Hybrid Connectionist and Classical Approaches in JANUS An Advanced Speech-to-Speech Translation System

A. Waibel [1,2], T.S. Polzin[1], U. Bodenhausen[2], F.D. Buø[2], N. Coccaro[1], H. Hild[2], B. Suhm[1]
Interactive Systems Laboratories
[1] Carnegie Mellon University, Pittsburgh, USA
[2] Karlsruhe University, Karlsruhe, Germany

***Abstract*— In this paper we report on our efforts to combine speech and language processing toward multi-lingual spontaneous speech translation. The ongoing work extends our JANUS system effort toward handling spontaneous spoken discourse and multiple languages. After an overview of the task, databases, and the system architecture we will focus on how connectionist modules are integrated in the overall system design. We will show that these modules can because of their learning capabilities adapt themselves to the problem space. Moreover, because of their inherent robustness against noise they seem to be an adequate tool for analyzing spontaneous speech.**

1 Introduction

The goal of the JANUS project is *multi-lingual* machine translation of *spontaneously spoken dialogs* in a limited domain. Currently we are using the scheduling domain, i.e. two people scheduling a meeting with each other. We are working with German, Spanish, and English as source languages and German, English, and Japanese as target languages. This paper reports on our efforts to make Natural Language Processing (NLP) robust over spontaneous speech and to use NLP to constrain speech recognition. In this article we focus on connectionist approaches to these problems. For statistical and knowledge-based approaches to robust parsing and dialog modeling within the JANUS project see [19]. We consider the connectionist modules in JANUS as additional processing stages complementing other alternate modules. Our investigation of connectionist approaches in JANUS tries to overcome the following problems:

1. So far, specifying the parsing grammars takes most of the time in the development. Note, that each source language requires its own parsing grammar. In using connectionist learning algorithms we try to skip this step in the development.
2. When recognizing or parsing spontaneous speech one has to handle phenomena like restarts, repairs and repetitions. Spontaneous language does not agree with traditional competence based grammar theories. The inherent robustness of connectionist modules will give us an approximation of the solution even when the input was noisy.
3. Connectionist systems allow an easy integration of different information sources. Traditional parsers base their decisions only on syntactic information. The integration of eg. semantic or prosodic information in a connectionist module stabilizes processing and reduces the amount of ambiguity in its output.

In the following we will describe the Scheduling Task. Then we will give a brief overview of the system architecture. Within this architecture we use connectionist modules at three different processing stages:

- Recognition
- Parsing
- Discourse Processing

Each of these connectionist processing stages will be described in the following sections.

2 The Scheduling Task Database

To be able to develop a system for spontaneous speech, we are collecting a large database of human-to-human dialogs on the scheduling task. Several sites in Europe, the US, and Japan have now adopted scheduling as a common task under several research projects. These projects include the German Government's Verbmobil project for German and English translation, the Enthusiast project for Spanish-to-English translation and the activities of the C-STAR consortium of companies and universities in the U.S., Germany, and Japan for translation of German, English, and Japanese; other languages are now being added as new members are joining the consortium.

The data collection procedure involves two subjects who are each given a calendar and are asked to schedule a meeting. There are 13 different calendar scenarios differing from each other in what is scheduled and how much overlap there is in the free time of the two participants. Other scenarios/calendars are periodically added. Data has been collected in English, German, and Spanish using the same data collection protocols at Carnegie Mellon University, Karlsruhe University, and the University of Pittsburgh.

	English		German		Spanish	
	dialogs	utterances	dialogs	utterances	dialogs	utterances
recorded	383	4000	451	4628	146	2920
transcribed	328	3300	215	2293	68	1080

Table 1: State of Data Collection March 1994

Speaker 1: /h#/ /um/ when can we get together
again {comma} < on our
[m(eeting)] > {comma} /um/ to discuss
our project {period} {seos}
/um/ how's @how is@ {comma} /um/ Monday
the eighth {quest} around two
thirty {quest}
#key_click# #paper_ruffle# {seos}

Speaker 2: #key_click# /ls/ /h#/ /uh/
Monday afternoon's @afternoon is@
no good {period} {seos}
I've @I have@ got a meeting from two to
four {comma} {seos}
that's @that is@ not gonna @going to@
give us enough time to get together
{comma} {seos}
/h#/ /um/ *pause* Tuesday afternoon {comma}
the ninth {comma}
would be okay for me though {comma}
#key_click# /h#/ {seos}

Speaker 1: /ls/ /h#/ unfortunately I'll @I will@
be out of town {comma}
from {comma} the ninth {comma} through
the eleventh {period} {seos}
/um/ checking my calendar {comma}
/im/ /h#/ Friday's @Friday is@ no
good {comma} either {period} {seos}
let's @let us@ see {comma} maybe next week
{comma} {seos} /h#/ /oh/ /h#/ that's
@that is@ bad {comma} {seos}
< my class schedule's @schedule is@
[t] {comma} {seos} >
okay {comma} /h#/ how 'bout on
Tuesday the sixteenth {comma}
any time after twelve thirty {period}
#key_click# /h#/ /h#/ {seos}

Figure 1: Sample Transcription: Text contained in slashes represent human noise; hash marks–non-human noise; curly braces-intonation (except {seos}); angle brackets–false starts; square brackets–mispronunciations; @–contractions; {seos}–end of semantic sentence unit.

The advantages of this experimental design using the same calendars for all languages is that it solicits similar domain-limited dialogs while ensuring a spontaneous, natural (not read or contrived) speaking style. Thus techniques can be compared across languages, and have enabled us to explore automatic knowledge-acquisition and MT techniques in several languages on a comparable task. Table 1 specifies the amount of data collected in each language in terms of the number of dialogs and the number of utterances that have been recorded and transcribed.

We have developed standard transcription conventions that are employed across languages, ensuring uniformity and consistency. Words are transcribed into their conventional spelling. The transcription also indicates human non-speech noises, non-human noises, silences, false starts, mispronunciations, and some intonation. A sample of part of a dialogue is given in Figure 1.

Recent studies [14] and our own observations show that there are a higher rate of disfluencies in human-human dialogs and significantly larger speaking rate variations, compared to human-machine database queries. Table 2 compares disfluencies in human-human spontaneous scheduling tasks (SST) in German, English, and Spanish and human-machine queries (ATIS). The table shows the utterance length in words as well as human noises (filled pauses, laughter, coughs, etc. but not intelligible words such as "okay", "well") and false starts (chopped words and repetitions, deletions, substitutions and insertions of words, but not filled pauses) as percentages of the total number of words in the transcripts [1]. Table 2 suggests that human–human dialogs lead to longer utterances which are more disfluent.

In addition, Table 3 shows perplexities for bigram and trigram language models for English, Spanish, and

[1] To exclude artifacts from differing data collection set–ups we didn't consider non–human noises (e.g. clicks, paper rustle) in this statistics.

Figure 2: System Diagram

We employ a multi-strategy approach for several of the main processes. For example, we are experimenting with TDNN, MS-TDNN, MLP, LVQ, and HMM's for acoustic modeling; n-grams, word clustering, and automatic phrase detection for language modeling; statistically trained skipping LR parsing, connectionist parsing, and robust semantic parsing for syntactic and semantic analysis; and statistical models as well

[2]Discourse processing has not yet been implemented. In this paper we are reporting our plans for this component.

as plan inferencing for identification of the discourse state. The multi-strategy approach should lead to improved performance with appropriate weighting of the output from each strategy.

Processing starts with speech input in the source language. Recognition of the speech signal is done with the acoustic modeling methods mentioned above, constrained by the language model, which is influenced by the current discourse state. This produces a list of the N-best sentence candidates, which are then sent to the translation components of the system.

At the core of our machine translation system is an interlingua, which is intended to be a language-independent representation of meaning. The parser outputs a preliminary interlingua text (ILT) or some ILT fragments corresponding to the source language input. After parsing, the ILT is further specified by the discourse processor. The discourse processor performs functions such as disambiguating the speech act or discourse function, resolving ellipsis and anaphora, and assembling ILT fragments into full ILTs. It also updates a calendar in the dynamic discourse memory to keep track of what the speakers have said about their schedules. Based on the current discourse state, a flag is set, which is used by the parser to resolve ambiguities in the next sentence to be parsed, and by the recognizer to dynamically adapt the language model to recognize the next utterance. Once the ILT is fully specified, it can be sent to the generator to be rendered in any of the target languages.

The formalism used to specify an ILT is called a *feature structure*. Feature structures and variants of them[3] is a frequently used representation scheme in computational linguistics.

Connectionist modules enter this architecture at three places:

1. the recognizer
2. the parser
3. discourse processor

4 Connectionist Speech Recognition

The connectionist speech recognition modules of the system are based on two different approaches: Time-Delay Neural Networks (TDNN) and Learning Vector Quanitzation (LVQ). These approaches and the current research issues are briefly introduced in the next sections.

4.1 Continuous Speech recognition using LVQ/HMM-Hybrids

LVQ is a neural network based learning vector clustering technique. We have used LVQ to automatically cluster speech frames into a set of acoustic features. These features are fed into a set of output units which compute the emission probability of HMM states.

In the LVQ based speech recognizer designed for the Conference Registration Task and the Resource Management Task, an LVQ algorithm with context-dependent phonemes is used for speaker independent recognition. For each phoneme, there is a context independent set of prototypical vectors. The output scores for each phoneme segment are computed from the euclidean distance using context dependent segment weights.

Recent improvements of these recognizers include the introduction of noise models as well as the improvement of the training algorithms; the 1994 results in table 4 were obtained using triphone clustering, corrective training and feature weights [18].

	1991	1994
Conference Registration	9.1 %	3.7 %
Resource Management	24 %	5.1 %

Table 4: Comparison of error-rates

The noise modeling in the recognizer was recently improved for the spontaneous speech tasks (ESST, GSST, SSST, see above). In order to generate acoustic models for the human and nonhuman noises, we have build classes of noises to maximize available training data per model. Frequent human noises ("ah", breathing, lip smack, "uh", "um") and nonhuman noises (key klick, paper rustle) form a class of their own. Less frequent human noises build one class and rare nonhuman noises are joined in another class. A special class was introduced to handle those word fragments which were generated by restarts, repeats, etc., and could not be modeled as regular words. For each of these 10 noise classes a dedicated phoneme was added to the list of phonemes. Although approx. 20% of all words in the spontaneous speech tasks are noises, the lack of training data remains the main problem of acoustic noise modeling. Current research investigates improvements through clustering of the 10 noise classes [11].

[3] Basics can be found in [5, 13, 15, 16].

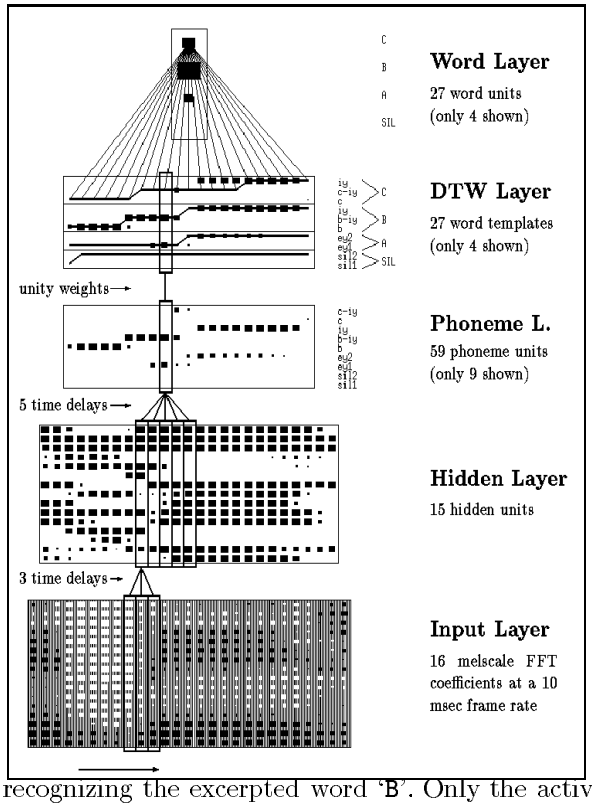

Figure 3: The MS-TDNN recognizing the excerpted word '`B`'. Only the activations for the words '`SIL`', '`A`', '`B`', and '`C`' are shown.

4.2 Letter Recognition with The MS-TDNN

The recognition of spelled strings of letters is essential for speech recognition application involving proper names or addresses. However, it is very difficult to get good recognition results on the highly confusable letters, if they constitute only a small fraction among thousands of words of a large vocabulary recognizer. In addition, since most letters are very short, they are easily inserted everywhere, and many words are pronounced like letter sequences, for example"See you" $\equiv$ "C U", or "R U E C" $\equiv$ "are you easy". To achieve reasonable results on letter strings, we developed a specialized letter recognizer, which is based on the *MS-TDNN* architecture.

The MS-TDNN [7, 8] is an extension of the TDNN architecture[20]. It integrates the time-shift invariant architecture of a TDNN and a nonlinear time alignment procedure (DTW) into a high accuracy word-level classifier. Figure 3 shows the MS-TDNN in the process of recognizing the excerpted word '`B`', represented by 16 melscale FFT coefficients at a 10 msec frame rate. The first three layers constitute a standard TDNN, which uses sliding windows with time delayed connections to compute a score for each phoneme for every frame, these are the activations in the "Phoneme Layer". Each word to be recognized is modeled by a sequence of phonemes. In the "DTW (Dynamic Time Warping) Layer", an optimal alignment path, i.e. the path with the highest accumulative phoneme scores is found for each word, the activations along these paths are then collected in the word output units. The network works with a relatively small number of parameters. 50 rows of hidden units are used for speaker–independent recognition, corresponding to about 20000 trainable parameters, i.e. network weights.

Training starts with "bootstrapping", during which only the front-end TDNN is trained as a frame-by-frame phoneme classifier, with phoneme boundaries fixed as given in the training data. In a second phase, training is extended to the word level, where phoneme boundaries within the given word boundaries are freely aligned in the DTW Layer. Instead of phonemes, the output are now words, and error derivatives are backpropagated from the word units through the alignment paths and the front-end TDNN.

The choice of sensible objective functions is of great importance. For training on the phoneme level, there is an output vector $Y = (y_1, \ldots, y_n)$ and a corresponding target vector $T = (t_1, \ldots, t_n)$ for each frame in time. T represents the correct phoneme j in a "1-out-of-n" coding, i.e. $t_i = \delta_{ij}$. Standard *Mean Square Error* ($MSE = \sum_{i=1}^{n} (y_i - t_i)^2$) is problematic for "1-out-of-n" codings for large n ($n > 50$ in our case); consider for example that for a target $(1, 0, \ldots, 0)$, the output $(0.0, \ldots, 0.0)$ has only half the error than the more desirable output $(1.0, 0.2, \ldots, 0.2)$. This problem is avoided by

$$E_{McClelland}(T, Y) = \sum_{i=1}^{n} log(1 - (y_i - t_i)^2)$$

which (like cross entropy) punishes "outliers" with an error approaching infinity for $|t_i - y_i|$ approaching 1.0.

For the word level training, we have achieved best results with an objective function similar to the "Classification Figure of Merit" (CFM) [6], which tries to maximize the distance $d = y_c - y_{hi}$ between the correct score y_c and the highest incorrect score y_{hi} instead of using absolute targets 1.0 and 0.0 for correct and incorrect word units:

$$E_{CFM}(T,Y) = f(y_c - y_{hi}) = f(d) = (1-d)^2$$

The philosophy here is not to "touch" any output unit not directly related to correct classification. We found it even useful to backpropagate error only in the case of a wrong or too narrow classification, i.e. if

$$y_c - y_{hi} < \delta_{safety\_margin}$$

Experiments. The recognizer was trained and tested both on large English and German data bases. The English performance was measured on the DARPA Resource Management Spell-mode data, consisting of a total of 1680 spelled words from 120 speakers. We achieved speaker-independent recognition results of 92.0% letter accuracy. The larger German data base consists of a training set of over 8000 strings spelled by 70 speakers. 90.1% letter accuracy was achieved on a test set of 1316 strings. With every tenth letter misrecognized, the string accuracy (as required for example to spell a name or word) is still only 56%. However, we have experimented with several techniques[10] to constrain the search space. When the search space was limited to a list of 40,000 unique names (from a telephone book with 111,000 entries), letter and string accuracy could be improved up to 97% and 92%, respectively.

4.3 Automatic Structuring of Neural Networks for Speech Recognition

Despite the use of powerful learning algorithms for most parameters of a speech recognizer, the best possible performance greatly depends on the tuning of the architecture to the particular task. For fast adaptation of connectionist speech recognizers to new domains without laborious manual tuning the following algorithms were developed:

- The Automatic Structure Optimization (ASO) algorithm [1] that does the architectural tuning automatically for neural network speech recognition systems.
- The Automatic Validation Analyzing Control System (AVACS) [2] that is designed to detect overfitting models on a class by class basis as early as possible and to selectively change their learning and automatic structuring process.

For the application of neural networks to speech recognition all of the following architectural parameters have to be well adapted to the task and the given amount of training data (see Figure 3):

- the number of hidden units
- the size of the input windows
- the number of phonemic states that model an acoustic event

The ASO algorithm automatically adapts all of these architectural parameters to the given task and amount of training data in a single training run. The algorithm offers the flexibility to apply neural net speech recognition systems to new domains without the need for manual tuning of the architecture.

ASO uses a constructive approach and starts with a small number of parameters for the given number of training examples and increases this number to improve the performance on the training set. Unlike the human developer, the ASO algorithm starts making decisions about resource allocations very early in the training run, i.e. it is tuning the architecture while the network is learning the task ("tuning by doing"). This allows the algorithm to complete the optimization process in a single training run.

AVACS monitors the learning and tuning process and is designed to detect and avoid overfitting models on a class by class basis. A validation set is used to test the generalization ability of the system frequently in the training run. The confusion matrices are computed for both the training and the validation data. From these matrices a new confusion-difference matrix is computed. Overfitting can be easily detected from this matrix. In this case, further allocation of new parameters is delayed and all the weights involved in overfitting are contaminated with 10 - 30% of noise.

In addition to the advantage of offering an automatic architecture optimization that is automatically validated and controlled, our approach also offers an attempt towards controlled error equalization.

Experiments. The algorithms were applied to the optimization of an MS-TDNN (see above) used for speaker dependent connected English letter recognition. The automatically tuned MS-TDNN achieved 97.4% letter accuracy compared to a 97.5% letter accuracy of a handtuned MS-TDNN with a manual tuning effort of more than one person-year.

5 Connectionist Parsing

We use two different approaches to connectionist parsing, both of which are descendants of the PARSEC-system developed by Jain [12]. However, both systems improved their expressive power compared with

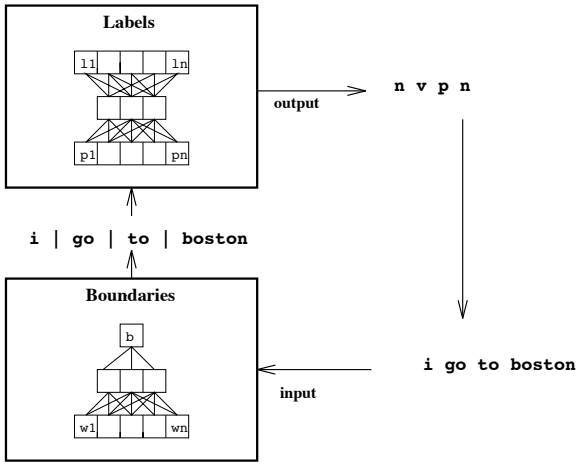

Figure 4: Two backpropagation networks that are recursively connected using symbolic procedures to store interim results. The `Boundaries`-module computes phrase boundaries. Given a certain context window of words w_m-w_n the network decides whether a particular word w_i ($m \leq i \leq n$) marks a phrase boundary. The `labeling`-module computes labels for the phrases found in the former processing stage. Given a certain context window of phrases p_r-p_s (which are represented by the respective words) the network computes the label l_j for a particular phrase p_k ($r \leq k \leq s$).

the old PARSEC-system which could only output a flat case-based structure without more specific feature information. As a consequence, the internal structure and features of eg. a noun phrase was left unanalyzed. The first approach [17] computes a traditional syntax tree for an input sentence. The second approach [4] tries to output an ILT , or more general, a feature structure (cf. section 3). Both approaches try to include semantic and prosodic information in their processing. Moreover, both approaches are not purely connectionist but are based on a hybrid architecture where symbolic procedures are used to map information from one network to another or to focus on certain information. This hybrid architecture helps to keep the networks small and accelerate the training process.

5.1 Integrated Compositional Connectionist Parsing (ProPars)

This approach starts with the observations that sentences have no upper bound on their length and on the depth of the resulting parse trees. This unboundedness of sentence length and output structure has posed a problem for connectionist based parsing systems so far. Moreover, basing parsing decisions only on syntactic information would result in ambiguities or wrong parse trees. Therefore we include semantic and prosodic information in the parsing process. The overall system architecture of this module (ProPars) consists of two backpropagation networks. The basic architecture is given in Figure 4. The network labeled `Boundaries` is responsible for breaking up the input string into phrases.[4] The second module, `Labels`, is responsible for labeling these phrases. The combined work of the `Boundaries`- and `Labels`-module assigns a constituent structure to the string, where the assigned label is the dominating node and the daughters are the corresponding nodes within the phrase.

Words are represented in the lexicon as binary feature vectors. Within each vector a lexeme has a unique binary-Id encoding, a vector segment representing syntactic information, and a segment encoding information about semantic properties (cf. Figure 5). During processing the `Boundaries`-module checks for each word in the current input whether it marks a phrase boundary. To do this all vectors of the words in the current context window are presented to the network. The context window moves over the string from left to right. After all boundaries have been determined, the symbolic procedures map all words (i.e. their vectors) and the phrase boundaries to the input nodes of the second `Labels`-network. This network assigns labels such as noun (`n`), noun phrase (`np`), verb (`v`), or verb phrase (`vp`) to each phrase. The procedure is similar to the one used for computing phrase boundaries. A context window moves over the phrases in the current input from left to right and for each phrase the network determines its label. Note the recursion in this architecture. The assigned list of constituent labels at some time *t* serves as the input at time *t+1*. The recursion exits if a string of constituents can be reduced to the single sentence symbol *s*. Since the list of constituent labels forms a "sentence" the respective labels have to be defined in the lexicon, too. Thus, we have to specify lexical items like nouns `n` and verbs `v` and their projections. In this setup we need symbolic procedures for:

- Storing intermediate results: We have to store for each pass (i.e. recursion step) phrase boundaries and labeling.
- Mapping from the `Boundaries`-module to the `Labels`-module.

[4] As the discussion below will show, we extend the usage of the word *phrase* which traditionally refers to wellformed strings of terminals to wellformed strings of non-terminals as well.

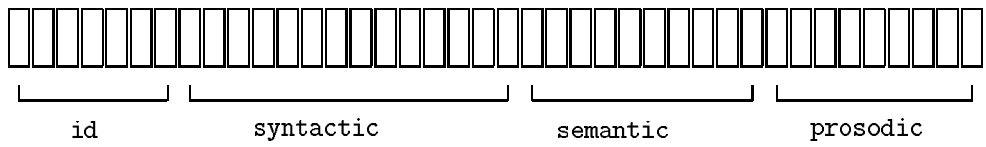

Figure 5: **Lexical vector** with additional prosodic information

```
([]((speech-act *confirm)
    (sentence-type *state)
    (frame *clarify))
        ([]
                ([topic]((frame *simple-time))
                        ([]                                  by)
                        ([]((day-of-week monday))            monday))
                ([]     ([]                                  i))
                ([]((adverb perhaps))
                        ([]                                  assume)))
        ([clarified]
                ([]     ([]                                  you))
                ([]     ([]                                  mean))
                ([]((frame *simple-time))
                        ([]((day-of-week monday))           monday)
                        ([]                                  the)
                        ([]((day 27))                        ([rego] twenty seventh)))))
```

Figure 6: Sentence aligned with its feature structure

- Assembling the final syntax tree from the information stored during parsing. This procedure involves, for example, deleting unary productions of the form $\gamma \rightarrow \gamma$.

As indicated in Figure 5 we integrate prosodic information into the parsing process. Consider an utterance "Let us talk about the conference on Monday". The prepositional phrase "on Monday" can either attach to the verb or to the noun phrase "the conference" giving rise to two different meanings. We hope that the two different meanings are marked prosodically different such that we can use this information to disambiguate between these two readings. Another application for an integrated connectionist parsing approach is what we call *utterance parsing*. Consider the sample description in Figure 2. The output of the recognizer - and the input to the parser-modules - is, of course, the unstructured string comprising most of the time more than one sentence. This means that we have to deal with sentence boundaries as well. We hope that an integrated approach by using syntactic, semantic, and prosodic information is needed to handle this problem.

5.2 Feature Structure Connectionist Parsing (FeasPars)

In a different approach we tried to design a connectionist parsing system FeasPars that directly computes the ILT for a given input sentence (cf. section 3).
When we compare a sentence with its ILT representation, we see that there is a correspondence between parts of the feature structure, and specific constituents of the sentence.

Aligning our sentence with parts of the feature structure, we get a structure as shown in figure 6. We note that:

- The sentence is hierarchically split into constituents.
- Feature pairs are listed with their corresponding constituent.
- Paths are shown in square brackets, and express how a constituent relates to its parent constituent.

Over 600 sentences from the ESST task where labeled according to this scheme.

FeasPars consists of three main parts:

1. The Chunker
2. The Linguistic Feature Labeler
3. The Constituent Path Finder

The *Chunker* splits an input sentence into constituents. It consists of three networks. The first network finds regular expressions, such as numbers. Numbers are classified as being ordinal or cardinal numbers. These regular expressions are presented as words by the following networks. The next network groups

words together to phrases. The third network groups phrases together to clauses. In total, we get **four levels** of constituency; word/regular expressions, phrases, clauses and sentence.

The *Linguistic Feature Labeler* attaches features and feature values (if applicable) to these constituents.

For each feature, there is a classifier, which finds one or zero atomic value. Since there are many features, each constituent may get none, one or several pairs of feature and atomic values. Since a feature normally only occurs at a certain constituent level, the classifier is specialized for deciding about a particular feature at a particular constituent level. This specialization is there to prevent the learning task from being too complex, thus rendering it well learnable.

The *Constituent Path Finder* determines how a constituent relates to its parent constituent. It has one classifier per constituent level and constituent path element.

The following example illustrates how the three parts work:

The parser receives the English sentence:

Can you meet in the morning

The Chunker segments the sentence before passing it to the Linguistic Feature Labeler, which adds semantic labels (shown in **boldface** below):

```
(((speech-act *suggest)
  (sentence-type *query-if))
  ((frame *free))
   ((                              can))
   (((frame *you))
    (                              you))
   ((                              meet))
   ((frame *special-time))
    (                              in)
    (((specifier definite))        the)
    (((time-of-day morning))       morning)))
```

The Constituent Path Finder then adds paths, where appropriate (shown in **boldface**), and we get:

```
([](( speech-act *suggest)
    ( sentence-type *query-if))
    ([]( frame *free))
      ([][]                        can))
      ([who](( frame *you))
        ([]                        you))
      ([][]                        meet))
      ([when](( frame *special-time))
        ([]                        in)
        ([](( specifier definite)) the)
        ([](( time-of-day morning)) morning)))
```

Converting this to a feature structure, we get the ILT:

```
(((speech-act *suggest)
  (sentence-type *query-if)
  (frame *free)
  (who ((frame *you)))
  (when ((frame *simple-time)
         (time-of-day morning)))))
```

6 Connectionist Discourse Modeling

Work is also underway to model the discourse by making predictions of subsequent ILTs based on the previous ones, using a connectionist implementation. The ILT generated by our LR parser is a language independent frame structure containing three main slots, *speech-act*, *sentence-type* and *semantic frame* along with a few other secondary slots. The speech-act refers to the action performed by the sentence, e.g., suggest, accept, and reject. The sentence type refers to the surface form of the sentence, e.g., statement, yes/no question, wh-question, directive. The semantic frame refers to main semantic content of the sentence, e.g., busy, free, out-of-town. Other slots in the ILT are *who*, which is the person referred to by the frame, *what*, a possible non-person object, and *when* and *topic*, a representation of any temporal component of the utterance.

Spoken Utterance: Actually the twenty sixth and the twenty seventh I'll be at a seminar all day.

```
((SPEECH-ACT *REJECT)
 (SENTENCE-TYPE *STATE)
 (FRAME *SCHEDULED)
 (WHO ((FRAME *I)))
 (TOPIC
     ((FRAME *TIME-LIST)
      (CONNECTIVE AND)
      (ITEMS
          (*MULTIPLE*
              ((FRAME *SIMPLE-TIME)
               (DAY 26))
              ((FRAME *SIMPLE-TIME)
               (DAY 27))))))
 ( WHAT  ((FRAME *SEMINAR)
          (SPECIFIER INDEFINITE)))
 (WHEN
     ((FRAME *SPECIAL-TIME)
      (SPECIFIER WHOLE)
      (NAME DAY)))
 ( ADVERB ACTUALLY))
```

Figure 7: Example for the ILT representation

The top level slots of the most recent ILT are encoded into a pattern of binary inputs. This information along with a bit indicating whether the next utterance comes from the same or different speaker in the dialog is fed into a multi layer neural network trained by the back-propagation learning algorithm. The network has to map the input vectors onto a representation of the subsequent ILT. In preliminary work, the network was trained on 24 dialogues of hand coded ILTs from the ESST database. The network learned some characteristics of discourse behavior, and is good at making some predictions of likely fillers for the speech-act and sentence-type slots of the subsequent ILT. The relative strength of the output units can be used to determine the relative probability of competing fillers for a particular slot.

There are drawbacks with this experiment that are easy to solve: Twenty-four dialogues is not sufficient for good modeling of discourse. Increasing data for training is underway to yield improved results. Interjections often disrupt the context of a sentence; for instance small utterances, such as *Well* between two full sentences interfere with the association of the ILTs for the two sentences. Using the previous content bearing ILT to predict the next ILT, rather than just the previous ILT, increases context, and can boost results. With improved results, the predictions will be used to aid speech recognition by interpolating language models appropriate for sentences containing the predicted slots.

7 Acknowledgments

This work was supported by grants, donations, and discounts from the Advanced Research Project Agency, ATR, BMFT (Verbmobil), DEC, the National Science Foundation, NEC, the Research Council of Norway, and Siemens. We gratefully acknowledge their support. The views and conclusions contained in this document are those of the authors and should not be interpreted as necessarily representing the official policies, either expressed or implied, of the supporting organizations.

References

[1] U. Bodenhausen and S. Manke. Connectionist Architectural Learning for High Performance Character and Speech Recognition. In *ICASSP-93, International Conference on Acoustic, Speech & Signal Processing, San Francisco, CA*, 1993.

[2] U. Bodenhausen and A. Waibel. Tuning By Doing: Flexibility Through Automatic Structure Optimization. In *EUROSPEECH, Berlin, Germany*, 1993.

[3] J. Bresnan. *The Mental Representation of Grammatical Relations*. The MIT Press, Cambridge, MA, 1982.

[4] F.D. Buø, T.S. Polzin, and A. Waibel. Learning Complex Output Representations in Connectionist Parsing of Spontaneous Speech. In *ICASSP-94, International Conference on Acoustic, Speech & Signal Processing, Adelaide, Australia*, pages I–365–368, 1994.

[5] G. Gazdar, E. Klein, G. K. Pullum, and I. A. Sag. *Generalized Phrase Structure Grammar*. Blackwell Publishing, Oxford, England and Harvard University Press, Cambridge, MA, USA, 1985.

[6] J. Hampshire and A. Waibel. A Novel Objective Function for Improved Phoneme Recogn. Using Time Delay Neural Networks. *IEEE Trans. on Neural Networks*, June 1990.

[7] P. Haffner, M. Franzini, and A. Waibel. Integrating Time Alignment and Neural Networks for High Performance Continuous Speech Recognition. In *Proc. Intern. Conf. on Acoustics, Speech and Signal Processing*, IEEE, 1991

[8] H. Hild and A. Waibel. Multi-Speaker/Speaker-Independent Architectures for the Multi-State Time Delay Neural Network. In *Proc. Intern. Conference on Acoustics, Speech, and Signal Processing*. IEEE, 1993.

[9] H. Hild and A. Waibel. Speaker-Independent Connected Letter Recognition With a Multi-Sate Time Delay Neural Network. In *3rd European Conference on Speech, Communication and Technology (EUROSPEECH) 93*, September 1993.

[10] H. Hild and M. Betz. Language Models for a Letter Recognizer. *Submitted to: Proc. IEEE International Conference on Acoustics, Speech, and Signal Processing*. IEEE, 1995.

[11] T. Schultz and I. Rogina. Acoustic and Language Modeling of Human and Nonhuman Noises for Human- to-Human Spontaneous Speech Recognition. *Submitted to: Proc. IEEE International Conference on Acoustics, Speech, and Signal Processing*. IEEE, 1995.

[12] A.J. Jain. Parsing complex sentences with structured connectionist networks. *Neural Computation*, 3:110–120, 1991.

[13] R. Kaplan and J. Bresnan. Lexical-functional grammar: A formal system for grammatical representation. In *In [3]*, pages 173–281. The MIT Press, Cambridge, MA, 1982.

[14] S. Oviat. Predicting and managing spoken disfluencies during human-computer interaction. In *Proc. the ARPA Human Language Technology Workshop*, Plainsboro, 1992.

[15] C. Pollard and I. Sag. *Information-Based Syntax and Semantics, Volume 1, Fundamentals*. Number 13 in CSLI Lecture Notes Series. Distributed by University of Chicago Press, Stanford: Center for the Study of Language and Information, 1987.

[16] C. Pollard and I. Sag. *Head-Driven Phrase Structure Grammar*. CSLI Lecture Notes Series. Distributed by University of Chicago Press, Stanford: Center for the Study of Language and Information, 1993.

[17] T. S. Polzin. Parsing spontaneous speech: A hybrid approach. In *Workshop on Combining Connectionist and Symbolic Processing, ECAI-94*, Amsterdam, The Netherlands, 1994.

[18] I. Rogina and A. Waibel, *Learning State-Dependent Stream Weights for Multi-Codebook HMM Speech Recognition Systems*, ICASSP 1994.

[19] B. Suhm, L. Levin, N. Coccaro, J. Carbonell, K. Horiguchi, R. Isotani, A. Lavie, L. Mayfield, C.P Rose, and C. Van Ess-Dykema nd A. Waibel. Speech-language integration in a multi-lingua speech translation system. In *Proc. of AAAI*, 1994.

[20] A. Waibel, T. Hanazawa, G. Hinton, K. Shikano, and K. Lang. Phoneme Recognition Using Time-Delay Neural Networks. *IEEE, Transactions on Acoustics, Speech and Signal Processing*, March 1989.