SEMANTIC EVALUATION FOR SPOKEN-LANGUAGE SYSTEMS

Robert C. Moore

Artificial Intelligence Center
SRI International
Menlo Park, California 94025

ABSTRACT

Development has begun on a semantic evaluation (SemEval) methodology and infrastructure for the ARPA Spoken Language Program. SemEval is an attempt to define a task-independent technology-based evaluation for language-understanding systems consisting of three parts: word-sense identification, predicate-argument structure determination, and identification of coreference relations. An initial spoken-language SemEval on ATIS data is planned for November/December 1994, concurrent with the next ATIS CAS (database answer) evaluation.

1. INTRODUCTION

Since the summer of 1993, there has been considerable discussion in the ARPA HLT community of moving the evaluation of understanding systems for both spoken and written language away from application-based metrics (such as correct database response in ATIS, or template fills in MUC) toward technology-based metrics. The benefits hoped to be derived from such a shift include greater focus on underlying technology issues, rather than application issues, and lowering the overhead required to participate in evaluations in terms of developing application systems. The discussions have focused on the concept of a semantic evaluation, or "SemEval" consisting of three components: word-sense identification, predicate-argument structure determination, and coreference determination. This paper reports on how these ideas are being developed within the ARPA spoken-language community, in preparation for an initial spoken-language SemEval on ATIS data concurrent with the ATIS CAS (database answer) evaluation planned for November/December 1994.

A meeting was held at SRI, 21-23 October 1993, to begin fleshing out these ideas for the evaluation of spoken-language understanding systems. The meeting was attended by researchers, annotators, and evaluators involved in both the ARPA Spoken Language Program and the ARPA Written Language Program: Fernando Pereira from AT&T Bell Laboratories; Rusty Bobrow and Dave Stallard from BBN; Wayne Ward and Sergei Nirenburg from CMU; Stephanie Seneff and Eric Brill from MIT; Robert Moore, Kate Hunicke-Smith, Jerry Hobbs, Harry Bratt, and Mark Gawron from SRI; Debbie Dahl and Lew Norton from Unisys; Mitch Marcus and Grace Kim from the University of Pennsylvania; Nancy Chinchor from SAIC; George Doddington from ARPA; George Miller from Princeton University; Dave Pallett and Bruce Lund from NIST; and Ralph Grishman from NYU. This paper is derived from the discussions at the October meeting and from subsequent proposals made and discussed by the participants.

2. WHY DO SEMEVAL?

A question that has received, and continues to receive, extensive discussion is "Why do we want to do SemEval at all?" George Doddington [1] has addressed this from ARPA's perspective as follows:

> Why is SemEval a good idea? Well, first, ARPA's goal in the HLT Program is to make strategic advances in core human language technology. This goal is a technology goal (to produce useful functionality). It is not a science goal (to produce scientific understanding), nor is it an application goal (to produce useful applications embodying human language technology). So, why do SemEval instead of only doing task-level evaluations (such as ATIS CAS)? There are three reasons:
>
> 1. SemEval offers the possibility of providing a much more direct and objective evaluation of underlying technical issues. It thus promises greater diagnostic leverage which would yield more rapid and efficient development of core technology.
> 2. SemEval, by measuring performance at a technical level rather than an application level, eliminates much overhead and research inefficiency by obviating the need to support application effort and other back-end issues. This makes research much more efficient by focusing a greater fraction of effort on research and technical issues of direct interest. It also makes it more attractive and much easier for a new player to enter the game.
> 3. SemEval, by virtue of measuring performance below the application level, offers the opportunity to compare performance across different applications and to support formal evaluation among a much larger research community. Thus the potential benefit and evaluation support goes far beyond the relatively few ARPA HLT sites.
>
> The risk is that SemEval may end up measuring technical aspects of systems that are not directly relevant or do not represent the important issues in HLT. We need to try hard to avoid this. And we

won't immediately abandon other task-level evaluations such as the ATIS CAS. But a successful SemEval has the potential to create a larger research challenge and to accelerate HLT progress very significantly. The potential payoff is high and we need to give SemEval the best shot we can.

3. METHODOLOGICAL PRINCIPLES

Perhaps the main difficulty in defining SemEval is that there is no single generally accepted notation for representing the meaning of natural-language utterances. Instead, there are numerous notations and theories, from which we have to synthesize a notation that is compatible with as wide a variety of points of view as possible. Two methodological principles have been proposed to guide us in this task.

The first is proposed as a strategy for helping us define annotations without bogging down in theoretical disputes: Take as our mantra "It's just a notation, not a theory." That is, the overriding consideration should be whether a proposed notation is a convenient way of marking a distinction that we agree we should mark, and that whether it meets other, theory-driven conditions (e.g., supporting a truth-conditional semantics, assigning a type-theoretic interpretation to every subexpression, being compositional, assigning one predicate per morpheme) is beside the point.

The second methodological principle is a default rule for deciding when to make distinctions. There are many cases where it is not immediately clear whether to mark a distinction or not. The proposed default is *not* to mark distinctions. That is, it should take some positive argument that if the distinction is not marked, two utterances that we agree should be assigned different structures will be assigned the same structure. (Often, the most compelling such arguments are truth-conditional. That is, we can describe a situation where one utterance is clearly true and the other is clearly false.)

The reason for defaulting to not making distinctions is that systems that make more distinctions than necessary should be able to collapse them for purposes of translation to the canonical representation fairly easily, but a system that doesn't make a distinction at all will be severely penalized if it is scored incorrect for not making it. Hence, we need evidence that the system is *wrong* not to make the distinction. Note that marking distinctions does not mean that the notation will represent a superficial analysis of the utterance. Quite the contrary, it will often require giving a common representation to many expressions that are quite different syntactically and, hence, push toward deeper representations.

4. PREDICATE-ARGUMENT STRUCTURE ISSUES

So far, most of the detailed proposals that have been discussed pertain to predicate-argument structure issues.

4.1. Syntax for Predicate-Argument Structure

In discussions of a possible syntax for predicate-argument structure, it has been evident that there are considerable variations in preferences for the amount of syntactic sugar to be used. To accommodate these differences in preferences, three intertranslatable levels of notation have been proposed. The first is simply LISP-style nested functor-argument notation, with two notational additions. First, we use angle brackets <...> to indicate implicit conjunction, arising, for example, from iterated modifiers in the utterance. Second, we use numerical indices followed by a colon to label expressions that may fill more than one argument position in the predicate-argument structure. For example, if we make the assumption that `tall`, `block`, and `blue`, are one-place predicates and we ignore tense, *Every blue block is tall* might be represented:

```
(decl (tall 1:(every <(block 1)(blue 1)>)))
```

(The recursion implicit in the use of the index `1` will be explained in the discussion of quantification below.)

A second level of representation is obtained by assigning every expression an index, and breaking the structure down into a list of atomic predications, interrelated by the indices:

```
2:(decl 3)
3:(tall 1)
1:(every <4 5>)
4:(block 1)
5:(blue 1)
```

Finally, we may wish to break this notation down to an even more atomic level for the purpose of counting errors for scoring:

```
(funct 2 decl)
(arg1 2 3)
(funct 3 tall)
(arg1 3 1)
(funct 1 every)
(arg1 1 4)
(arg1 1 5)
(funct 4 block)
(arg1 4 1)
(funct 5 blue)
(arg1 5 1)
```

4.2. Scope of Quantified Noun Phrases

There has been general agreement that quantified noun phrases should be represented "in place," without their exact scope being represented. To be more precise, for a generalized quantifier $Q(X,Y)$ corresponding to a noun phrase determiner like *some* or *every*, since the restriction X is essentially the content of the noun phrase, that would be indicated in the notation, but the body Y, not being structurally determined by the syntax, would be left vague. This means we will give different representations to

Every tall block is blue.
Every blue block is tall.

since the difference between these is structurally determined and unambiguous, but we will not give different representations to the two scopings of

Some girl likes every boy.

that is,

There is some girl such that she likes every boy.
For every boy, there is some girl who likes him.

since that distinction is not structurally determined (although structure has some influence), and it is often difficult to judge.

An additional constraint that has been agreed on is that the notation should not have what has come to be called the "linchpin problem"; that is, the notation should not be such that, if one key piece is missed, the whole thing falls apart and credit is given for virtually nothing. In particular, it should be possible to miss which pieces belong inside or outside the restriction of the quantifier, and still get credit for recognizing all the predications of the "quantified variable."

The solution that has been developed is illustrated by the predicate-argument structure given above for the phrase *every blue block*:

```
1:(every <(block 1)(blue 1)>)
```

The expression as a whole is given an index that is also used in the argument positions inside the restriction of the quantifier that correspond to "quantified variable" positions. That way, the notation can clearly indicate which predications are part of the quantifier restriction, but every argument position that would be filled by the quantified variable in a standard logical representation is filled with the same index, whether the predication is inside or outside of the quantifier restriction.

Quantified noun phrases are not the only constructs where the issue of scope arises. Others include modal verbs, negation, and propositional attitude verbs (e.g., *want, know*). It has generally been agreed that items whose scope is largely determined by linguistic structure would have their scope indicated in predicate-argument structure (usually by treating them as having propositions as arguments), and that items whose scope is not determined by linguistic structure (such as *only*), would have their structure left underspecified in predicate-argument structure.

4.3. What are the Predicates and Arguments?

There are two widely used schemes for mapping linguistic heads (e.g., main verbs) and complements (e.g., subjects and objects) into predicate-argument structures. In one approach, the head is treated as a multi-argument predicate and the complements are distinguished by the position they fill in the argument list:

```
(P X Y Z)
```

In the other approach, a "Davidsonian" [2] notation is used, in which an "event" described by the head is introduced, and the complements are treated as fillers of role relations for the event. In the notation we have adopted, this comes out looking something like

```
<(ev-type 1 P) (R1 1 X) (R2 1 Y) (R3 1 Z)>
```

We have chosen the Davidsonian-style notation, because of its flexibility in leaving open exactly what complements (and adjuncts) a head has.

After the decision to use a Davidsonian representation, the question came up of how widely to apply it. Davidson's original proposal was intended to apply only to verbs (in particular, only to action verbs), but examples arise with adjectives, adverbs, nouns, and even prepositions that seem to require a similar treatment. We have tentatively decided, therefore, to apply it to all of these types of expressions, but to provide syntactic sugar to hide some of the complexity in simple cases.

4.4. Collapsing Lexical and Syntactic Distinctions

It has been tentatively agreed that a number of syntactic distinctions should be collapsed in predicate-argument structure:

Active vs. passive: *Mary kissed John,* vs. *John was kissed by Mary.*
Dative movement: *John gave Mary a book,* vs. *John gave a book to Mary.*
Raising verbs: *It seems that John is here,* vs. *John seems to be here.*

It has also been agreed that verbs and their event nominalizations should be given the same underlying predicates, for example, *arrive* and *arrival.*

It also appears that there are cases where multiple subcategorization patterns for the same verb can be handled by a single underlying predicate with different roles expressed. For example, in *John baked Mary a cake,* and *John baked a cake, bake* would be taken to express the same predicate, but with who the cake was for expressed by a role relation in only the first case. Other examples falling under this heading include "control" verbs, where if a certain role is not expressed, it is constrained to be filled by the same item as one of the other roles—for example, *John expected Mary to win* vs. *John expected to win* (= *John expected himself to win*).

Another category of verbs that first seemed to fall into this class are those for which both transitive and intransitive forms exist and it appears that the object of the transitive form may fill the same role as the subject of the intransitive form—for example, *John melted the butter*, vs. *The butter melted.* On closer examination, however, it seems there may be good reasons for treating these as distinct predicates, so this issue remains open.

4.5. Complex Predicates

So far we have represented all predicates as atomic, but we might in some cases want to have predicates that are themselves structurally complex. One case is complex determiner phrases. We have treated determiners like *some* and *every* as a sort of predicate, but sometimes determiners are complex phrases like *no more than seven.* A second potential example of complex predicates are families of related prepositions, like *at*, *before*, and *after.* It has been suggested that these might be profitably treated as utilizing a single underlying predicate, whose interpretation varies from domain to domain, together with a set of numerical comparison operations defined in terms of that predicate but fixed across all domains.

5. WORD SENSE AND ROLE IDENTIFICATION ISSUES

Much of the discussion of word-sense identification issues has revolved around whether WordNet [3] would be a suitable lexical resource to use as a source for word senses. The general impression seems to be that it probably is, but the details remain to be worked out.

The choice of the Davidsonian representation for head-complement relations raises an important issue closely related to that of word-sense identification—namely, identification of the role-relations that hold between the events and the complements (R1, R2, and so forth in the discussion above). One possibility is to use fairly superficial identifiers (such as abbreviations for "logical subject" and "logical object") or surface prepositions for role names, to keep the annotations domain-independent. An objection to this is that it is too syntactically oriented and does not represent a deep enough level of understanding.

The approach currently being explored is to attempt to define a set of semantic classes relevant to the domain and construct role names from those classes. If there is only one relation between a pair of classes salient enough to be expressed by a grammatical role or preposition, then a simple concatenation of the class names would be used. For example, for "a flight on an airline", there is really only one salient relation between flights and airlines, so `flight_airline` might as well be used to name that relation. If there is more than one salient relation between two semantic classes, a grammatical relation name or preposition can be interpolated between the semantic class names. So, since `flight_airport` would be ambiguous between origin and destination, we would have `flight_from_airport` and `flight_to_airport` instead.

Since theory-laden terms are not used to name the roles, this approach should avoid arguments such as whether a particular role is really an agent or is merely an experiencer. It also offers the possibility of expressing deeper regularities than grammatical roles or surface prepositions, since it allows us to say that

ticket's price
price of a ticket
price for a ticket
price on a ticket

all involve the same relation between a ticket and a price.

A number of key issues raised by this approach remain to be resolved, including the following:

1. Roles may be expressed (at least) by grammatical relations, prepositions, possessives, and the verb *have.* One question is whether we ever want to treat any of these constructions as having an autonomous sense, rather than expressing a role of some predicate. For example, one might want to say that in *a book on a table*, *on* simply expresses a relation between the book and the table that depends only on an autonomous sense of *on* independent of the predicate *book*, while in *a price on a ticket*, *on* expresses the role of the predicate *price* that is filled by *a ticket.*
2. Under this proposal, it is necessary to know what the semantic class of the head of a phrase is in order to know what to call the roles that are expressed. Some phrases have null heads or contentless heads that pose a problem for this approach—for example,

   Show me the ones on United.

   In this case, we need to know what *the ones* refers to in order to know what relation *on* expresses. Conversely, conjunction can create situations where a phrase provides a role filler for two different heads:

   Show me the flights and fares to Boston.

   In a case like this, the notation needs to allow *to Boston* to supply a role filler for both *flights* and *fares.*

6. COREFERENCE DETERMINATION ISSUES

We take the term "coreference" very broadly to include a variety of types of constraints from context. Most of the cases that have been considered so far can be classified into three categories:

1. Strict coreference, where one expression denotes exactly the same entity as some other expression:

   Show the flights from Boston to Dallas and the times they arrive.
   they = the flights from Boston to Dallas
2. Relational coreference, where one expression denotes something bearing a specific relation to an entity denoted by some other expression:

   Show flights from Boston to Dallas and discount fares.
   discount fares = discount fares for flights from Boston to Dallas
3. General constraints from context:

I need to go from Boston to Dallas. Show me all the morning flights.
the morning flights = the morning flights from Boston to Dallas

The current proposal for annotating these relations is to use a combination of co-indexing and expressing contextual constraints by constructing additional pieces of predicate-argument structure (which might or might not be copies of pieces of predicate-argument structure in the context). The feasibility of specifying these additional pieces of predicate-argument structure in a sufficiently constrained way to yield a canonical representation is currently being assessed.

7. ANNOTATION AND TEST ISSUES

We assume that SLS SemEval will work as much as possible like the ATIS CAS evaluations in terms of how we organize the collection and annotation of data and administration of the evaluation. In the general case, the expected process is:

1. At multiple sites, data will be collected, transcribed, and shipped to NIST.
2. NIST will partition data into training and test and ship data to third-party annotators.
3. Annotators will perform classification and annotate word-sense, predicate-argument structure, and coreference, and ship annotations back to NIST.
4. NIST will distribute training data with classifications and annotations to system developers.
5. A committee will resolve issues about how to classify and annotate data and to maintain documentation on the same.
6. A mechanism will be established for reporting training data bugs to NIST, having the bugs corrected, and distributing the fixes.
7. NIST will release test data shortly before the evaluation, and participants will submit annotations produced by their systems to NIST for comparison with reference annotations.
8. An adjudication process will be set up to resolve disputes about the transcription, classification, and annotation of test data.

Note that for the initial SLS SemEval the first two steps have already been completed, because we will use some of the same data for training and test that has already been collected for ATIS CAS.

Obtaining consistent annotation of the data is an important requirement. It is clear that a detailed annotation manual and good annotation tools need to be developed and that tight feedback between the annotators and an analog of the ATIS CAS Principles of Interpretation committee will be required.

Another important question is whether a detailed lexicon with patterns illustrating word senses and roles for the majority of the vocabulary will need to be constructed before annotation, or whether this can be developed by the annotators as they proceed. One suggestion is an iterative process, whereby a subset of the data would be annotated using a partial lexicon, with the annotators having the option of choosing "none of the above" for a word sense or role relation. A concordance would be produced for the "none of the above" occurrences of each lexical item and role marker, and new word senses and roles would be added to the lexicon based on an analysis of the concordance. It has also been suggested that a threshold be set in terms of frequency of occurrence, and until some word sense or role relation exceeded the threshold, it could be left in the none-of-the-above bucket, and that none-of-the-above would be deemed the correct answer if that word sense or role relation turned up in test data. (None-of-the-above would also be deemed the correct answer if a completely new word sense or role relation turned up in test data.)

8. ANNOTATION AND TEST SOFTWARE

A number of pieces of software will be needed to support the overall process:

1. Annotator aids — Annotators cannot be expected to create complex annotations for utterances completely by hand. For ATIS CAS, NLParse was used, but for SemEval, NLParse is not suitable. One possibility would be to use one or more participants' systems to produce a first-pass annotation, which the annotators would then correct. There is some concern that this would produce annotations that are biased in favor of the system used to produce the initial structures. This might be partly alleviated by using multiple systems to produce a first pass, perhaps presenting to the annotators only the parts of the annotation that multiple systems agree on. The annotators will also need specialized editing tools tailored to creating and correcting SemEval structures. Such tools have been created by the Penn Treebank project [4] for producing syntactic bracketings of utterances, and it may be possible to adapt these for SemEval.
2. Annotation checker — The annotations themselves will have a quite complex syntax and semantics. Software to check the resulting annotations will no doubt catch many annotation errors. It might be possible to build this functionality directly into the editing tools.
3. Annotation translators — We have defined several levels of notation for SemEval. The highest, most syntactically sugared level seems likely to be used by the annotators, and the lowest level seems likely to be that to which the comparator is applied. However many levels are used, software to translate between them will be needed.
4. Comparator — It will be necessary to build a comparator for hypotheses and reference answers. This needs to be implemented in a way that permits all sites to use it, which would probably make C the implementation language of choice.

9. CONCLUSIONS

SemEval holds out the promise of shifting the focus of language-understanding evaluation from specific application tasks to generic technology. It is hoped that this will lower the overhead of participation in evaluations and focus the efforts of participating sites on key technological issues, thereby accelerating the rate of progress in the field. A substantial start has been made on defining SemEval for spoken-language understanding, but much work remains to be done. Over the next few months we expect to see these efforts converge on a workable plan for a spoken-language SemEval in late 1994.

Acknowledgments

I wish to thank everyone who has participated in the discussions of SLS SemEval, particularly those who attended the SLS SemEval meeting at SRI in October 1993. Special thanks are due to Debbie Dahl whose very extensive and excellent notes of that meeting provided an invaluable resource in the preparation of this paper, which was supported by the Advanced Research Projects Agency under Contract N00014-93-C-0142 with the Office of Naval Research.

References

1. G. Doddington, edited transcript of comments made at spoken-language SemEval Meeting at SRI International, Menlo Park, California (21-23 October 1993).
2. D. Davidson, "The Logical Form of Action Sentences," in *Essays on Actions and Events*, pp. 105-148 (Clarendon Press, Oxford, England, 1980).
3. G. A. Miller (ed.), "WordNet: An On-Line Lexical Database," International Journal of Lexicography (special issue), Vol. 3, No. 4, pp. 235-312 (1990).
4. M. P. Marcus, B. Santorini, and M. A. Marcinkiewicz, "Building a Large Annotated Corpus of English: The Penn Treebank," *Computational Linguistics*, Vol. 19, No. 2, pp. 313-330 (June 1993).