. Second, our similarity measure is based on information in the utterances themselves, so information not relevant to the discourse will not bias the scoring.

Our system also performed significantly better than Overlay on precision, recall, and overgeneration. We attribute this improvement to the only significant difference between our system and Overlay: our system uses transformation rules while Overlay uses only taxonomic knowledge. On average, our approach applied 3.9 transformations per PREQ.

These results are encouraging, and they show that flexible matching is essential for relating logical forms to the KB to build a model of the user's goal. This is because logical forms and the KB can encode similar content but differ substantially in form. We achieve flexible matching using transformation rules, but more importantly, useful transformations do not have to be tailored for the domain at hand. Instead, they can be derived from a domain-independent upper ontology and still be effective at overcoming structural differences between two representations.

5. CONCLUSION

In this paper, we presented a matcher to bridge the representational gap between the logical forms generated by a language interpreter and semantically related content in a rich knowledge base. We argued that a semantic matcher with transformations is needed because logical forms and representations in the KB can differ structurally, even though their intended content is similar.

We evaluated our system on a MUC-like template-filling task using e-mail messages about computer equipment purchases, and compared our approach to two similar ones – LaSIE-II DI and Overlay. We showed that our approach performed significantly better than both on precision, recall, and overgeneration. We also showed how using the logical forms directly is insufficient. We also learned that useful transformations do not have to be ad hoc or domain-specific. They can be derived from a domain-independent upper ontology and still be effective at resolving mismatches. Our contribution is showing how a semantic matcher with a library of domain-independent transforms can bridge the representational gap between a language interpreter and a KR&R system.

Our approach can also be applied to other domains with minimal effort because it is based on a domain-independent upper ontology. All that needs to be done is to extend the upper ontology to include concepts and representations specific to the domain of interest. This was what we did to apply our approach to all three domains in which it has been used.

Several issues still need to be explored. First, we plan to examine whether domain-specific transformations can augment domain-independent ones to further improve matching. Next, we plan to study whether this matching technique can be used for other tasks such as topic identification. Finally, we plan to study whether the output of this match process can provide context to the language interpreter to improve interpretation.

6. ACKNOWLEDGMENTS

Support for this research was provided by SRI International as part of DARPA's PAL project. The authors would like to thank Geoffery King, Jason Chaw, James Fan, and Dan Tecuci for their help on this project.

7. REFERENCES

[1] J. Alexandersson and T. Becker. Overlay as the basic operation for discourse processing in a multimodal dialogue system. In *IJCAI Workshop on Knowledge and Reasoning in Practical Dialogue Systems*, 2001.

[2] J. Alexandersson and T. Becker. The formal foundations underlying overlay. In *IWCS-5*, 2003.

[3] K. Barker, B. Porter, and P. Clark. A library of generic concepts for composing knowledge bases. In *KCAP*, 2001.

[4] N. Chinchor. MUC-3 evaluation metrics. In *MUC-3*, 1991.

[5] N. Chinchor. Overview of MUC-7/MET-2. In *MUC-7*, 1998.

[6] P. Cimiano. Ontology-driven discourse analysis in GenIE. In *NLDB*, 2003.

[7] N. Foo, B. Garner, A. Rao, and E. Tsui. Semantic distance in conceptual graphs. In P. Eklund, T. Nagle, J. Nagle, L. Gerhotz, and E. Horwood, editors, *Current Directions in Conceptual Structure Research*, 1992.

[8] R. Gaizauskas and K. Humphreys. Quantitative evaluation of coreference algorithms in an information extraction system. In S. Botley and T. McEnery, editors, *Corpus-based and Computational Approaches to Discourse Anaphora*, 1996.

[9] R. Gaizauskas, T. Wakao, K. Humphreys, H. Cunningham, and Y. Wilks. University of Sheffield: Description of the LaSIE system as used for MUC-6. In *MUC-6*, 1995.

[10] D. Genest and M. Chein. An experiment in document retrieval using conceptual graphs. In *ICCS*, 1997.

[11] N. Guarino, C. Masolo, and G. Vetere. Ontoseek: Content-based access to the web. *IEEE Intelligent Systems*, 14(3), 1999.

[12] J. Hobbs, M. Stickel, P. Martin, and D. Edwards. Interpretation as abduction. In *ACL*, 1988.

[13] K. Humphreys, R. Gaizauskas, S. Azzam, C. Huyck, B. Mitchell, H. Cunningham, and Y. Wilks. University of Sheffield: Description of the LaSIE-II system as used for MUC-7. In *MUC-7*, 1998.

[14] D. B. Lenat and R. Guha. *Building Large Knowledge-Based Systems.* Addison-Wesley, 1990.

[15] P. Mulhem, W. Leow, and Y. Lee. Fuzzy conceptual graphs for matching images of natural scenes. In *IJCAI*, 2001.

[16] J. F. Sowa. *Conceptual Structures: Information Processing in Mind and Machine*. Addison-Wesley, 1984.

[17] 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.

[18] P. Yeh, B. Porter, and K. Barker. Using transformations to improve semantic matching. In *KCAP*, 2003.

[19] P. Yeh, B. Porter, and K. Barker. Mining transformation rules for semantic matching. In *ECML/PKDD 2nd International Workshop on Mining Graphs, Trees, and Sequences*, 2004.

[20] J. Zhong, H. Zhu, J. Li, and Y. Yu. Conceptual graph matching for semantic search. In *ICCS*, 2002.