FeasPar - A Feature Structure Parser Learning to Parse Spoken Language

Finn Dag Buø and Alex Waibel
Interactive Systems Laboratories
University of Karlsruhe, Germany
and Carnegie Mellon University, USA
{finndag|waibel}@ira.uka.de

Abstract

We describe and experimentally evaluate a system, FeasPar, that learns parsing spontaneous speech. To train and run FeasPar (Feature Structure Parser), only limited handmodeled knowledge is required.
The FeasPar architecture consists of neural networks and a search. The networks spilt the incoming sentence into chunks, which are labeled with feature values and chunk relations. Then, the search finds the most probable and consistent feature structure.
FeasPar is trained, tested and evaluated with the Spontaneous Scheduling Task, and compared with a handmodeled LR-parser. The handmodeling effort for FeasPar is 2 weeks. The handmodeling effort for the LR-parser was 4 months. FeasPar performed better than the LR-parser in all six comparisons that are made.

1 Introduction

When building a speech parsing component for small domains, an important goal is to get good performance. If low hand labor is involved, then it's even better.

Unification based formalisms, e.g.(Gazdar et al., 1985; Kaplan and Bresnan, 1982; Pollard and Sag, 1987), have been very successful for analyzing written language, because they have provided parses with rich and detailed linguistic information. However, these approaches have two major drawbacks: first, they require hand-designed symbolic knowledge like lexica and grammar rules, and second, this knowledge is too rigid, causing problems with ungrammaticality and other deviations from linguistic rules. These deviations are manageable and low in number, when analyzing written language, but not for spoken language. The latter also contains spontaneous effects and speech recognition errors. (On the other hand, the good thing is that spoken language tend to contain less complex structures than written language.) Several methods have been suggested compensate for these speech related problems: e.g. score and penalties, probabilistic rules, and skipping words (Dowding et al., 1993; Seneff, 1992; Lavie and Tomita, 1993; Issar and Ward, 1993).

A small community have experimented with either purely statistical approaches(Brown et al., 1990; Schütze, 1993) or connectionist based approaches (Berg, 1991; Miikkulainen and Dyer, 1991; Jain, 1991; Wermter and Weber, 1994). The main problem when using statistical approaches for spoken language processing, is the large amounts of data required to train these models. All connectionist approaches to our knowledge, have suffered from one or more of the following problems: One, parses contains none or too few linguistic attributes to be used in translation or understanding, and/or it is not shown how to use their parse formalism in a total NLP system. Two, no clear and quantitative statement about overall performance is made. Three, the approach has not been evaluated with real world data, but with highly regular sentences. Four, millions of training sentences are required.

In this paper, we present a parser that produces complex feature structures, as known from e.g. GPSG(Gazdar et al., 1985). This parser requires only minor hand labeling, and learns the parsing task itself. It generalizes well, and is robust towards spontaneous effects and speech recognition errors.

The parser is trained and evaluated with the Spontaneous Scheduling Task, which is a negotiation situation, in which two subjects have to decide on time and place for a meeting. The subjects' calendars have conflicts, so that a few sug-

gestions have to go back and forth before finding a time slot suitable for both. The data sets are real-world data, containing spontaneous speech effects. The training set consists of 560 sentences, the development test set of 65 sentences, and the unseen evaluation set of 120 sentences. For clarity, the example sentences in this paper are among the simpler in the training set. The parser is trained with transcribed data only, but evaluated with transcribed and speech data (including speech recognition errors). The parser produces feature structures, holding semantic information. Feature structures are used as interlingua in the JANUS speech-to-speech translation system(Woszczyna et al., 1994). Within our research team, the design of the interlingua ILT was determined by the needs of unification based parser and generator writers. Consequently, the ILT design was not tuned towards connectionist systems. On the contrary, our parser must learn the form of the output provided by a unification based parser.

This paper is organized as follows: First, a short tutorial on feature structures, and how to build them. Second, we describe the parser architecture and how it works. Third, we describe the lexicon. Fourth, we describe the parser's neural aspects. Fifth, a search algorithm is motivated. Then results and conclusion follow.

2 Feature Structures

Feature structures(Gazdar et al., 1985; Pollard and Sag, 1987) are used as output formalism for FeasPar. Their core syntactic properties and terminology are:

1. A *feature structure* is a set of none, one or several *feature pairs*.
2. A *feature pair*, e.g. `(frame *clarify)` , consists of a *feature*, e.g. `frame` or `topic`, and a *feature value*.
3. A *feature value* is either:
   (a) an *atomic value*, e.g. `*clarify`
   (b) a *complex value*
4. A *complex value* is a *feature structure*.

3 The Chunk'n'Label Principle

In contrast to the standard feature structure definition of Section 2, an alternative view-point is to look at a feature structure as a tree[1], where *sets*

[1]This assumes that structure sharing is not possible, see Section 3.1.2.

```
((speech-act *confirm)
 (sentence-type *state)
 (frame *clarify)
 (topic ((frame *simple-time)
         (day-of-week monday)))
 (adverb perhaps)
 (clarified ((frame *simple-time)
             (day-of-week monday)
             (day 27))))
```

Figure 1: Feature structure with the meaning "by monday i assume you mean monday the twenty seventh"

of feature pairs with *atomic values* make up the branches, and the branches are connected with *relations*. Atomic feature pairs belonging to the same branches, have the same relation to all other branches. Further, when comparing the sentence with its feature structure, it appears that there is a correspondence between fragments of the feature structure, and specific chunks of the sentence. In the example feature structure of Figure 1, the following observations about feature pairs and relations apply:

- **feature pairs:**

feature pairs:	**corresponds to:**
`(day 27)`	"the twenty seventh"
`((frame *simple-time)` `(day-of-week monday)` `(day 27))`	"monday the twenty seventh"

- **relations:** the complex value of the feature `topic` corresponds to the chunk "by monday", and the complex value of the feature `clarified` corresponds to "you mean monday the twenty seventh".

Manually aligning the sentence with fragments of the feature structure, gives a structure as shown in Figure 2. A few comments apply to this figure:

- The sentence is hierarchically split into chunks.
- Feature pairs are listed with their corresponding chunk.
- Relations are shown in square brackets, and express how a chunk relates to its parent chunk. Relations may contain more than one element. This allows several nesting levels.

Once having obtained the information in Figure 2, producing a feature structure is straight forward, using the algorithm of Figure 3. Summing up, we can define this procedure as the *chunk'n'label* principle of parsing:

```
([]((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 2: Chunk parse: Sentence aligned with its feature structure (see text for explanation).

1. Split the incoming sentence into hierarchical chunks.

2. Label each chuck with feature pairs and feature relations.

3. Convert this into a feature structure, using the algorithm of Figure 3.

```
FUNCTION convert()
VAR
  S: set;
  C: chunk;
BEGIN
  S := empty set;
  assign(S,top_level_chunk);
  return(S);
END;
  PROCEDURE assign(VAR S: set;
                       C: chunk);
    BEGIN
      P := chunk_relation(C);
      FOR each relation element PE in P
        BEGIN
          S' := empty set;
          include (PE,S') in S;
          S := S';
        END;
      FOR each feature pair FP in C
        include FP in S;
      FOR each chunk C' in C
        assign(S,C);
    END;
```

Figure 3: Algorithm for converting a parse to a feature structure

3.1 Theoretical Limitations

The chunk'n'label principle has a few theoretical limitations compared with the feature structure formalisms commonly used in unification-based parsing, e.g. (Gazdar et al., 1985).

3.1.1 Depth

With the chunk'n'label principle, the feature structure has a maximum nesting depth. One could expect the maximal nesting depth to cause limitations. However, these limitations are only theoretical, because very deep nesting is hardly needed in practice for spoken language. Due to the ability to model relations of more than length 1, no nesting depth problems occurred while modeling over 600 sentences from the English Spontaneous Scheduling Task (ESST).

3.1.2 Structure Sharing

Many unification formalisms allow feature values to be shared. The chunk'n'label principle does not incorporate any mechanism for this. However, all work with ESST and ILT empirically showed that there is no need for structure sharing. This observation suggests that for semantic analysis, structure sharing is statistically insignificant, even if its existence is theoretically present.

4 Baseline Parser

The chunk'n'label principle is the basis for the design and implementation of the *FeasPar* parser. FeasPar uses neural networks to learn to produce chunk parses. It has two modes: learn mode and run mode. In learn mode, manually modeled chunk parses are split into several separate training sets; one per neural network. Then, the networks are trained independently of each other, allowing for parallel training on several CPU's. In run mode, the input sentence is processed through all networks, giving a chunk parse, which is passed

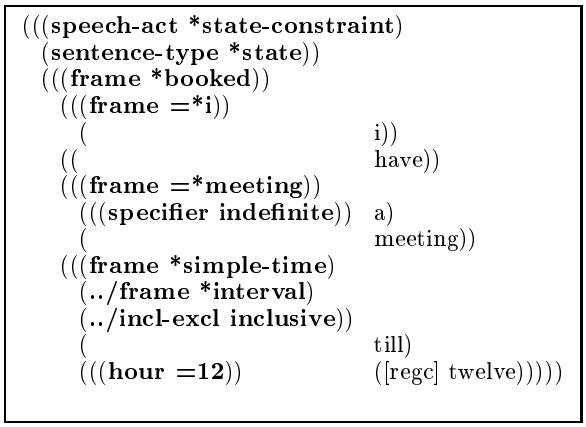

Figure 4: Chunked and labeled sentence (labels shown in **boldface**)

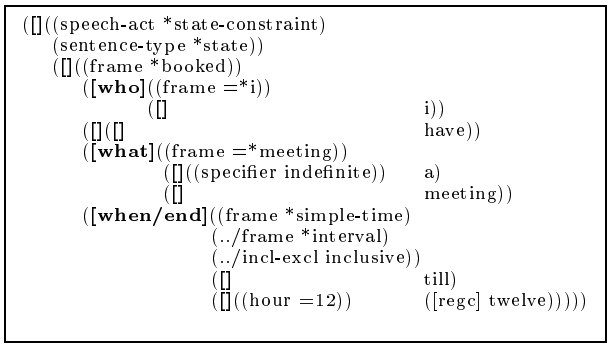

Figure 5: Chunk parse (chunk relations shown in **boldface**)

on to the converting algorithm shown in Figure 3.

In the following, the three main modules required to produce a chunk parse are described:

The Chunker splits an input sentence into chunks. It consists of three neural networks. The first network finds numbers. They are classified as being ordinal or cardinal numbers, and are presented as words to the following networks. The next network groups words together to phrases. The third network groups phrases together into clauses. In total, there are four levels of chunks: word/numbers, phrases, clauses and sentence.

The Linguistic Feature Labeler attaches features and atomic feature values (if applicable) to these chunks. For each feature, there is a network, which finds one or zero atomic values. Since there are many features, each chunk may get no, one or several pairs of features and atomic values. Since a feature normally only occurs at a certain chunk level, the network is tailored to decide on a particular feature at a particular chunk level. This specialization is there to prevent the learning task

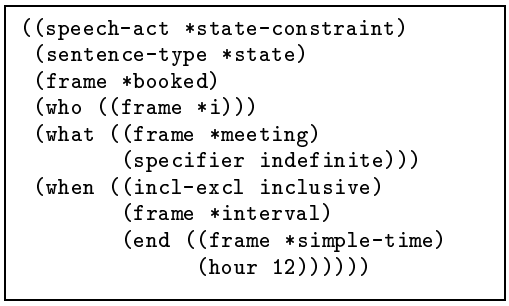

Figure 6: Feature structure parse

from becoming too complex. A special atomic feature value is called lexical feature value. It is indicated by '=' and means that the neural network only detects the *occurrence* of a value, whereas the value itself is found by a lexicon lookup. The lexical feature values are a true hybrid mechanism, where symbolic knowledge is included when the neural network signals so. Furthermore, features may be marked as *up*-features (e.g. `../incl-excl` in Figure 4 and 5). An up-feature is propagated up to its parent branch when building the feature structure (see Figure 6).

The Chunk Relation Finder determines how a chunk relates to its parent chunk. It has one network per chunk level and chunk relation element.

The following example illustrates in detail how the three parts work. For clarity, this example assumes that all networks perform perfectly. The parser gets the English sentence:

"i have a meeting till twelve"

The Chunker segments the sentence before passing it to the Linguistic Feature Labeler, which adds semantic labels (see Figure 4). The Chunk Relation Finder then adds relations, where appropriate, and we get the chunk parse as shown in Figure 5. Finally, processing it by the algorithm in Figure 3, gives the final parse, the feature structure, as shown in Figure 6.

4.1 Lexicon

FeasPar uses a full word form lexicon. The lexicon consists of three parts: one, a syntactic and semantic microfeature vector per word, second, lexical feature values, and three, statistical microfeatures.

Syntactic and semantic microfeatures are represented for each word as a vector of binary values. These vectors are used as input to the neural networks. As the neural networks learn their tasks based on the microfeatures, and not based on distinct words, adding new words using the same microfeatures is easy and does not degrade general-

ization performance. The number and selection of microfeatures are domain dependent and must be made manually. For ESST, the lexicon contains domain independent syntactic and domain dependent semantic microfeatures. To manually model a 600 word ESST vocabulary requires 3 full days.

Lexical feature values are stored in look-up tables, which are accessed when the Linguistic Feature Labeler indicates a lexical feature value. These tables are generated automatically from the training data, and can easily be extended by hand for more generality and new words. An automatic ambiguity checker warns if similar words or phrases map to ambiguous lexical feature values.

Statistical microfeatures are represented for each word as a vector of continuous values v_{stat}. These microfeatures, each of them representing a feature pair, are extracted automatically. For every feature value at a certain chunk level, if there exists a word such that, given this word in the training data, the feature value occurs in more than 50 % of the cases. One continuous microfeature value v_{stat} for a word w is set automatically to the percentage of feature value occurrence given that word w.

4.2 Neural Architecture and Training

All neural networks have one hidden layer, and are conventional feed-forward networks. The learning is done with standard back-propagation, combined with the constructive learning algorithm PCL(Jain, 1991), where learning starts using a small context, which is increased later in the learning process. This causes local dependencies to be learned first.

Generalization performance is increased by *sparse connectivity*. This connection principle is based on the microfeatures in the lexicon that are relevant to a particular network. The Chunker networks are only connected to the syntactic microfeatures, because chunking is a syntactic task. With ESST, the Linguistic Feature Labeler and Chunk Relation Finder networks are connected only to the semantic microfeatures, and to relevant statistical microfeatures. All connectivity setup is automatic. Further techniques for improving performance are described in (Buø, 1996). For the neural networks, the average test set performance is 95.4 %

5 Search

The complete parse depends on many neural networks. Most networks have a certain error rate; only a few networks are perfect. When building complete feature structures, these network errors multiply up, resulting in not only that many feature structures are erroneous, but also inconsistent and making no sense.

To compensate for this, we wrote a search algorithm. It's based on two information sources: First, scores that originates from the network output activations; second, a formal feature structure specification, stating what mixture of feature pairs are consistent. This specification was already available as an interlingua specification document.

Using these two information sources, the search finds the feature structure with the highest score, under the constraint of being consistent. The search is described in more detail in (Buø and Waibel, 1996; Buø, 1996).

6 Results

	FeasPar	GLR* Parser
PM1 - T	71.8 %	51.6 %
PM1 - S	52.3 %	30.3 %
PM2E - T	74 %	63 %
PM2E - S	49 %	28 %
PM3G - T	49 %	42 %
PM2G - S	36 %	17 %

Figure 7: Results

FeasPar is compared with a handmodeled LR-parser. The handmodeling effort for FeasPar is 2 weeks. The handmodeling effort for the LR-parser was 4 months.

The evaluation environment is the JANUS speech translation system for the Spontaneous Scheduling Task. The system have one parser and one generator per language. All parsers and generators are written using CMU's GLR/GLR* system(Lavie and Tomita, 1993). They all share the same interlingua, ILT, which is a special case of LFG or feature structures.

All Performance measures are run with transcribed (T) sentences and with speech (S) sentences containing speech recognition errors. Performance measure 1 is the feature accuracy, where all features of a parser-made feature structure are compared with feature of the correct handmodeled feature structure. Performance measure 2 is the end-to-end translation ratio for acceptable non-trivial sentences achieved when LR-generators are used as back-ends of the parsers. Performance measure 2 uses an English LR-generator (handmodeled for 2 years), providing results for English-to-English translation, whereas performance measure 3 uses a German LR-generator (handmodeled

for 6 months), hence providing results for English-to-German translations. Results for an unseen, independent evaluation set are shown in Figure 7.

As we see, FeasPar is better than the LR-parser in all six comparison performance measures made.

7 Conclusion

We described and experimentally evaluated a system, FeasPar, that learns parsing spontaneous speech. To train and run FeasPar (Feature Structure Parser), only limited handmodeled knowledge is required (chunk parses and a lexicon).

FeasPar is based on a principle of chunks, their features and relations. The FeasPar architecture consists of two major parts: A neural network collection and a search. The neural networks first spilt the incoming sentence into chunks. Then each chunk is labeled with feature values and chunk relations. Finally, the search uses a formal feature structure specification as constraint, and outputs the most probable and consistent feature structure.

FeasPar was trained, tested and evaluated with the Spontaneous Scheduling Task, and compared with a handmodeled LR-parser. FeasPar performed better than the LR-parser in all six comparison performance measures that were made.

References

George Berg. 1991. Learning Recursive Phrase Structure: Combining the Strengths of PDP and X-Bar Syntax. Technical report TR 91-5, Dept. of Computer Science, University at Albany, State University of New York.

Peter F. Brown, John Cocke, Stephen A. Della Pietra, Vincent J. Della Pietra, Fredrick Jelinek John D. Lafferty, Robert L. Mercer, and Paul S. Roossin. 1990. A Statistical Approach To Machine Translation. *Computational Linguistics*, 16(2):79–85, June.

Finn Dag Buø and Alex Waibel. 1996. Search in a Learnable Spoken Language Parser. In *Proceedings of the 12th European Conference on Artificial Intelligence*, August.

Finn Dag Buø. 1996. *FeasPar - A Feature Structure Parser Learning to Parse Spontaneous Speech*. Ph.D. thesis, University of Karlsruhe, upcoming.

J. Dowding, J. M. Gawron, D. Appelt, J. Bear, L. Cherny, R. Moore, and D. Moran. 1993. Gemini: A Natural Language System for Spoken-Language Understanding. In *Proceedings ARPA Workshop on Human Language Technology*, pages 43–48, Princeton, New Jersey, March. Morgan Kaufmann Publisher.

G. Gazdar, E. Klein, G. K. Pullum, and I. A. Sag. 1985. A theory of syntactic features. In *Generalized Phrase Structure Grammar*, chapter 2. Blackwell Publishing, Oxford, England and Harvard University Press, Cambridge, MA, USA.

Sunil Issar and Wayne Ward. 1993. CMU's robust spoken language understanding system. In *Proceedings of Eurospeech*.

Ajay N. Jain. 1991. *A Connectionist Learning Architecture for Parsing Spoken Language*. Ph.D. thesis, School of Computer Science, Carnegie Mellon University, Dec.

R. Kaplan and J. Bresnan. 1982. Lexical-Functional Grammar: A Formal System for Grammatical Representation. In J. Bresnan, editor, *The Mental Representation of Grammatical Relations*, pages 173–281. The MIT Press, Cambridge, MA.

A. Lavie and M. Tomita. 1993. GLR* - An Efficient Noise-skipping Parsing Algorithm for Context-free Grammars. In *Proceedings of Third International Workshop on Parsing Technologies*, pages 123–134.

R. Miikkulainen and M. Dyer. 1991. Natural Language Processing With Modular PDP Networks and Distributed Lexicon. *Cognitive Science*, 15:343–399.

C. Pollard and I. Sag. 1987. Formal Foundations. In *An Information-Based Syntax and Semantics*, chapter 2. CSLI Lecture Notes No.13.

Hinrich Schütze. 1993. Translation by Confusion. In *Spring Symposium on Machine Translation*. AAAI.

Stephanie Seneff. 1992. TINA: A Natural Language System for Spoken Language Applications. *Computational linguistics*, 18(1).

Stefan Wermter and Volker Weber. 1994. Learning Fault-tolerant Spreech Parsing with SCREEN. In *Proceedings of Twelfth National Conference on Artificial Intelligence*, Seattle.

M. Woszczyna, N. Aoki-Waibel, F. D. Buø, N. Coccaro, K. Horiguchi, T. Kemp, A. Lavie, A. McNair, T. Polzin, I. Rogina, C.P. Rose, T. Schultz, B. Suhm, M. Tomita, and A. Waibel. 1994. JANUS 93: Towards Spontaneous Speech Translation. In *International Conference on Acoustics, Speech & Signal Processing*, pages 345–348, vol. 1, Adelaide, Australia, April. IEEE.