CONTROLLING LIMITED-DOMAIN APPLICATIONS BY PROBABILISTIC SEMANTIC DECODING OF NATURAL SPEECH

Holger Stahl, Johannes Müller, Manfred Lang

Institute for Human-Machine-Communication
Munich University of Technology
Arcisstraße 21, D-80290 Munich, Germany
email: {sta,mue,lg}@mmk.e-technik.tu-muenchen.de

ABSTRACT

The paper describes a speech understanding system, which allows the online control of arbitrary running applications owning a well-defined command interface. A sequential combination of a signal preprocessor, a stochastic-driven one-stage semantic decoder and a rule-based intention decoder is proposed. Following this principle and using the respective algorithms, speech understanding front-ends for the domains 'graphic editor' and 'service robot' could be successfully realized.

Keywords: speech understanding, stochastic knowledge bases, semantic decoding, intention decoding.

1. INTRODUCTION AND SYSTEM OVERVIEW

For expressing a particular intention, a speaker encodes this intention in form of speech. The task of the speech understanding front-end is to record and analyze the acoustic signal and to infer the user's intention.

For the whole speech understanding process, we propose a sequence of a signal preprocessor (generating the observation sequence *O*), a semantic decoder (generating the semantic structure *S*) and an intention decoder (generating the user's intention *I*), as shown in figure 1.

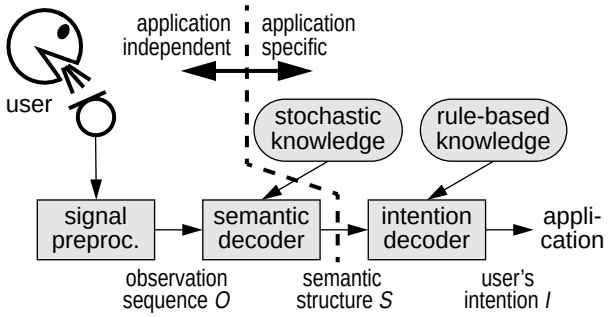

Fig. 1: The speech understanding front-end

We treat the intention *I* as equivalent with a certain program in an application-specific language (e.g. database query language, application command language), for executing the respective user's intention.

To demonstrate our speech understanding approach, we first implemented NASGRA (NAtural Speech understanding GRAphic editor), which allows the user to create, modify or delete three-dimensional objects such as cones, cuboids, cylinders or spheres exclusively by spoken commands. Another working system is the speech understanding service robot ROMAN [3], which is able to carry out everyday jobs like fetching and bringing things:

Fig. 2: Typical scenario from the NASGRA-Domain

Fig. 3: The speech-understanding service robot ROMAN

2. SIGNAL PREPROCESSING

The signal preprocessing module creates 64-dimensional feature vectors in intervals of 10 ms, each of them describing the spectral characteristics of the speech signal contained in a 25 ms-wide window [1]. The time alignment of these feature vectors is called 'observation sequence' *O*. Similar to the task of stochastic word chain decoding (speech recognition), our semantic decoder uses this observation sequence as input for stochastic pattern matching.

3. SEMANTIC DECODING

Unlike other approaches, we use a purely stochastic one-stage decoding algorithm. Since the knowledge for semantic decoding is automatically trained, the algorithm is independent of the domain, the application, and the language.

3.1 Maximum-a-Posteriori Top-Down Decoding

Stochastic methods have proved to be a powerful approach for speech *recognition*, so it is obvious to solve the task of speech *understanding* in a similar way, too. Thus, the prob-

lem of mapping a sequence of observation vectors O to its corresponding semantic structure S can be expressed by maximizing the maximum-a-posteriori (MAP) probability $P(S|O)$:

$$S_E = \underset{S}{\operatorname{argmax}}\, P(S|O)\,. \qquad (1)$$

Applying Bayes' inversion formula and taking into account just the most likely word chain W, we obtain the following classification rule [7][9]:

$$S_E = \underset{S}{\operatorname{argmax}} \max_{W} \left[P(O|W) \cdot P(W|S) \cdot P(S)\right] \qquad (2)$$

The probabilities $P(S)$ and $P(W|S)$ in eq. (2) have to be delivered by the grammar, i.e. the semantic and the syntactic model. The emission probability $P(O|W)$ is calculated by phoneme-based, continuous, speaker independently trained Hidden-Markov-Models (HMMs), which could be adopted from an existing speech recognition system [11].

In contrast to the bottom-up strategy, which is applied for the speech understanding systems of many research groups, we follow a top-down strategy of the decoding process satisfying eq. (2). As shown in chap. 3.3, we developed a very efficient chart-parsing algorithm, which solves semantic decoding in an incremental way.

3.2 The Semantic Structure

The definition of the semantic representation is essential for the architecture of the semantic decoder, and strictly speaking, for the prospect of realizing it at all. Unlike conventional multi-stage approaches for semantic decoding, we do not use a very precise representation in the linguistic sense. Similar to that description used in [7], our semantic structure [9] is a conceptual representation of the utterance with each word of the word chain assigned to one certain concept. However, the semantic structure is not a linear sequence of concepts, but it is hierarchic like a tree, with the power to express complex nested semantic dependencies in the utterance.

A semantic structure S is a set of a finite number N of concepts, called semantic units or shortly *semuns* s_n: $S = \{s_1, s_2, \ldots, s_n, \ldots, s_N\}$. Each semun s_n has a type $t[s_n]$, a value $v[s_n]$ and refers to a certain number $X \geq 1$ (depending on its type $t[s_n]$) of successor semuns $q_1[s_n], \ldots, q_X[s_n] \in \{s_2, \ldots, s_N, \text{blnk}\} \setminus \{s_n\}$. The blank semun 'blnk' forms an exception. It represents a leaf of the tree and has the type $t[\text{blnk}] = \text{blnk}$, no value and no successor. The number of non-blank semuns along the longest branch of S is called 'nesting depth' D. Fig. 4 shows an example of the semantic structure for the user's utterance *"please move the sphere two centimetres downwards"* with $N = 5$ and $D = 3$ within the NASGRA-domain.

To calculate the probability $P(S)$ and the conditional probability $P(W|S)$, the grammar occupies stochastic rules describing only dependencies local to one certain semun $s_n \in S$. Hence, the semantic model contributes rules for fixing the type, value and the types of the successors of

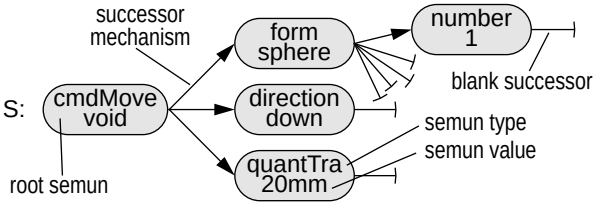

Fig. 4: Graphic depiction of a semantic structure S

each semun, and the syntactic model determines the words, which have to be produced for each semun and their alignment in the word chain. Since the grammar can be seen context-free with some additional extensions taken from HPSG-grammars [8], augmented transition networks [12], and ID/LP-grammars [4], an active chart-parsing algorithm could be applied for realizing the decoder.

3.3 Realization of the Semantic Decoder

The parsing algorithm (see [10]) consists of two layers:

- The *grammar layer* is an active chart-parser which was augmented by a probabilistic dynamic programming mechanism to satisfy the MAP-decoding in eq. (2). The parser is consequently realized in a top-down strategy and incrementally processes the input left to right.
- The *pronunciation layer* contains word hypotheses, each represented by one HMM-trellis, which is processed by Viterbi beam-search.

The grammar layer is tightly coupled to the pronunciation layer, the edges in the chart predict potential word hypotheses by *push-word operations*. After having consumed a certain part of the observation sequence, each word hypothesis acquires a score for matching the respective number of feature vectors. Every time the end state of the respective word HMM is reached, a *pop-word operation* is invoked to extend those edges, which formerly triggered the affiliated *push-word operation*.

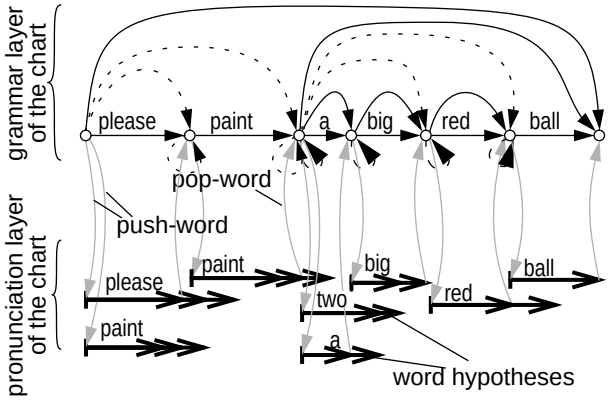

Fig. 5: Integration of word HMMs in an active chart

As shown in [10], the integration of probabilistic active chart-parsing and Viterbi beam-search is possible in a seamless, consistent way. High efficiency is gained through structure sharing both in the active chart and inside the

HMMs, by preserving as many occasions for recombination of hypotheses as possible. Apart from beam-search, further techniques are applied to reduce memory and computation effort, such as histogram pruning and discontinued grammar activity.

4. INTENTION DECODING

Generally, it is not possible to directly use the semantic structure as application input. Hence, it is necessary to transform the semantic structure into an application-specific code, denoted as user's intention *I*. For that purpose, we suggest an application-specific combination of a preprocessor and a compiler:

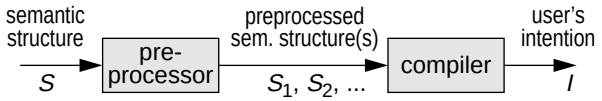

Fig. 6: Block diagram of the intention decoder

The **preprocessor** is necessary to correct inconsistencies in the semantic structure, which occur due to the assumption that each word in the word chain has to be assigned to one single semun. This module provides

- insertion of missing and necessary information (e.g. the semun of the recently modified object),
- deletion of redundant information (e.g. all semuns after an irrelevant "garbage semun"),
- splitting – if possible – a semantic structure *S* into separate semantic structures S_1, S_2, ... each stating an independent and complete command. For example, this *S* corresponding to the word chain *"create a yellow sphere and two cones"* is divided into S_1 and S_2 as shown in fig. 7:

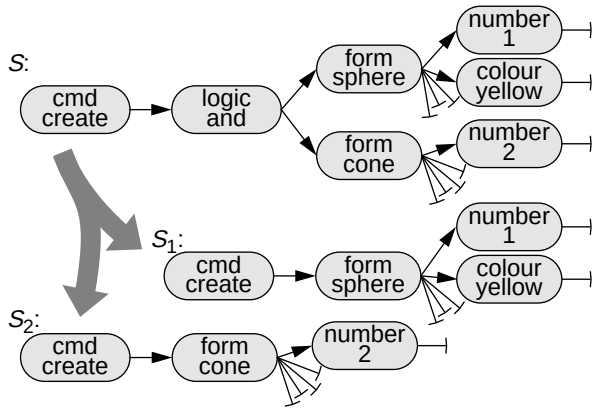

Fig. 7: The preprocessor: Splitting a semantic structure *S*.

The preprocessed semantic structure(s) $S_1, S_2, \ldots$ can be seen as programs in the source language of the compiler, its output is a program in the application-specific language. We call the latter the *user's intention*, since it does not only reflect semantic aspects in the utterance, but it is also influenced by the present status of the application.

The task of the **compiler** therefore is to translate a nested semantic structure tree into a linear code within the application-specific language. Since a main part of a semantic structure's information is held in the topology of the tree, it is not possible to transform each semun individually into one block of the output code. Instead, a *top-up* approach is introduced [2], by propagating context knowledge down into the tree from the root to the leaves and subsequently collecting the information required for generating the intention in the reverse way.

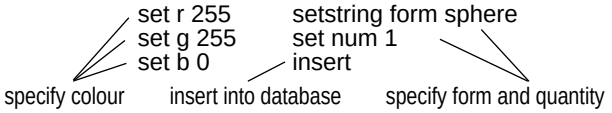

Fig. 8: Intention *I* for the semantic structure S_1 of fig. 7

5. PORTABILITY

One main benefit of the chosen architecture is the portability of the speech understanding front-end, which is enabled by adapting the distributed knowledge about the domain and the application itself.

To provide the stochastic knowledge needed for the semantic decoding process, first a set of semun types and values has to be determined, which covers the semantics of the domain. Then the grammar can be trained iteratively using data collected by a Wizard-of-Oz simulation [5]. Updating the rule-based knowledge needed for intention decoding should be usually possible by easily adding further rules, which are stored in external files. However, if new mechanisms have to be realized or if the intention decoder has to be switched to follow another output syntax, one can't avoid to change the implementation of the semantic preprocessor and the compiler.

6. RECENT RESULTS

6.1 Accuracy Evaluation

For assessing both the formalism of the semantic structure and our one-stage semantic decoder, we compared the semantic accuracy rates obtained with our one-stage system and the conventional two-stage approach. The latter consists of a semantic decoder for text input and a preconnected speech recognizer, delivering the first-best word chain *W*:

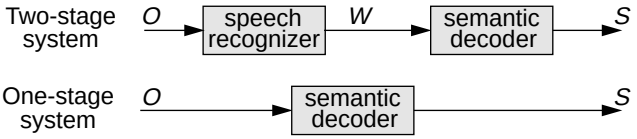

Fig. 9: Comparison of a one-stage and a two-stage decoder

The ***semantic accuracy*** is defined as the rate of correct conversions of the observation sequences *O* into the corresponding semantic structures *S*, evaluated on a certain test corpus. The results shown in tab. 1 were obtained with a test set consisting of 164 utterances out of the NASGRA-domain.

For the first experiments (DISJOINT), the probabilistic knowledge bases have been trained on 1643 utterances disjoint to the test corpus of 164 utterances. Even though Out-of-Vocabulary (OOV) occurences (see [6]) have been excluded, the DISJOINT-test is the more relevant for practical use of the speech understanding system.

The other experiments (RECLASS) were taken out as reclassification, using knowledge bases (i.e. the grammar and the acoustic-phonetic models), which have been trained on a corpus of 1843 utterances containing the 164 utterances as subset. The resulting vocabulary contains 853 words.

training set	one-stage system	two-stage system
DISJOINT	88.4%	86.0%
RECLASS	93.3%	95.9%

Tab. 1: Semantic accuracy rates

In the DISJOINT-case, the one-stage system outperforms the two stage, probably due to the better generalization abilities of our grammar and the missing of consistency problems between the two stages. In the RECLASS-case, the two stage-system does better, because all the utterances in the test set have been seen in the training, so that generalization is rather disadvantageous in this artificial case.

The intention decoder for the NASGRA-domain has been tested with 1843 semantic structures. The ***intention accuracy*** (correct conversion of a semantic structure *S* into its intention *I*) amounts to 97.3% [2]. The respective intention accuracy for the ROMAN-domain even results in 100% [3].

6.2 Trade-Off between Efficiency and Accuracy

Concerning the experiments in chap. 6.1, the parameters of the semantic decoder have been optimized to gain maximum semantic accuracy. However, the accuracy stands in concurrence with memory and computing efficiency. A very sensitive parameter is the pruning-offset, which influences the beam width of the search process.

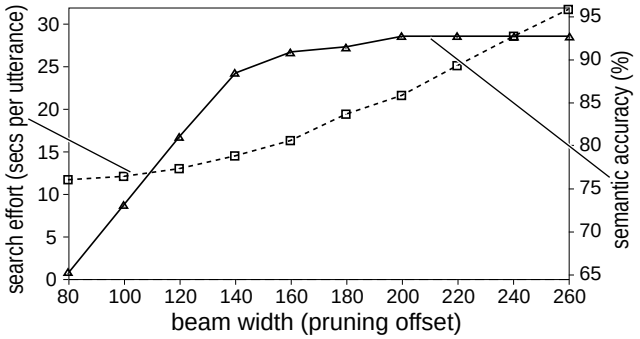

Fig. 10: System performance depending on the beam width

Fig. 10 clearly shows the trade-off between the semantic accuracy and the search effort for the RECLASS-experiment with our one-stage decoder. Maximum accuracy is gained by choosing a pruning offset of 200, taking on average about 22 seconds of CPU workload (on a SUN-*UltraSPARC*, 168 MHz) for semantic decoding.

7. CONCLUSION

A integral system for understanding natural spoken commands was presented, which can be easily adopted to different domains and applications. For the NASGRA-domain, we achieved a semantic accuracy of 88.4% for disjoint test and training corpora. Note, that these results are much better than those reported in [10], since now we incorporated contemporary acoustic models from [11].

REFERENCES

[1] M. Beham: *An Auditorily based Spectral Transformation of Speech Signals*. Proc. EUROSPEECH '91 (Genoa, Italy, 1991), pp. 1437-1440

[2] M. Ebersberger, J. Müller, H. Stahl: *A Compiler-Interpreter-System for Decoding the User's Intention within a Speech Understanding Application*, Proc. KI-96 (Dresden, Germany, 1996), in: Lecture Notes in Artificial Intelligence 1137, Springer, pp. 61-65

[3] C. Fischer, P. Havel, G. Schmid, J. Müller, H. Stahl, M. Lang: *Kommandierung eines Serviceroboters mit natürlicher, gesprochener Sprache*, Proc. 'Autonome Mobile Systeme 1996' (Munich, Germany), Springer, Informatik aktuell, pp. 248-261 (in German)

[4] G. Gazdar, E. Klein, G.K. Pullum, I.A. Sag: *Generalized Phrase Structure Grammar*, Basil Blackwell, Oxford (Great Britain), 1985

[5] J. Müller, H. Stahl: *Collecting and Analyzing Spoken Utterances for a Speech Controlled Application*, Proc. EUROSPEECH '95 (Madrid, Spain), pp. 1437-1440

[6] J. Müller, H. Stahl, M. Lang: *Predicting the Out-of-Vocabulary Rate and the Required Vocabulary Size for Speech Processing Applications*, Proc. ICSLP '96 (Philadelphia, USA, 1996), pp. 1922-1925

[7] R. Pieraccini, E. Levin, E. Vidal: *Learning how to Understand Language*, Proc. EUROSPEECH '93 (Berlin, Germany, 1993), pp. 1407-1412

[8] C. Pollard, I.A. Sag: *Head-Driven Phrase Structure Grammar*, The University of Chicago Press, 1994

[9] H. Stahl, J. Müller: *A Stochastic Grammar for Isolated Representation of Syntactic and Semantic Knowledge*, Proc. EUROSPEECH '95 (Madrid, Spain, 1995), pp. 551-554

[10] H. Stahl, J. Müller, M. Lang: *An Efficient Top-Down Parsing Algorithm for Understanding Speech by Using Stochastic Syntactic and Semantic Models*, Proc. ICASSP-96 (Atlanta, USA, 1996), pp. 397-400

[11] F. Wolfertstetter, G. Ruske: *Structured Markov Models for Speech Recognition*. Proc. ICASSP-95 (Detroit, USA, 1995), pp. 544-547

[12] W.A. Woods: *Transition Network Grammars for Natural Language Analysis*, Communications of the ACM, vol. 13 (1970), no. 10, pp. 591-606

An online version of NASGRA for natural German text input is available on WWW. Please be aware of OOV errors:

http://www.mmk.e-technik.tu-muenchen.de/~mue/nasgra/