SENSEVAL-3: Third International Workshop on the Evaluation of Systems
for the Semantic Analysis of Text, Barcelona, Spain, July 2004
Association for Computational Linguistics

The MITRE Logical Form Generation System

Samuel Bayer and **John Burger** and **Warren Greiff** and **Ben Wellner**
The MITRE Corporation
Bedford, MA 01730
{sam,john,greiff,wellner}@mitre.org

Abstract

In this paper, we describe MITRE's contribution to the logical form generation track of Senseval-3. We begin with a description of the context of MITRE's work, followed by a description of the MITRE system and its results. We conclude with a commentary on the form and structure of this evaluation track.

1 Introduction

The logic form identification track of the 2004 Senseval evaluation requires its participants to produce a version of each input sentence with each input word in citation form, annotated with both a scope-free Davidsonian logic and lexical category information for major categories. The output ignores elements like determiners and negation, and features such as plurals and verb tenses.

This evaluation is of interest to the MITRE Corporation because it has a long-standing interest in text processing and understanding, in all its various dimensions. In our current internally funded Reading Comprehension (RC) project, we focus on the detailed understanding of individual stories, using the ability to answer comprehension questions associated with these stories as our evaluation metric. At the moment, we are interested in getting a sense of how much inference is routinely needed in order to answer RC questions; so generation of sentence meanings is not currently our research focus. However, in the context of our exploration, we continue to maintain an automated system for producing sentence meanings from text.

2 The MITRE logic generation system

The system which MITRE employed for the Senseval-3 logical form evaluation consists of the following components:

- the Humphreys/Carroll/Minnen morphological analyzer (Minnen et al., 2001)
- the CMU Link Grammar parser (Sleator and Temperley, 1991)
- a link interpretation language which is used to produce a dependency graph
- additional lexical knowledge sources
- an argument canonicalizer based partially on the principles of Relational Grammar (Perlmutter, 1983)
- a task-specific logical form generator

The morphological analyzer is straightforward, and we will not say more about it. We discuss the remaining components below.

2.1 The CMU Link Grammar parser

The Link Grammar formalism consists of labeled, undirected links among pairs of words. Each word in the Link Grammar dictionary is mapped to a complex logical expression of the link ends the word can participate in. These link ends have a major component (indicated by uppercase letters), a minor component (indicated by lowercase letters), and a required direction (looking leftward (-) or rightward (+)). Two words can be joined by a link if their link ends are compatible. The Link Parser provides reasonable performance achieving 75% labeled constituent accuracy on the TreeBank data. There are a large number of link types some of which provide very detailed distinctions beyond those found in phrase structure grammars. For further details, see (Sleator and Temperley, 1991).

Figure 1 shows the processing of the simple sentence *Chris loves Sam*. We describe link parser output as a set of 6-tuples, consisting of the index, word, and link end for each end of the link; we omit the direction information from the link, since it can be inferred from the tuple. For instance, *loves* at index 2 is joined to *Sam* at index 3 via an O link; *loves* bears O looking rightward in the lexicon, and *Sam* bears O looking leftward, and these link ends are compatible. As mentioned, ndividual lexical items may (and often do) have multiple link types associated with them (e.g. *Sam* also bears S looking rightward for the case when *Sam* is a subject.)

input sentence	`Chris loves Sam`
link parser output	`1 Chris Ss 2 loves Ss` `2 loves O 3 Sam Os`

rules	`(1) LINK ( S SF SX ) role[left]: arg:S role[right]: head`
	`(2) LINK O role[left]: head role[right]: arg:O`
	`(3) FEAT ( S- SX- ) category: v`

dependency object	`[v [ Chris_1:H]:S loves_2:H,3singular,present [Sam_3:H]:O]`
logic form	`Chris (x1) loves:v_ (e1, x1, x2) Sam (x2)`

Figure 1: Processing "Chris loves Sam"

Link parses contain a great deal of detail, but because the link parser is a general-purpose tool, extracting this detail for a particular task may require further processing. In particular, the category and head/dependent information that is needed for logical form generation can be computed to a large degree, but is not explicitly present. Our link interpretation language addresses this issue.

2.2 The link interpretation language

Our link interpretation language operates on the output of the link parser, and assembles a dependency graph. The link interpretation language can assign properties and categories to individual link ends via FEAT rules, and assign head/dependency relations to links via LINK rules.

Look again at Figure 1. Rule (1) applies to any link whose ends are compatible with the link ends S, SF or SX[1] . This rule assigns the `arg:S` role (i.e., subject argument) to the left end of the link, and the `head` role to the right end. In other words, if two words are linked by an S link, the left element is the subject of the right element. Rule (2) creates an analogous dependency for the O link, making the right element the object of the left element. Rule (3) says that anything on the leftward-looking end of an S or SX link) should be assigned the category `v`; i.e., it's a verb.

The LINK rules can assign a range of roles, including:

- head
- argument of a particular type (e.g., S or O)
- modifier of a particular type (e.g., DET)
- merge, which promotes all dependents of the merged element and constructs a complex lexical head (e.g., for idioms or multi-word proper names)
- filler and hole, which establish relationships related to unbounded dependencies

In addition, LINK and FEAT rules can assign roles, properties and categories to the parents of the left and right elements when necessary, and the processor postpones these assignments until the appropriate parent relationships are established.

The processor which interprets this language begins by assigning a dependency object to each word in the sentence; the word is the head of the dependency object, and the object has no dependents. The processor then looks at each of the links, in any order. It applies all relevant FEAT operators to each link end, and finds the first LINK rule which applies. If any LINK rules which must be postponed are found, the processor collects all candidate rules, and chooses among them after the parent relationships are established.

The output of this procedure as shown in the fourth row of Figure 1 is a set of interconnected dependency objects. Every dependency object which has been identified as a non-head link end will have the object it depends on as its parent. In the ideal case, this set will have only one parentless object, which will be the dependency object associated with the matrix verb. Figure 1 also shows the topmost dependency object for our example sentence; in this representation, each word or constituent bears a suffix indicating that it is the head (`:H`) or the relationship it bears to the head (e.g., `:O`).

In general the process of adding LINK and FEAT rules was carried out in a data-driven manner. Currently, there are 88 LINK rules and 63 FEAT rules. While the number of potential rules is quite large due to a large number of link types, catagories, and properties, we have found that these rules generalize reasonably well and expect that the remaining rules that would be required to represent very specific cases.

[1] S links are simple subject-verb relations, SF is used for the special case where the subject is *it* or *there* (e.g. It was raining.), and SX is used whent he subject is the first person pronoun *I*.

2.3 Additional lexical knowledge sources

For the purposes of deriving logical forms, the link parser output doesn't contain quite enough information. We rely on two additional sources of lexical knowledge: a small dictionary, developed in concert with the link interpretation language, which identifies features such as *auxiliary* for verbs, and a body of lexical control information, derived from subcategorization classes in Comlex (Macleod et al., 1998). The first source informs the link interpretation process, by identifying which verbs are dependents of other verbs. The second source informs our next step, the argument canonicalizer.

2.4 The argument canonicalizer

In this step, we construct an argument network for each dependency object, in the spirit of Relational Grammar (Perlmutter, 1983). For those predicative phrases in argument positions which lack a subject, we determine and assign a subject to control the phrase. We use the lowest available grammatical relation (first object, then subject) as the controller, unless the information we've collected from Comlex indicates otherwise (e.g., in the case of *promise*). We then identify those argument networks to which Passive has applied, and undo it, and do the same for Dative Movement, in order to derive the canonical predicate argument order.

2.5 Deriving the logical forms

At this point, we have all the information we need to derive the logical forms required for this evaluation track. We generate logical forms via the following steps:

1. We eliminate those words for which no output is required (e.g., determiners).
2. We identify the remaining words which require a part of speech suffix (e.g., nouns but not proper nouns).
3. We identify the remaining words which take arguments (e.g., verbs but not nouns) and those which add their own instance variable (e.g., verbs but not prepositions).
4. We add the appropriate argument structures for noun-noun compounds, and make other task-specific adjustments.
5. We collect and format the appropriate predicates and argument lists.

In some cases, a subject argument was required, but we could not infer the appropriate filler; in these cases, we insert the string "MISSING" as the logical subject in the logical form.

3 Results

Table 1 shows the precision and recall over both arguments and predicates. Table 2 includes the precentage of sentences of which all arguments were identified (SentArg) and all predicates were identified (SentPred). SentArgPred indicates the percentage of sentences for which all arguments were identified correctly out of sentences that had all predicates identified correctly. SentArgPredSent is the percentage of sentences for which all arguments and all predicates were identified correctly (SentArgPredSent).

	Precision	Recall
Arguments	0.74	0.66
Predicates	0.84	0.78

Table 1: Argument and predicate precision and recall.

	Accuracy
SentArg	0.27
SentPred	0.21
SentArgPred	0.40
SentArgPredSent	0.087

Table 2: Sentence-based accuracy of extracted logic forms.

Clearly, these results indicate room for improvement in this task.

4 Comments on the evaluation

We found some problems in this evaluation.

4.1 Resolving vagueness in the task

In some cases, the details of the task are vague. One example is collocations. The task description clearly allows for collocations (e.g. *proud of, at a loss*), but there is little guidance about how to decide whether some word sequence should be a collocation. These decisions affect the system scores, and the absence of clear guidance on this issue clearly suggests uncertainty about what the scores mean. Having an official list of collocations is only one part of the solution, however. Since collocations obscure internal structure, creating a collocation potentially loses information; so the issue isn't simply to know what's on the list, but to have some guideline for deciding what should be on the list.

One way in which to motivate guidelines, define scoring metrics, etc. is to include a more goal-directed task description. The last two decades of research in computational linguistics have cemented

the crucial role of system evaluation, but the summary in (Hirschman and Thompson, 1996) makes it clear that the best evaluations are defined with a specific task in mind. In a previous attempt to define predicate-argument structure, Semeval, the effort was abandoned because so many constructs would require detailed attention and resolution, and because most information-extraction systems did not generate full predicate-argument structures (most likely because the task did not require it) (Grishman and Sundheim, 1996). While introducing a task creates its own problems by removing domain independence, the constraints it provides are worth consideration. For example, in a task such as Question Answering, certain distinctions in the logic-form presented here may serve no purpose or perhaps finer grained distinctions are required.

As another example of this issue, the scorer provided for this task computes the precision and recall for both predicates and predicate arguments in the logic forms. In some circumstances, the scorer assigns the same score for predication of an incorrect, independently specified variable (e.g., `x2` instead of `x1` as the first argument of `loves` in Figure 1) as for predication of an otherwise unspecified variable (e.g., `x3` instead of `x1`). This may be an informative scoring strategy, but having a more specific task would help make this decision.

4.2 Suggested improvements in the logic

In many ways, it's also impossible to make judgments about the syntax and implied model for the logic without a more specific task, but it's still worth pointing out some inconsistencies.

First, the implied account of noun-noun compounds introduces an `nn` predicate, but assigns to the resulting phrase a different variable than either of the nominal constituents. Adjectival modification, on the other hand, is represented by sharing of variables. (Rus, 2002) argues for this account of noun-noun compounds (p. 111), but provides no motivation for treating the noun-noun compound `goat hair` as having a separate variable from its head but not doing the same for the adjective-noun sequence `curly hair`.

Second, the account of pronominal possessives (*our*, *my*) would lead to a poor account of full possessives. The possessive pronoun shares a variable with its possesseed, which does not allow a parallel or adequate account at all of the full possessives (e.g., *the poor boy's father* could only have *boy*, *poor*, and *father* assigned to the same index). The possessive should be treated like noun-noun compounds, with a `poss` operator.

Finally, adverbs which modify adjectives have nothing to attach to. In the single example of this construction in the sample data (*Sunshine makes me very happy*) the modifier *very* is predicated of *me*, because *happy* is predicated of *me*. This account leads immediately to problems with examples like *John is very tall but hardly imposing*, where all four modifying elements would end up being predicated of John, introducing unnecessary ambiguity. Introducing properties in the logic as individuals (cf. (Chierchia and Turner, 1988)) would almost certainly be an improvement.

References

G. Chierchia and R. Turner. 1988. Semantics and property theory. *Linguistics and Philosophy*, 11:261–302.

Ralph Grishman and Beth Sundheim. 1996. Message understanding conference - 6: A brief history. In *Papers presented to the Sixteenth International Conference On Computational Linguistics (COLING -96)*, University of Copenhagen.

L. Hirschman and H. Thompson. 1996. Overview of evaluation in speech and natural language processing. In R. Cole, editor, *Survey of the State of the Art in Human Language Technology*, chapter 13. Cambridge University Press, Cambridge.

Catherine Macleod, Ralph Grishman, and Adam Meyers, 1998. *COMLEX Syntax Reference Manual*. Proteus Project, NYU.

G. Minnen, J. Carroll, and D. Pearce. 2001. Applied morphological processing of English. *Natural Language Engineering*, 7(3):207–223.

David M. Perlmutter, editor. 1983. *Studies in Relational Grammar 1*. University of Chicago Press.

Vasile Rus. 2002. *Logic Forms for Wordnet Glosses*. Ph.D. thesis, Southern Methodist University.

Daniel Sleator and Davy Temperley. 1991. Parsing English with a Link Grammar. Technical Report CMU-CS-91-196, Carnegie Mellon University Dept. of Computer Science, October.