Making the Most of Multiplicity: A Multi-Parser Multi-Strategy Architecture for the Robust Processing of Spoken Language

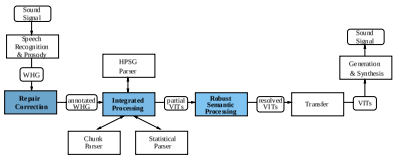
Jörg Spilker *Hans Weber* — *C. J. Rupp* *Karsten L. Worm* — *Tobias Ruland*

bmb+f Verbmobil Verbundvorhaben

The Verbmobil Project

- Machine Translation of Spontaneously Spoken Dialogues in Appointment Scheduling and Travel Planning Dialogues
- Several Translation Strategies:
  - Semantic Transfer
  - Statistical Translation
  - Dialogue Act-Based Translation
  - Example-Based Translation
- First Phase: 1993-1996, German → English and Japanese → English
- Second Phase: 1997-2000, German ↔ English and German ↔ Japanese, 10.000 words for German and English, 2500 for Japanese; 21 partner institutions

The Relevant Part of the System Architecture



The Topics of this Paper

We aim at making the best use of the multiplicity of parsers and strategies employed in the system in order to maximise the

1. Robustness and
2. Efficiency

of the system. We discuss the following points:

- Treatment of Self-Repairs
- Control of Several Parsers
- Integration of Partial Analyses
- Selection of Results

In this paper, we focus on the linguistic processing stream of the Verbmobil system.

Treatment of Self-Repairs

- The "repair correction" step itself relies on the classical treatment of speech repairs as *Reparandum* (RD), Interruption point (IP), *Edit Term* (ET) and *Reparans* (RS) as in "[Monday]$_{RD}$ IP [no]$_{ET}$ [Tuesday]$_{RS}$".
- The word lattice correction of repairs divides into two phases of search, given a preprocessed word lattice as input, where word boundaries are classified according to prosodic cues whether they might constitute a word boundary immediately following a reparandum.
- First the word lattice is collapsed to a Part of lattice.
- Second a set of nodes prosodically marked as interruption points is selected.
- For each of these nodes a probabilistic model on POS sequences is used to classify the incoming and outgoing word sequences into RD, ET, and RS. We use a specialized tag set for that step which covers semantic features as well according to their linguistic relevance for the repair phenomenon.
- The last phase — the editing step — monotonically adds new edges to the word lattice spanning the original RD ET RS sequences but being labelled only by the RS label.
- Within the Verbmobil corpus with spontaneous negotiation dialogues about 20% of the utterance exhibit self-repairs. We can currently isolate about 94% of the reparanda correctly given the correct string and irregular boundary, if we restrict ourselves to the self-repairs with less than five words (95% of the corpus).

Control of Several Parsers

- Four different parsing methods are incorporated in the "Integrated Processing" module. All of these produce semantic representations in the same formalism, which can be combined with each other.
  1. The first method is a *deep linguistic HPSG parser*, which is not very robust but produces very detailed descriptions for its inputs.
  2. The second method is a *probabilistic context free grammar LR-parser*, where the grammar and the stochastic parameters are derived from a tree bank. The grammar is supplied with a semantic construction mechanism. However, the representations it produces are usually less detailed than those of the HPSG parser. In many cases where the HPSG fails the probabilistic grammar still produces an interpretation.
  3. The third method is a *chunk parser* based on cascaded finite state automata, producing rough interpretations on analysable fragments of the input.
  4. As a fall-back an HMM-based dialogue-act recognizer is used as the fourth method. This method produces a template intepretation for the dialogue act recognized in each input where special slots like weekdays and clocktimes are filled by additional rules.
- The backbone of the module is an A*-lattice-search with a trigram-based rest cost calculation which guides the search of all the parsing methods through the input lattice.
- Since the increasing robustness of the methods (increasing from HPSG to dialogue-act based analysis) corresponds to their decreasing precision and computational resources needed, the "Integrated Processing" module as a whole can be parametrized to show an anytime behaviour.

Integration of Partial Analyses

- In many cases, no parser will find an analysis spanning the whole input utterance. This may be due to speech recognizer errors, spontaneous speech phenomena which have not been caught earlier, and ungrammaticalities in the utterance itself.
- Although a complete analysis would be preferable, the parser can usually come up with a set of partial analyses in these cases which can often be assembled to yield larger, more meaningful units. This is the basic idea of what we call *robust semantic processing*.
- The task of robust semantic processing then consists of three subtasks:
  1. *store* the partial results in a chart-like data structure (which we call a *VIT Hypothesis Graph* (VHG),
  2. *combine* the partial results on the basis of rules, yielding new entries in the VHG,
  3. *select* a result from the VHG, i.e. a sequence of partial results (or a complete one, if available), if no parser was able to find a spanning analysis in the time available.

- Consider as an example the utterance

  (1) Wir treffen uns in den nächsten zwei Wochen.
  (*We (will) meet during the next two weeks*)

  and assume that the speech recognizer dropped the preposition *in*, as it is just a short word.
- In this case, the parser will analyze the input as two fragments, a sentence (*wir treffen uns*) and a nominal phrase (*den nächsten zwei Wochen*). These two fragments are stored by the robust semantic processing. A rule stating that a temporal NP such as *den nächsten zwei Wochen* can be re-interpreted as a modifier is applied, entering a new edge into the chart. This temporal modifier edge is then combined with the edge for the proposition, yielding a complete and accurate analysis of the complete utterance.

The VIT Format

```
vit( vitID(sid(102,a,ge,0,200,1,ge,y,semantics),  % Segment ID
           [word(das,r0,[1h14]),                  % WHG String
            word(geht,r1,[1h16])]),
     index(1h12,1h11,ih13),                       % Index
     [decl(1h18,hh17),                            % Conditions
      gehen_passen(1h16,ih13),
      pron(1h14,ih15),
      arg1(1h16,ih13,ih15)],
     [ccom_plug(hh17,1h11),                       % Constraints
      in_g(1h18,1h12),
      in_g(1h16,1h11),
      in_g(1h14,1h11),
      leq(1h11,hh17)],
     [s_sort(ih15,object),                        % Sorts
      s_sort(ih13,move_sit)],
     [prontype(ih15,third,demon)],                % Discourse
     [num(ih15,sg),                               % Syntax
      pers(ih15,3),
      gend(ih15,neut)],
     [ta_tense(ih13,pres),                        % Tense and Aspect
      ta_perf(ih13,nonperf),
      ta_mood(ih13,ind)],
     [pros_mood(1h18,prog)]                       % Prosody
)
```

- Theory-independent underspecified semantic representation.
- Also carrying syntactic, prosodic, sortal and discourse information.

Selection of Results

- In search of a good spanning sequence of VITs we select VITs to combine on the basis of a stochastic model on VITs and combine the VITs themselves using symbolic rules.
- In order to have some empirical source of information we designed a special VIT-N-Gramm describing the probability of VIT sequences.
- It is used in combination with some heuristics preferring longer VITs which are more likely to represent a correct analysis. In addition, we give increasing penalties to the less precise models.
- The maximization formula is roughly (neglecting some details) as:

$$V = \max_{V_i^n} \left[\sum_{0 \le i \le n} Log P(V_i) + L(V_i) + W(V_i) \right]$$

  where L stands for a length penalty and W for a penalty for certain sources (parsing methods).
- In first tests some empirically determined length and source weights led to acceptable results. In the future, it is planned to adjust the weights using optimization procedures on ideal outputs.

ICSLP '98, Sydney

{spilker,weber}@fau80.informatik.uni-erlangen.de {cj,worm}@coli.uni-sb.de Tobias.Ruland@mchp.siemens.de