EFFICIENT SEMANTIC PARSING OF CONVERSATIONAL SPEECH

Michel Généreux
Information Technology Research Institute, Brighton, United Kingdom

Abstract

This paper presents an empirical method for mapping speech input to shallow semantic representation. Semantic parsing is realized through a bottom-up type parsing paradigm where the operators are based on semantic concepts, obtained from a lexicon. A statistically trained model specializes the parser, by guiding the runtime beam-like search of possible parses. The semantic representation is a logical form equivalent to a Discourse Representation Structure (DRS). Each output of the parser is given a probability according to how similar, given a contextual word similarity measure, the parsing process for the input was to those collected during the training phase. Contextual information during parsing allows for better coverage of large domains. The non syntactic but very semantic nature of the parser would make it very tolerant to noisy (recognized) speech input. Shallow parsing using First-Order Logic (FOL) allows for fast but meaningful enough processing of the input, which makes the parser well suited for real-time Spoken Dialogs Systems (SDS).

Keywords: Semantics, Corpus, Discourse

1. Introduction and Motivation

For task oriented systems, the quality of the spoken interaction between man and machine have seen constant progress over the last decade. Today, lower word error rate in speech technology, expertise gained in dialog management, more flexible natural language generation and better speech synthesis allows us to take dialog systems to the next level: open (or very broad) domain of interaction. To cope with the complexity of open domain and noisy speech input, semantic parsers will have to put a strong emphasis on context to supplement for syntactical analysis, and outputs a suitable meaning representation for discourse to be further interpreted. We present a parser with strong contextual capabilities that delivers a DRS as output.

The choice of our bottom-up parser is motivated by its manageability and its similarity to how humans parse a sentence [Hermjakob and Mooney (1997)]: compositionally build meaning from left to right by adding concepts as they appear (INTRODUCE), combining them (COREF and DROP operations) and keeping in mind contextual information (SHIFT operation). Finally, FOL [Light and Schubert (1994)] offers a reasonably deep semantic representation and a convenient way to translate DRSs, for carrying out sensible discourse conversations.

2. System Architecture

We propose an approach in which a bottom-up parser similar to [Mooney and Tang (2000)] is combined with a statistical model. Mapping input to logical form is triggered by keywords (INTRODUCE operation) from a semantic lexicon collected from training. Words in the input not in the semantic lexicon are used as contextual information (SHIFT operation).

Three other operations are available to the parser: co-referencing variables (COREF), droping one term into the argument of another (DROP) and giving scope to quantifiers (SCOPE). In the parser, all operations are conducted within a particular context, a context being a word or group of words following (in the strictly left-to-right sense) a parsing operation. The output of the parser is a a partially resolved DRS, ready to be processed by a Discourse Manager. Figure 1 shows the various elements of the parser.

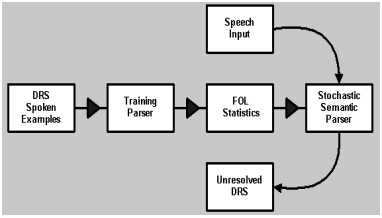

Figure 1: Parser Architecture

3. Overview of the Parsing Process

This section is meant to give a flavor of the parsing process and provides a ”light” introduction to the parser. All statistical considerations are for the moment deferred to section 5. The parser used is a variant of a *Shift-Reduce* parser (only the SHIFT operation has been retained).

The Input String The input string is a list of words to give an interpretation for. When no actions are applicable and the input string is empty, then the parsing process is completed. Typically, one word is removed from the list for a SHIFT action and one or more for an INTRODUCE action. It also provides some contextual information while applying a parsing action. Example 1 shows an input string.

(1) [I,read,The,Little,Mermaid,Did,you,write,it]

The Parse Stack The parse stack is the actual parse state, the current interpretation of the input string found so far. It is a list of binary elements, each element representing a combination of the introduced predicate (or concept) with its context of introduction. The context gives partial (but useful) information on the words following the concept at the time of introduction. Each concept must be in the semantic lexicon. Here is the general format of the parse stack: [**concept1:[context1],concept2:[context2],. . . ,start:[context]]**
The *start* predicate is there only to provide room for words from the input string which would be shifted at the very beginning of the parse; it does NOT contribute to the meaning of the input phrase. The representation of operators, parse state and final state follow.
SHIFT(word_to_be_shifted) A SHIFT action simply puts the first word from the input string at the end of the context of the concept on the top of the parse stack.
INTRODUCE(concept_to_be_introduced) The INTRODUCE action takes a concept from the semantic lexicon (the lookup is triggered by keywords in the input) and puts it on the top of the parse stack, initializing its context of introduction to the word (or list of words) that triggered this concept. These concepts will then participate to the meaning representation.

DROP(source_term, target_term) The DROP action attempts to place a term from the parse stack as argument to another term of the parse stack. The context of the source term is lost in the process. This action has no effect on the input string.
COREF(variable1, variable2) The COREF action attempts to co-reference two variables, in the case at least one of them is underspecified (_). The result is that they become specified (they have the same name). This action has no effect on the input string.
SCOPE(source_term, target_term) The SCOPE action is similar to the DROP action, with the two exceptions that it applies only to quantifiers and that the 'droping' is slightly different. For example, SCOPE(exists(A,human(A)),forall(B,thing(B))) results in the target term being forall(B,exists(A,(human(A),thing(B)))). This action has no effect on the input string.
op(ACTION(arguments)#Parse_Stack#Input_String) indicates in which context, i.e. how the Parse Stack and the Input String looked like, when the action took place. *Op* is simply a container for all types of actions.
final(Parse_Stack) indicates the final aspect of a parse, i.e. the meaning we have found for an input string.

Semantic Lexicon The semantic lexicon comprises all the concepts and their triggering phrase(s) that we wish our parser to process. A triggering phrase is simply a word (or group of words) in the input string that triggers some concept. The format of a lexical entry is: **lexicon(CONCEPT, [TRIGGERING_PHRASE]).**

The bottom-up parser We are now ready to present the variant of the shift-reduce parser we are using. The algorithm of the parser is as follows:

1. Try to INTRODUCE a new concept or SHIFT a word.
2. Do a subset of the following operations {DROP, COREF, SCOPE}.
3. If there are more words in the input string, go back to Step 1. Otherwise stop.

A parsing example We show a complete parsing in the case the user turn is *OOV did you write it?*, where OOV is a out of vocabulary symbol produced by the speech recognizer:

INPUT-OPERATION
*NEW PARSE STACK in FOL
[OOV did you write it]-SHIFT
*[start:[OOV]]
[did you write it]-SHIFT
*[start:[OOV,did]]
[you write it]-INTRODUCE
*[∃(_,system(_)):[you],start:[OOV,did]]
[write it]-COREF
*[∃(A,system(A)):[you],start:[OOV,did]]
[write it]-INTRODUCE
*[write(_,_):[write],∃(A,system(A)):[you],start:[OOV,did]]
[it]-DROP
*[∃(A,system(A),write(_,_)):[you],start:[OOV,did]]
[it]-COREF
*[∃(A,system(A),write(A,_)):[you],start:[OOV,did]]
[]-INTRODUCE
*[∃(_,nonhuman(_)):[it],∃(A,system(A),write(A,_)):[you],start:[OOV,did]]
[]-COREF
*[∃(B,nonhuman(B)):[it],∃(A,system(A),write(A,_)):[you],start:[OOV,did]]
[]-SCOPE
*[∃(A,∃(B,nonhuman(B),system(A),write(A,_))):[you],start:[OOV,did]]
[]-COREF
*[∃(A,∃(B,nonhuman(B),system(A),write(A,B))):[you],start:[OOV,did]]

Discourse Representation Structure Discourse Representation Theory [Kamp and Reyle (1993)] provides a well formalized framework for handling discourse phenomena such as pronoun and presupposition resolutions. Moreover, DRT means of representing meaning, DRSs, can be translated directly into FOL formula. DRSs can be assimilated to boxes having two regions: the top half region contains the *discourse referents* and the bottom half the *conditions*.

4. Training

In training, a training parser is used to generate FOL statistics from DRS annotated training examples.

Spoken Examples in DRS Training format is: **training([phrase], DRS).**
Had we trained the system on recognized output, we could have the following entry:
tr([OOV,did,you,write,it],drs([A,B],[nonhuman(B),system(A),write(A,B)]))

Training Parser While training, a *Training Parser* is used. It tries any possible actions to get to the final parse, without considering any information (such as *statistics*) that could be helpful to guide the parsing process. In training, a *training beam* can be specified. This means that only a certain number of parses will be recorded in the *FOL statistics* for each training example.

FOL statistics The training parser parses the examples to generate the *FOL statistics*. Every step needed to go from the *phrase* to the *DRS* is recorded, as well as final states themselves. Final states are simply the states of the parse stack themselves at the end of the parse. Each of them (actions and final states) are assigned a frequency measure. Each line has either one of the following format (recall that *op* is a container for any action):

```
op(ACTION#PARSE_STACK#INPUT_STRING#FREQUENCY).
final(FINAL_STATE#FREQUENCY).
```

Here are two examples:

```
op(SHIFT(did)#[start:[OOV]]#[did,you,write,it]#0.3).
final([exists(A,exists(B,nonhuman(B),system(A),
 write(A,B))):[you],start:[OOV,did]]#0.2).
```

These statistics are used by the *Stochastic Parser* to compute the best parse.

5. Statistical Parsing

The actual parsing of the input phrase is done by a *Stochastic Parser*. It uses a statistical model to process all the information available from the training phase in order to get the best possible parse (the one with the highest probability). This section describes the statistical parser in some details.

The Search space Like in the training phase, the most obvious way to influence the parse is to tell the parser how many parses it should try before taking a decision. We call it the *search beam* parameter.

Measure of similarity between lists When the parser tries to choose a suitable parse, it must compare list of words (to compare *Actions*, *Parse stacks* or *Input strings*). A good *similarity* measure between lists is essential, but because computing similarity is very demanding on computer resources, one must find a trade-off that preserves computational efficiency. The approach taken is based on n-grams [Cavnar and Trenkle (1994)].

Parametrizing the model The best parse P is found by taking the highest probability P_i among the possible parses (limited by the search beam) available:

$$\mathsf{P} = \max_i P_i \tag{2}$$

Each of these parses P_i have a probability that amounts to combining the probability of the individual *op* or actions together ($\prod_k a_k$, see equ. 4) and adding the probability of the final state ($ProbF$, see equ. 5). These two components are weighted by Pop and $Pfinal$. Those weighting values must be chosen in such a way that translates the importance of the steps needed to get to a final parse compared to the final state itself. In short, the weighting of actions taken together must be high enough to discriminate among similar final states (in terms of probability), should that case arise.
Multiplying by 100 gives a more readable value between 0 and 100.

$$P_i = (Pop * (\prod_k a_k) + Pfinal * ProbF) * 100 \tag{3}$$

The way each *op* a_k is assigned a probability is by taking into account its *similarity* with one of the *ops* in the statistics (P_m) as well as the *frequency* of this *op* ($Frequency$). These two components are also weighted by Pop_sim and Pop_occ. Default values are chosen with respect to how one would want to consider the respective importance of similarity over frequency.

$$a_k = max_m(Pop\_sim * P_m + Pop\_occ * Frequency) \tag{4}$$

Computing the probability of a final parse state is similar to computing the one for actions. A final state probability $ProbF$ is the weighted sum of the most similar final state in the statistical file P_f (see 6) and the frequency of this final state $Frequency$:

$$ProbF = \max_f(Pfinal\_sim * P_f + Pfinal\_occ * Frequency) \tag{5}$$

$$P_f = max_n(sim(t_n, F)) \tag{6}$$

Conventional smoothing techniques are applied whenever necessary.

6. Experimental Results

We have conducted the usual cross-validation testing by dividing our 250 sentence corpus into 10 testing samples of 25 sentences. In training, we produced at most 5 parses for each example. Results for parsing are reported in the following table:

Average/N-Best	1-Best	2-Best	3-Best
Recall-Precision	62%-64%	78%-80%	88%-91%

The parser always produces a valid output, unless there is no keywords introducing a concept in the input: this explains why *Recall* and *Precision* are very similar. The N-Best column is interpreted as follows: the correct output was AMONG the N results produced. A correct output must be exactly like the one produced by the human annotator. The 62% result for 1-Best may seem very modest, but in the context of a conversation, we believe it is more important to focus on the more comfortable 88% 3-Best result. The reason is that in a dialog system, the dialog manager (DM) have access to some sort of history or context to arbitrate between the N-Best semantic interpretations delivered by the semantic parser; in some occasions, it may therefore be preferable for the DM to get more than one parse. We ran the experiment on a 2GHz laptop computer under Sicstus Prolog. Keeping in mind that the system would be run for real-time conversations, we set a threshold of 20 parses or 3 seconds (whatever is reached first) for parsing to complete.

7. Conclusion

In this paper, a new probabilistic framework for semantic parsing is presented. The combination of a *bottom-up* parser and a purely statistical model makes it unique. More precisely, the parser learns efficient ways of parsing new sentences by collecting statistics on the context in which each parsing action takes place. It computes probabilities on the basis of the similarities of those contexts and their frequencies. The result is a simple and robust parser for speech. At this point, we believe that the 1-Best hypothesis recall could be improved by a higher ratio training/lexicon size. However, testing shows excellent results for the 3-best hypothesis. This system offers an approach in which linguistics can play a decisive role. One crucial aspect of the parser, the computation of similarities between context, relies on a good interpretation of linguistic patterns found in phrases, and how those configurations may determine the particular meaning of a word or group of words. This is essential to interpret, and maybe *understand*, conversational speech.

References

Cavnar, W.; Trenkle, J. 1994. N-gram-based text categorization. In: *Proc. of SDAIR-94, 3rd Annual Symposium on Document Analysis and IR*, Las Vegas, US. 161–175

Hermjakob, U.; Mooney, R. 1997. Learning Parse and Translation Decisions From Examples With Rich Context. Technical report: Dept. of Comp. Sciences, Univ. of Texas

Kamp, H.; Reyle, U. 1993. From Discourse to Logic. Dordrecht: Kluwer

Light, M.; Schubert, L. 1994. Knowledge representation for lexical semantics: Is standard first order logic enough?. In: *"Future of the Dictionary" workshop*

Mooney, R.; Tang, L. 2000. Automated construction of database interfaces: Integrating statistical and relational learning for semantic parsing. In: *Proc. of EMNLP/VLC-2000*, Hong Kong. 133–141

Young, S. 2002. Talking to machines (statistically speaking). In: *Proc. ICSLP*. 9–16

MICHEL GÉNÉREUX is a Research Fellow, Information Technology Research Institute, Brighton. He received his Dr.Phil. in Computational Linguistics at the University of Vienna, dealing with Spoken Dialogue Systems. His current research interests concern style in Speech Generation.
E-mail: michel.genereux@itri.brighton.ac.uk