A Flexible and Integrated Interface between Speech Recognition, Speech Interpretation and Dialog Management

Robert Lieb, Matthias Thomae, Günther Ruske, Daniel Bobbert [*] *and Frank Althoff* [**]

Institute for Human-Machine Communication, Technische Universität München, Germany
[*]CLT Sprachtechnologie GmbH, Saarbrücken, Germany
[**]BMW Forschung und Technik GmbH, München, Germany
{lie,tho,rus}@mmk.ei.tum.de, bobbert@clt-st.de, Frank.Althoff@bmw.de

Abstract

This paper presents an integrated interface between speech recognition, speech interpretation and dialog control intended for spoken dialog systems coping with natural speech input. During the system design phase the interface co-ordinates corpus acquisition and annotation, grammar development and the construction of stochastic hierarchical language models. During system runtime, it links together speech recognition and interpretation by efficient one-stage decoding of semantic trees, from which semantic content can easily be extracted. To gain robustness, the interface provides a way to interpret semantic confidences estimated during the decoding process. Furthermore, the dialog control can manage dynamic vocabulary and language model parts depending on the dialog context. The suggested interface helps the developer to build up and maintain the speech understanding part of a spoken dialog system in a consistent and flexible way. In addition, the reported experimental results show that information extraction performance can be increased by the presented methods.

1. Introduction

The task of automatic speech understanding in a spoken dialog system is to extract the semantic content from a user utterance, which is passed to the dialog control to generate a suitable system response. In the limited domain of a specific use case, semantic content can be represented by a selection of instances from a finite set of predefined slot-value pairs, also called semantic frame variables (see [1] for a summary of frame-based systems for the ATIS task). Traditionally, the speech understanding problem is tackled in several processing stages. First, a speech recognizer translates the speech signal from acoustic feature vectors into hypothesized words. Then, a semantic parser determines the tree of semantic units that best matches these words. Its grammatical model is designed in a way that the root-leaf paths of a parsed semantic tree correspond to the slot-value pairs which are to be extracted. However, due to the limited power of the grammatical model, it's generally not possible to establish a simple relation between tree paths and slot-value pairs. Therefore the parser needs additional rules that handle slot creation and filling while building up the semantic tree.

A drawback of the multi-stage approach is the fact, that there are separate knowledge sources for speech recognition and interpretation, namely the speech recognizer's language model and the semantic parser's grammatical model. Performance and efficiency of the understanding process suffer from the fact that these knowledge sources cannot be exploited simultaneously. Because the grammatical model cannot be applied during the recognition process, an interface relying only on the best matching word sequence may cause performance losses. A common method to prevent these performance losses is to pass a bunch of possible word hypothesises in form of n-best lists or word graphs which are rescored by the semantic parser (e.g. in [1, 2]).

The presented interface concept, however, solves this problem by the application of our one-stage decoder ODINS [3], that determines the best matching tree of semantic units directly from the speech signal (see [4, 5] for similar approaches). The one-stage decoder uses a so-called hierarchical language model consisting of a hierarchy of weighted transition networks, which is equivalent with the semantic interpretation grammar used for slot filling. In consequence, the search is constrained to the space of interpretable solutions, which guarantees a consistent interface. Using the Viterbi decoding principle, ODINS determines the best possible solution within the limits of the integrated search network hierarchy formed by grammatical, lexical and acoustical model. Efficiency is increased as well, because there's no need for an additional parsing step. Slot-value pairs are extracted in an amortized amount of time from the decoded semantic tree.

In [3], the hierarchical language model was constructed from a corpus of semantic tree annotations. Now, we take the semantic interpretation grammar as primary knowledge source and build up the hierarchical language model by the conversion of grammar rules into corresponding transition networks. Due to the small size of our domain specific data collection, we work with handcrafted semantic interpretation grammars (for sophisticated grammar inference methods based on the EM algorithm, see [1, 4]). To consider corpus statistics, we use a robust parser that automatically annotates semantic trees on the orthographic corpus transcription. This allows the estimation of transition weights by simple counting and smoothing methods, including n-gram statistics (see [6] for a similar approach).

Furthermore, ODINS allows the estimation of semantic tree node confidences [7]. This feature is used to increase robustness against unavoidable recognition and interpretation errors caused by natural speech input. Another important feature is the dynamic management of the hierarchical language model's transition network hierarchy which allows the dialog control to exchange specific model parts, depending on the current dialog context.

The paper is organized as follows: Section 2 describes the system design phase, regarding corpus acquisition, grammar development, automatic annotation and the generation of hierar-

This work was partly funded by the NADIA research project from the Bayerische Motorenwerke (BMW).

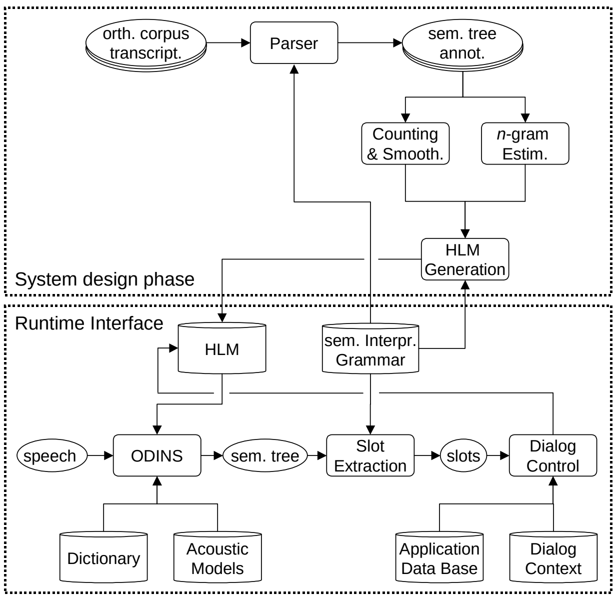

Figure 1: *Diagram depicting system design phase and runtime interface of the suggested automatic speech understanding approach.*

chical language models. Section 3 presents the runtime interface including one-stage decoder, slot filling and dialog control. Section 4 explains how semantic tree node confidences are assigned to slot-value confidences and how the analysis of these values is translated into suitable dialog behavior. Experimental results concerning the information extraction performance of different system setups are given in Section 5.

2. System Design Phase

In order to test the dialog application design as soon as possible, system engineers usually make the effort to conduct usability tests using systems with speech understanding capabilities that are simulated by the aid of human ”wizards”. Even if only a small amount of data can be collected and, in consequence, interpretation grammars have to be crafted by hand, these data collections greatly facilitate their design: The developer is released from being the ”oracle” of possible user utterances and can examine the orthographic corpus transcription to devise suitable grammar rules. To specify interpretation grammars, we use the ABNF format, that allows the compact representation of context-free grammar and slot filling rules by regular expressions containing semantic interpretation tags [8, 9].

In the limited domain of our dialog application, user utterances usually have a low degree of ambiguity. Therefore, we avoid[1] ambiguous grammar rules and resolve the few cases of slot-value pairs with ambiguous interpretations inside the dialog control by initiating a dialog step to ask the user for clarification. Furthermore, semantic interpretation grammars have to be specified without using recursive rule dependencies which are not supported by our one-stage decoder. However, for the description of natural language, this didn’t turned out to be a serious restriction.

[1]Remaining accidental rule ambiguities are resolved by the statistical weighting.

We iteratively improve grammar coverage by the analysis of the semantic trees automatically annotated on the corpus subset used for training. In addition to grammar rules designed for information extraction, we specify filler rules for verbal expressions that occur frequently, but don’t carry any valuable information. To cope with partially ungrammatical utterances, we use a robust parser. It determines the partial parse that includes the minimal number of unmatched words. Although we use unambiguous grammars, ambiguity is still a problem in the presence of ungrammatical utterance sections. Here, it may happen that the robust parser finds more than one semantic tree with the minimal number of unmatched words. Thus, the grammar should be improved until all slot-value pairs in the training corpus subset can unambiguously be extracted. Furthermore, the slot-value pairs extracted on the test corpus subset, serving as reference for the experiments in Section 5, have to be corrected by hand in all cases of wrong solutions, caused by ambiguity or insufficient grammar coverage.

The generation of the hierarchical language model (HLM) for our one-stage decoder ODINS is represented in the upper part of Figure 1. The semantic interpretation grammar is translated to a hierarchy of transition networks by converting the regular expression of each grammar rule into an equivalent finite state automaton, using Lextools [10]. To consider corpus statistics, we use two different methods: Counting and smoothing, as well as n-gram statistics. The first method is used to estimate the transition weights of each subnet in the network hierarchy, independently of the contexts in which the corresponding grammar rule can be applied. The transition weights are estimated from the number of traversals of each transition in each subnet, counted while walking through the semantic trees that have been automatically annotated on the training corpus subset. To take into account unused parts of the semantic interpretation grammar and to correct unreliable counts caused by data sparsity, we smooth the transition weights by Good-Turing discounting. Grammar rules describing alternatives, that are apriori equally probable, have to be excluded from the counting and smoothing process and are represented by subnets with uniformly distributed transition weights. The second method employs n-gram models for grammar rules that consist of a ”Kleene closure” expressed by the ”star” or ”plus” operator in regular expression syntax. Being equivalent with an unweighted zero-gram, this construction can be replaced by an n-gram model which is estimated over the collection of sequences produced by the corresponding grammar rule, while parsing the training corpus subset. Up to now, we use this method to estimate a bigram for the grammar’s head rule which defines a Kleene closure over all main rules. To compute n-gram models with back-off smoothing, and to convert them into finite state automata we use the SRILM toolkit [11].

3. Runtime Interface

The runtime interface is depicted in the lower part of Figure 1. While the user is talking to the dialog system, ODINS determines the best matching semantic tree by time synchronous Viterbi search. As stated in the last section, its hierarchical language model ensures that the decoded semantic tree follows the semantic interpretation grammar. Therefore a constrained parse along the decoded semantic tree is sufficient to retrieve the slot-value pairs. This is done by evaluating the semantic interpretation tags which are encountered while matching the grammar rule definitions with the decoded semantic tree. Semantic interpretation tags contain scripting commands that handle slot creation and filling [9]. Each encountered semantic tag corresponds to a specific scope of consecutive tree siblings. The contained slot filling commands may refer to subordinate slots that already have been created in these sibling nodes. This allows the concatenation of slot names to form nested data structures, as well as the arbitrary combination of values using scripting facilities, like arithmetical operators or string processing. While walking upwards the semantic tree, this mechanism provides a flexible way to extract information and to translate it into the desired format. Figure 2 shows an example from our German flight information application domain. It illustrates the extraction of two slot-value pairs from a decoded semantic tree that contains a flight code consisting of airline code and flight number. To the right hand side of every non-terminal semantic tree node one can see the matching part of the corresponding grammar rule that contains the evaluated semantic interpretation tags[2] (in curly braces). The resulting slot-value pairs are taken from the root of the semantic tree. If multiple instances are extracted for the same slot (e.g. when the user makes a self-correction), the corresponding value is simply overwritten in the order of occurrence. The resulting collection of slot-value pairs is passed to the dialog control that determines the next dialog step.

While the user is prompted for the next response, the dialog control can modify the hierarchical language model depending on the current dialog context. The dialog control uses this interface to update subnets representing specific content from the dialog application data base. This allows the dynamic selection of data base content into the search space, depending on information extracted in earlier dialog steps, as well as the consideration of data base content that is not known during design time. For this purpose ODINS provides an interface for the dynamic management of transition network hierarchies. Subnets are managed in resource collections, which can be selected in a specific order to build up the network hierarchy for the next recognition run. Vocabulary can be loaded dynamically from pronunciation dictionaries, which are automatically generated by grapheme to phoneme translation[3].

4. Interpretation of Confidences

Interpretation errors are caused either by imperfect modeling, concerning grammar coverage and acoustic-phonetic models, or by the user itself in the case of out-of-domain utterances. Dialog quality can benefit a great deal from the robust detection of these errors, because misguided as well as unnecessary dialog steps can be avoided. For this purpose we use the method presented in [7] to estimate confidences for semantic tree nodes. During slot-value pair extraction, semantic tree node confidences are translated into corresponding slot and value confidences, using a rule-based policy described in the next paragraph. The dialog control exploits slot and value confidences in the following way: If the slot confidence is below a certain threshold, the whole slot is discarded. If the slot confidence holds, but the value confidence is below the threshold, the dialog control may initiate a step to clarify the slot content. If slot and value confidences are both accepted, the dialog control can safely rely on the slot-value pair when making decisions that determine the subsequent dialog steps.

[2]"$" denotes a temporary slot that can be accessed from higher-ranked semantic interpretation tags

[3]for our German vocabulary we use the freely available tool txt2pho [12]

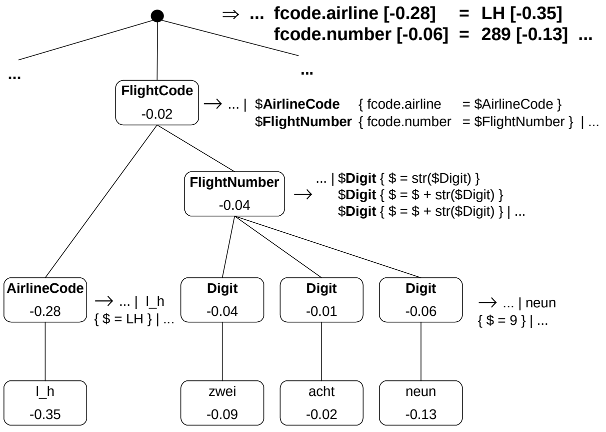
⇒ ... fcode.airline [-0.28] = LH [-0.35]
fcode.number [-0.06] = 289 [-0.13] ...
...
...
FlightCode -0.02
→ ... | $AirlineCode { fcode.airline = $AirlineCode }
$FlightNumber { fcode.number = $FlightNumber } | ...
FlightNumber -0.04
→ ... | $Digit { $ = str($Digit) }
$Digit { $ = $ + str($Digit) }
$Digit { $ = $ + str($Digit) } | ...
AirlineCode -0.28
→ ... | l_h { $ = LH } | ...
Digit -0.04
Digit -0.01
Digit -0.06
→ ... | neun { $ = 9 } | ...
l_h -0.35
zwei -0.09
acht -0.02
neun -0.13

Figure 2: *Example illustrating the extraction of slot-value pairs and corresponding confidences from a decoded semantic tree that contains a flight code.*

The confidence of a semantic tree node, that corresponds either to a non-terminal grammar rule or to a terminal word, is an estimation of the posterior probability of it's occurrence in the decoded semantic tree. Currently, we use the following rule-based policy to translate semantic tree node confidences into slot and value confidences: For the case that a slot is created and filled without referring to subordinate slots, the slot confidence is set to the confidence of the current tree node, in which the slot is created. The value confidence is calculated by taking the minimum over the confidences belonging to the sibling nodes in the scope of the semantic tag, that contains the slot creation command. If, however, subordinate slots are involved, the confidence of the newly created slot is set to the minimum over the confidence of the current tree node and the slot confidences of the subordinate slots. The value confidence of the new slot is set to the minimum of the value confidences of the subordinate slots. We chose the minimum operator to combine the negative confidence values, since they are estimations of logarithmized posterior probabilities. Figure 2 shows the extracted slot and value confidences (indicated in square brackets) that result from the depicted semantic tree node confidences when applying the rule-based policy as specified above. In the case that the value of an already existing slot instance should be overwritten, this is only done if the slot confidence of the new instance is above the confidence threshold used for discarding slots.

5. Experimental Results

To measure the performance of information extraction we use the slot error rate SER [13]

$$SER = \frac{N_S + N_I + N_D}{N_C + N_S + N_D}$$

which takes into account the number of correct (N_C), substituted (N_S), inserted (N_I) and deleted (N_D) slots. A slot is counted as correct, if it can be found in the hypothesis and the reference and if the values are equal. If the values are not the same, the slot is counted as substituted. An insertion is counted if the slot occurs only in the hypothesis, a deletion if it occurs only in the reference. Sequence aligning is not necessary, because the slots extracted from a specific user utterance are unique and can be processed independently of the order of their

%	WER	SER	CER	CER_{BL}
two-stage	18.6	12.0	-	-
one-stage	19.5	10.2	4.5	6.0

Table 1: *Word (WER) and slot error rate (SER) of two-stage and one-stage system setup, as well as slot confidence error rates CER and CER_{BL}.*

extraction. To incorporate the confidence estimation, we discard slots having a slot confidence below a specific threshold which is empirically adjusted by cross-validation experiments. If the confidence estimation performs well, this decreases the slot error rate by the rejection of incorrectly inserted slots. Furthermore, we evaluate the slot confidence estimation by counting the number of false accepted (N_{FA}) and false rejected (N_{FR}) slots and comparing the confidence error rate CER with its base line CER_{BL}:

$$CER = \frac{N_{FA} + N_{FR}}{N_C + N_S + N_I}$$
$$CER_{BL} = \frac{N_I}{N_C + N_S + N_I}$$

Experiments were carried out within our application domain, which is a German airport information system. This system allows the user to query information like arrival or departure times, flight codes, gates and parking sites. The training corpus subset includes about 3500 utterances of 55 speakers, the test corpus subset about 450 utterances of 12 speakers. Currently, the semantic interpretation grammar covers about 80 rules and 20 different slot types. The task's vocabulary includes about 500 different words. The percentage of words in the test set that are not covered by the semantic interpretation grammar is about 11% (which is a lower bound for the word error rate). The acoustic modeling is done by speaker-independent tied intra-word triphone HMMs with about 25k Gaussian mixture components, as described in [3]. Table 1 shows preliminary results regarding two different system setups: The first one is a two-stage interpretation system, running ODINS with a bigram class language model to determine the best matching word sequence. The word sequence is passed to the robust parser, which applies the interpretation grammar to extract the slot-value pairs. The second system setup uses ODINS with a hierarchical language model, that is constructed as explained in Section 2. Slot-value pairs and corresponding confidences are extracted from the decoded semantic tree following the method presented in Section 3 and 4. The results show that information extraction performance can be improved by one-stage decoding and confidence estimation, although the word error rate is not improved in comparison to the two-stage setup. Further, the slot confidence error falls below its base line indicating that slot confidences can be used to find out incorrectly inserted slots.

6. Conclusions

In this paper we proposed an interface that tightly couples speech recognition, speech interpretation and dialog control. This is achieved by the construction of a hierarchical language model that includes the semantic interpretation grammar and corpus statistics. Our one-stage decoder ODINS uses the hierarchical language model to determine the best matching semantic tree directly from the speech signal. By matching the semantic interpretation grammar with the decoded semantic tree, slot-value pairs can easily be extracted, including separate confidences for slot and value. The reported results show that information extraction performance and robustness against interpretation errors can be improved using this approach. From the system engineer's point of view, the tight integration of the involved knowledge sources prevents errors and inconsistencies and thus greatly facilitates the development and the administration of the speech understanding part of a spoken dialog system.

7. References

[1] Y. He and S. Young, "A Data-Driven Languag Understanding System," in *Proc. ASRU*, St. Thomas, U.S. Virgin Islands, November 2003.

[2] K. Hacioglu and W. Ward, "A Word Graph Interface for a Flexible Concept Based Speech Understanding Framework," in *Proc. Eurospeech*, Aalborg, Denmark, September 2001.

[3] M. Thomae, T. Fabian, R. Lieb, and G. Ruske, "A One-Stage Decoder for Interpretation of Natural Speech," in *Proc. NLP-KE'03*. Beijing, China: IEEE, October 2003. [Online]. Available: http://www.thomae-privat.de/publications/nlpke2003.pdf

[4] A. Acero and Y. Wang, "A Semantically Structured Language Model," in *Special Workshop in Maui (SWIM)*, Hawaii, January 2004.

[5] C. Fügen, H. Holzapfel, and A. Waibel, "Tight Coupling of Speech Recognition and Dialog Management - Dialog-Context Dependent Grammar Weighting for Speech Recognition," in *Proc. ICSLP*, Jeju Island, Korea, October 2004.

[6] K. Hacioglu and W. Ward, "Dialog-Context Dependent Language Modelling Combining N-Grams And Stochastic Context-Free Grammars," in *Proc. ICASSP*, Salt Lake City, Utah, May 2001.

[7] R. Lieb, T. Fabian, G. Ruske, and M. Thomae, "Estimation of Semantic Confidences on Lattice Hierarchies," in *Proc. ICSLP*, Jeju Island, Korea, October 2004.

[8] W3C Voice Browser Working Group, "Speech Recognition Grammar Specification," 2004. [Online]. Available: http://www.w3.org/TR/speech-grammar/

[9] ——, "Semantic Interpretation for Speech Recognition," 2004. [Online]. Available: http://www.w3.org/TR/semantic-interpretation/

[10] R. Sproat, "Lextools: a toolkit for finite-state linguistic analysis." [Online]. Available: http://www.research.att.com/sw/tools/lextools/synth.pdf

[11] A. Stolcke, "SRILM - An Extensible Language Modeling Toolkit," in *Proc. ICSLP*, Denver, Colorado, USA, September 2002.

[12] T. Portele, J. Krämer, and D. Stock, "Symbolverarbeitung im Sprachsynthesesystem HADIFIX," in *6. Konferenz Elektronische Sprachsignalverarbeitung*, Wolfenbüttel, Germany, 1995.

[13] J. Makhoul, F. Kubala, R. Schartz, and R. Weischedel, "Performance Measures for Information Extraction," in *DARPA Broadcast News Workshop*, 1999.