Empowering Knowledge Based Speech Understanding through Statistics

J. Fischer, J. Haas, E. Nöth, H. Niemann, F. Deinzer

University of Erlangen–Nuremberg
Chair for Pattern Recognition
Martensstraße 3, 91058 Erlangen, Germany
(fischerj,haas,noeth,niemann,fkdeinze)@informatik.uni-erlangen.de

ABSTRACT

In this paper we present an innovative approach to speech understanding which is based on a fine–grained knowledge representation automatically compiled from a semantic network and on iterative optimization. Besides allowing an efficient exploitation of parallelism, any–time capability is provided since after each iteration step a (sub–)optimal solution is always available. We apply this approach to a real–world task, which is a dialog system able to answer queries about the German train timetable. In order to speed up the search for the best interpretation of an utterance we make use of statistical methods, e.g. neural networks, *n*–grams, and classification trees, which are trained on application relevant utterances collected over the public telephone network. At the moment the real–time factor for interpreting the initial user's utterance is 0.7.

1. INTRODUCTION

In order to make use of automatic speech understanding systems in real–world applications, those systems have to be featured with real–time and any–time capabilities. Furthermore, they should be easily adaptable to new application domains, and should be able to integrate speech with other sources of information, for instance gestures, which would increase the accuracy and naturalness of human computer interaction. The combination of statistical methods with knowledge based methods and parallel processing seems to be a promising means to achieve some of the capabilities mentioned above.

Whereas the use of statistics on the level of speech recognition is state–of–the art, speech understanding is usually based on parsing algorithms and context–free grammars. In the past few years, statistical methods were gaining more and more importance also on the level of understanding (see for instance [1, 2]). A variety of parallel algorithms for problems from data–driven processing have been developed. In contrast, parallel symbolic processing is much less investigated, although some major problems of the field, like e.g. parallel knowledge representation [3], are discussed in the literature. In our approach, we combine knowledge based speech understanding using semantic networks with statistical methods to speed–up the search for the best interpretation. The semantic network provides an integrative knowledge representation formalism for speech and image understanding. By using statistical methods, a fast adaptability to new application domains is enabled, provided that a corpus of data is available. Furthermore, a control algorithm based on parallel iterative optimization is used, providing the desired any–time and real–time behaviour.

2. THE DIALOG SYSTEM EVAR

As a framework for our approach we use the dialog system EVAR [4] which answers queries about the German train timetable. The linguistic knowledge representation of EVAR is arranged in 5 *levels of abstraction*: The *Word–hypotheses* level represents the interface between speech recognition and speech understanding; on *Syntax* level syntactic constituents are represented; the *Semantic* level is used to model verb and noun frames with their deep cases for task independent interpretation; on *Pragmatic* level, semantic information is interpreted in a task–specific context; the *Dialog* level models possible sequences of dialog acts. Figure 1 shows an excerpt of the semantic network representing linguistic knowledge about the destination of a trip.

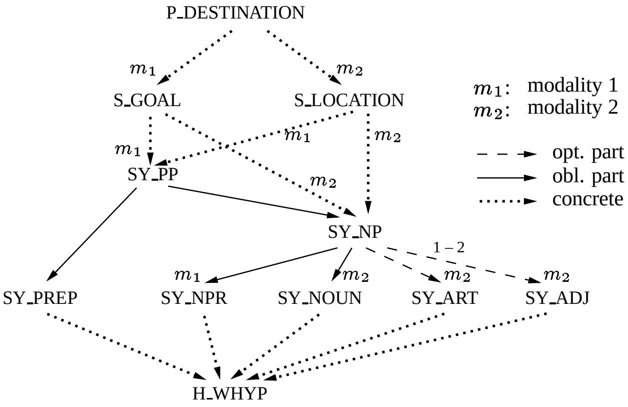

Figure 1: Excerpt of the knowledge base of EVAR.

The knowledge itself is represented using the semantic network formalism of ERNEST (ERlanger NEtzwerk SySTem) [5]. Knowledge about general terms, events, etc. is represented in *concepts* C (e.g. SY_NOUN represents knowledge about nouns), actual realizations of a concept are represented by *instances* $I(C)$ (e.g. the noun "train" is represented by an instance $I(\text{SY\_NOUN})$). Relations between the concepts (nodes) are established by *part*–, *concrete*–, and *specialization–links*.

The main components of a concept C itself are, besides its *parts* P and *concretes* K, a set of *attributes* A and *structural relations* S. Each of them references a function F which computes the value of the corresponding attribute and a measure of the degree of fulfillment of the relation. Since there may be different possibilities for the actual realizations of a concept and in order to allow a compact knowledge representation, *modalities* H_l are introduced with the implication that each individual modality $H_l^{(k)}$ may define the concept C_k. In Figure 1, for example, a Noun Phrase (SY_NP) can be defined by:

- Modality 1: obligatory part: *Proper Noun* (e.g. *"Berlin"*)
- Modality 2: obligatory part: *Noun* optional parts: *Article, Adjective* (e.g. *"the next train"*)

For the computation of an instance $I(C)$, instances for all of its parts and concretes, and values for all of its attributes and relations have to be computed. Due to word recognition errors and ambiguities in the knowledge base (arising, for example, from the various modalities), a *confidence measure* $\boldsymbol{G}$ is available, which computes the degree of confidence of $I(C)$ and its expected contribution to the success of the analysis. The ultimate goal of the analysis itself is represented by one or more *goal concepts* C_{g_i}, which represent the required overall symbolic description of the initial segmentation. Subsequently, an interpretation of the initial word–hypotheses is given by an instance of a goal concept $I(C_{g_i})$. Task of the control–algorithm is thus to search for an *optimal instance* $I^*(C_{g_i})$ with highest confidence value. (since in our application domain we have only one goal concept, we will refer only to C_g in the rest of the text).

3. THE ITERATIVE CONTROL

The control algorithm we employ (cf. [6]) treats the search for an optimal interpretation as a *combinatorial* optimization problem and solves it by means of *iterative* optimization methods, e.g. simulated annealing, stochastic relaxation, and genetic algorithms. Since in principle each word–hypothesis can be "attached" to each primitive attribute of a concept on word–hypotheses level (cf. Section 2), and since an instance $I(C)$ can be computed for each modality of C, one can say that the computation of $I(C_g)$ and its confidence value are completely determined by

- the choice $(A_i, O_j^{(i)}), i = 1, \ldots, m$ of a word–hypothesis O_j for each primitive attribute A_i of concepts representing the interface to the initial segmentation and
- the choice $(C_k, H_l^{(k)}), k = 1, \ldots, n$ of a modality H_l for each instance of a concept C_k which has more than one modality.

This allows us to characterize a *current state of analysis* by way of the following vector:

$$\boldsymbol{r}_c = [(A_i, O_j^{(i)}); (C_k, H_l^{(k)}) \mid i = 1, \ldots, m \; ; \; k = 1, \ldots, n] \; ,$$

and to compute $I(C_g)$ subject to $\boldsymbol{r}_c$. Each value assignment to the parameters of this vector reflects exactly one out of the finite set of possible interpretations. The task of the control algorithm can thus be defined as to find an optimal state of analysis $\boldsymbol{r}_c^*$ which consequently reflects an optimal instance $I^*(C_g)$. Therefore, a cost function ϕ is introduced and $\boldsymbol{r}_c$ is treated as the current state of a combinatorial optimization problem. In each iteration step i, an actual state $\boldsymbol{r}_i$ is chosen out of $\boldsymbol{r}_{i-1}$ and an instance $I_{r_i}(C_g)$ and its corresponding costs ϕ_{r_i} are computed for $\boldsymbol{r}_i$. Iterations are performed until an optimal solution is found or until no more processing time is available. The determination of how to carefully choose an *initial* state $\boldsymbol{r}_0$ is topic of Section 4. The iterative strategy provides the system with *any–time* capability, since after each iteration step a (sub–)optimal solution is always available.

An efficient exploitation of *parallelism* is enabled by compiling the concept–centered semantic network into a fine–grained task–graph, the so–called *attribute network* (cf. Figure 2). This is done by splitting up each concept to be considered for the computation of an instance $I(C_g)$ into its *subconceptual* entities (attributes, relations, and confidence measures) and by determining the dependencies between them. For the computation of an instance $I(C_g)$, a state of analysis is chosen, mapped onto the attribute network, and all nodes of the attribute network are processed in a single bottom–up step. This step corresponds to a single iteration in the iterative optimization. The attribute network can be mapped to a multiprocessor system for parallel processing (*parallel bottom–up instantiation*). Furthermore, several competing instances of goal concepts (i.e., several $I_{r_i}(C_g)$ for different $\boldsymbol{r_i}'s$) can be computed in parallel (*parallel search*), e.g. on a local network of heterogeneous workstations (WS$_1$, ..., WS$_\mathrm{p}$ in Figure 2).

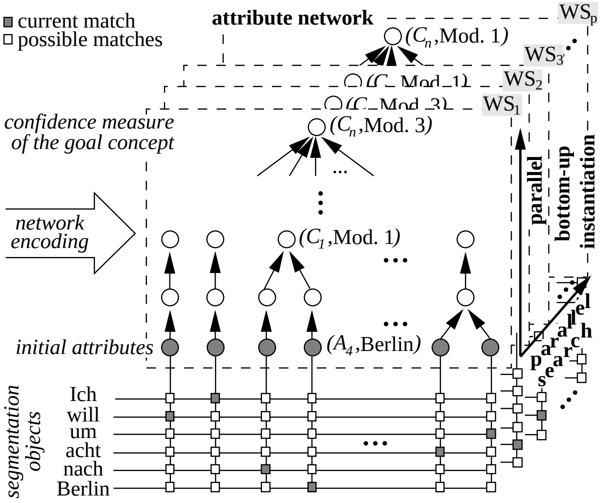

Figure 2: Scheme of the parallel iterative control algorithm.

Figure 2 shows an example for the analysis of the word–chain "*Ich will um acht nach Berlin*" ("*I want to go to Berlin at eight o'clock*", in a word to word translation "I want at eight to Berlin"). The current state of analysis for which an instance is computed bottom–up on WS$_1$ is

$$\boldsymbol{r}_{c_{WS_1}} = [(A_1, \text{will}), \ldots, (A_4, \text{Berlin}), \ldots, (A_m, \text{um}); \\ (C_1, \text{Mod.1}), \ldots, (C_n, \text{Mod.3})] \; ,$$

with C_n being the goal concept C_g. The current state of analysis on WS$_2$ differs from that on WS$_1$ at least by the current word–hypothesis assigned to A_m ("nach"); the current state of analysis on WS$_\mathrm{p}$ differs from that on WS$_1$ and WS$_2$ at least by the Modality assigned to the goal concept, indicating that competing instances are computed for different states of analysis on the various workstations.

4. STATISTICAL INITIALIZATION

If an initial state of analysis $\boldsymbol{r}_0$ is chosen at random, it often occurs that the algorithm starts searching quite far from the optimal solution and thus needs a lot of iterations (and hence a great deal of processing time) until the optimum is found. Thus, in order to find the best interpretation in an efficient manner, the careful choice of an initial state of analysis is indispensable.

For example, if a Noun Phrase for the words "*the next train*" is to be instantiated, and modality 1 is chosen to compute the corresponding I(SY_NP) (cf. Section 2), the confidence value of this instance will be quite low or "invalid". So, the basic idea is to make a prediction of the optimal state of analysis by means of the given word–chain $\boldsymbol{w}$ (which can be the best word–chain computed from the word–graph). This would lead to the prediction of modality 2 for computing the demanded instance in the above example. The initialization problem is thus viewed as a classification problem

$$\boldsymbol{w} \mapsto (\Omega_{O_j^{(1)}}, \ldots, \Omega_{O_j^{(m)}}, \Omega_{H_l^{(1)}}, \ldots, \Omega_{H_l^{(n)}})^T \mapsto \boldsymbol{r}_0 \qquad (1)$$

which we solve by means of neural networks, n–grams, and classification trees. Since the number of classes for each Attribute A_i varies for each $\boldsymbol{w}$ to be analyzed (according to the number of concurring hypotheses for each A_i), we will only consider the initialization concerning the assignment of a modality $H_l^{(k)}$ for each ambiguous C_k, $k = 1, \ldots, n$ (in our application, $n = 223$; each C_k has an average of 2.4 concurring modalities). For more details cf. [7].

4.1. Initialization by TDNNs

In order to represent the temporal information contained in w we make use of Time Delay Neural Networks (TDNNs) [8], which are Artificial Neural Networks with multiple *time delay* links between successive layers. This allows us, furthermore, to process word–chains of any length.

A widely used approach if working with statistical methods and large vocabularies is to cluster the words into categories. For example, in order to classify "*at* DIGIT *o'clock*" as being a time expression, it is not important if DIGIT is an "eight" or "ten". Since not only syntactic information is important for our application, but also semantic information, we use a method for categorization which allows to represent several features of a single word in one category. Therefore, we define the category itself as consisting of a *basic part* and a *special part*. The basic part contains syntactic information about the word, the special part semantic information (it may also consist of more special syntactic information, if necessary). Consider, for example, the word "Intercity". It's basic category is NOUN, it's special category is TRAIN. The resulting category system is not disjoint. With some slight modifications it is also used for the other methods (cf. Section 4.2).

Each category is represented as a binary vector $\boldsymbol{i}$, which serves as *input* for the TDNN. Each binary vector consists, according to the categories, of a binary part representing the basis, and a binary part representing the special information:

$$w_i \;\mapsto\; \underset{\textit{category c}}{\boxed{\text{basis} \mid \text{special}}} \;\mapsto\; \underset{\textit{input vector } \boldsymbol{i}}{\boxed{(0\,0\,0\,1\,0\ldots0 \mid 0\,1\,0\,0\ldots0)}}$$

If a word w_i is part of two or more categories (recall that the category system is not disjoint), the appropriate input vector results from a combination of the binary representation of these categories. The structure of the input vectors is chosen in such a way that the combination results in a unique representation of the categories. The *output* $\boldsymbol{o}$ of the TDNN is also a binary vector which is mapped to the initial state of analysis vector $\boldsymbol{r_0}$. For example, assuming that the concept C_1 has 3, C_i has 4, and C_n has 3 competing modalities, the output vector will be

$$\boldsymbol{o} = (\mathcal{O}_1, \ldots, \mathcal{O}_i, \ldots, \mathcal{O}_n) = (\underbrace{0,1,0}_{C_1}, \ldots, \underbrace{0,1,0,0}_{C_i}, \ldots, \underbrace{0,0,1}_{C_n})$$

if modality 2 is assigned to C_1 and C_i, and modality 3 is assigned to C_n. The number of output nodes of the TDNN results from the sum of the amount of competing modalities for each ambiguous concept (which, in our application, is a total of 542 nodes).

4.2. Initialization by n-grams and Classification Trees

Other approaches to the classification problem formalized in equation (1) where we try to find a corresponding class for a spoken (or recognized) word chain $\boldsymbol{w}$ are semantic classification trees (SCTs) [9] and n-gram language models [10]. As the number of nodes to be initialized is very high and it is not feasible to train a classifier based on SCTs or n-grams for each node, we reduce the task for SCTs and n-grams to some very important nodes in the attribute network. In our experiments we concentrate on the initialization of the verb frame which gives us ten different classes. We use a system with 71 disjoint categories which comprise syntactic, semantic and pragmatic information.

When using SCTs we have to specify in advance the set of possible questions which mainly check the appearance of words at special positions in the word chain. In addition we have to define a criterion to be optimized during training iterations. We decided for the Gini criterion (cf. [9]) to scale the impurity of sets generated by the tree. As output we get the number of the class the input word chain is classified on, i.e. the verb frame we have to choose for our initial state of analysis vector.

For the classification of word chains with n-gram language models we suggest the following framework: For each class to be distinguished we train a separate n-gram on those sentences from the training set corresponding to the according class. In the case of classifying verb frames we train ten different language models. For a new sentence to be analyzed we compute the different probability scores of the models and decide for that with highest probability.

5. EXPERIMENTAL RESULTS

In this section we show the classification ability of the statistical methods and we evaluate the performance of the system empowered through statistics in comparison to the system without statistics. The latter was done with respect to the amount of iterations necessary to find an optimal interpretation. The quality of an interpretation is measured according to the amount of pragmatic information units found. These are units the system needs to know in order to access the database and retrieve the requested information.

For our experiments we use the transliteration of 6 712 *spontaneous* utterances selected from the EVAR–Spontan (cf. [11]) corpus (simulating a 100% word–accuracy). These spontaneous utterances were collected via the public telephone network. Training is done with 5 767 of these utterances, for the tests we use the remaining 945 ones. The labeling of the test data for the training of the TDNN was done automatically by the system itself, using a heuristic initialization which consists of a small set of rules working on the incoming word–chain. These rules were developed on a corpus of about 140 *thought up* sentences of the same domain. One of these rules, for example, asks whether the words *"to go"* appear in the word–chain and if so, the verb frame to go is adjusted. By this heuristic initialization, we improve (on the 140 read sentences) the analysis in the first iteration step from 56% to 89% correctly analyzed pragmatic intentions. Per sentence we performed 25 iterations (because of time limitation; we plan to carry out a more accurate labeling by performing more iterations) and each sentence was, thus, labeled with a state of analysis vector which reflects the interpretation after these 25 iterations. The task of the TDNN is to learn this state of analysis and to predict it for each new word–chain to be analyzed, so we can save computing time (the time to perform 25 iterations) without loosing accuracy. For the training of the SCT and n-gram, whose task is only to adjust the verb frame (cf. Section 4.2), the labeling with verb frames was done semi–automatically: a small set of rules classified the word–chain in a first step and in a second step we checked and corrected the results.

The trained TDNN consists of 42 input nodes (we enter two words a time) and 542 output nodes (cf. Section 4.1). There are five hidden layers, each consisting of 70 nodes. The network thus is able to consider a context of 16 words.

SCT	n-gram	TDNN
97.9%	92.2%	98.2%

Table 1: Prediction results for the verb frame with SCT and n-gram and for the state of analysis after 25 iterations with TDNN.

In Table 1 the percentage of correctly classified verb frames using the SCT and n-gram and the percentage of correctly classified modalities for the 223 ambiguous concepts using the TDNN (with respect to the state of analysis after 25 iterations as explained above) are listed. As one can see, the SCT and the n-gram are able to predict the verb frame to be adjusted for a given word–chain and the TDNN is a very good oracle for the prediction of the state of analysis resulting after 25 iterations. The attribute network for the evaluation of the overall system's improvement consists of about 10 000 nodes. The goal concept is P_CONNECTION_INFO which models the user's first utterance in a dialog for train timetable information. The optimization method used is *stochastic relaxation*. In Table 2 the number of correctly analyzed pragmatic units (i.e. the number of pragmatic units found) by computing n iterations on five processors is shown. It should be noted that the bottom–up processing is done sequentially, and the parallel processing on control level (cf. Section 3) is simulated on a single processor. We are at the moment working on the implementation of the parallel processing on several workstations.

	spoken word chain		
# of Iterations	1	5	25
Heuristic	78.8	79.0	79.7
TDNN	80.7	82.3	82.6
TDNN + n-gram	79.5	80.8	80.8
TDNN + SCT	81.5	83.1	83.3

Table 2: Percentage of correctly analyzed pragmatic units.

The starting point for the statistically initialized analysis is a state which conforms to the heuristically initialized analysis after 25 iterations. It can be seen that the more iterations performed, the more information is found and accuracy increases. Additionally, since the computing time for the statistical initialization ($\approx$ 0.4 seconds) is about the same as for performing two iterations (one iteration needs $\approx$ 0.2 seconds on a 9000/735 HP–Workstation), we drastically accelerate the analysis. At the moment, the system's real–time factor for the interpretation of the initial user's utterance is 0.7, performing 5 iterations on 5 processors (the average time per utterance in our corpus is 3 seconds). We achieve the best result by employing the TDNN for the initialization of the modalities combined with a verb frame initialization computed by the SCT. This can be explained by the fact that verb frames in German often are defined by key–words which can have long distance dependencies between them. These can be modeled with the SCT but not with the n-gram (cf. also Table 1). Furthermore, the verb frame initialization by the SCT can intercept errors made by the TDNN, and thus lead to an overall better result.

6. CONCLUSION AND FUTURE WORK

In this paper we proposed the employment of statistical methods to initialize the state of analysis of a knowledge based speech understanding and dialog system with any–time and real–time capabilities. Experimental results showed the success of the approach. Meanwhile we have improved heuristic rules through which the system can achieve significant better results than those in Table 2. The experiments with statistical initialization will be rerun using these new heuristics. Furthermore, we will concentrate on the processing of word hypotheses graphs, and on the further improvement of processing time and convergence speed.

7. REFERENCES

1. R. Pieraccini and E. Levin. A Learning Approach to Natural Language Understanding. In *NATO-ASI, New Advances & Trends in Speech Recognition and Coding*, volume 1, pages 261–279, Bubion (Granada), Spain, 1993.
2. H. Stahl, J. Müller, and M. Lang. An Efficient Top–down Parsing Algorithm for Understanding Speech by using Stochastic Syntactic and Semantic Models. In *Proc. Int. Conference on Acoustics, Speech, and Signal Processing*, pages 397–400, Atlanta, 1996.
3. M. Evett, J. Hendler, and L. Spector. Parallel Knowledge Representation on the Connection Machine. *Journal of Parallel and Distributed Computing*, 22(2):168–184, 1994.
4. M. Mast, F. Kummert, U. Ehrlich, G. Fink, T. Kuhn, H. Niemann, and G. Sagerer. A Speech Understanding and Dialog System with a Homogeneous Linguistic Knowledge Base. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 16(2):179–194, 1994.
5. H. Niemann, G. Sagerer, S. Schröder, and F. Kummert. ERNEST: A Semantic Network System for Pattern Understanding. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 9:883–905, 1990.
6. V. Fischer and H. Niemann. A Parallel Any–time Control Algorithm for Image Understanding. In *Proceedings of the 13^{th} International Conference on Pattern Recognition (ICPR)*, Wien, October 1996. IEEE Computer Society Press.
7. F. Deinzer. Untersuchung stochastischer Ansätze zur Zustandsinitialisierung eines parallelen wissensbasierten Sprachverarbeitungssystems. Studienarbeit, Lehrstuhl für Mustererkennung (Informatik 5), Universität Erlangen–Nürnberg, 1997.
8. Y. Bengio. *Neural Networks for Speech and Sequence Recognition*. International Thomson Computer Press, London, 1995.
9. R. Kuhn. *Keyword Classification Trees for Speech Understanding Systems*. PhD thesis, School of Computer Science, McGill University, Montreal, 1993.
10. E. G. Schukat-Talamazzini. *Automatische Spracherkennung – Grundlagen, statistische Modelle und effiziente Algorithmen*. Vieweg, Braunschweig, 1995.
11. W. Eckert, E. Nöth, H Niemann, and E.G. Schukat-Talamazzini. Real Users Behave Weird — Experiences made collecting large Human–Machine–Dialog Corpora. In *Proc. of the ESCA Tutorial and Research Workshop on Spoken Dialogue Systems*, pages 193–196, Vigsø, Denmark, June 1995.