Learning to Parse Spontaneous Speech

Finn Dag Buø and Alex Waibel

Interactive Systems Laboratories
University of Karlsruhe (Germany)
Carnegie Mellon University (USA)
finndag@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 two samples of a handmodeled GLR* parser, developed for 4 months and 2 years, respectively. The handmodeling effort for FeasPar is 2 weeks. FeasPar performes better than the GLR* parser developed 4 months in all six comparisons that are made and has a similar performance as the GLR* parser developed for 2 years.

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.[6, 10, 13], 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 [5, 15, 11, 8].

A small community have experimented with either purely statistical approaches[2, 14] or connectionist based approaches [1, 12, 9, 16]. Their main advantages are learnability and robustness. 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[6]. 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 suggestions 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, the evaluation set of 120 sentences, and the final evaluation set of 350 sentences (99 utterances). 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[7]. 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 output provided by a unification based parser.

This paper is organized as follows: First, we describe the parser architecture and how it works. Second, we describe the lexicon and the parser's neural aspects. Third, a search algorithm is motivated. Fourth, performance comparison measures are described. Then evaluation, results, and conclusion follow.

2. Parser Architecture

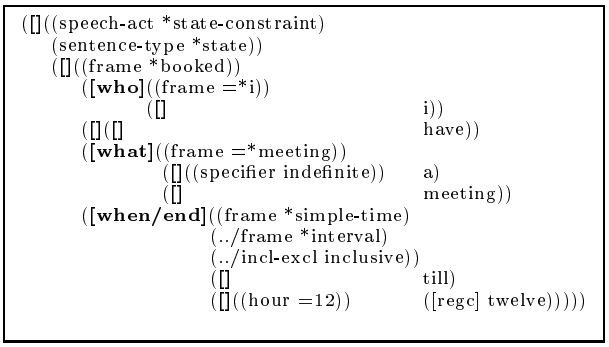

Figure 1: Chunk parse (chunk relations shown in **boldface**)

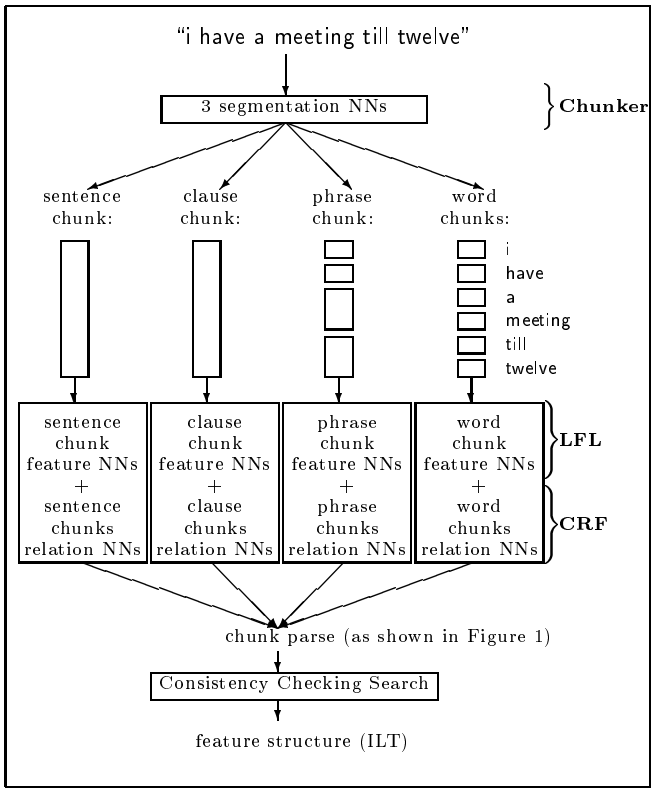

Figure 2: FeasPar's architecture for run mode.

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. In run mode, the input sentence is processed through all networks, giving a chunk parse, which is passed on to the Consistency Checking Search, see Section 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. In total, there are four levels of chunks: word/numbers, phrases, clauses and sentence.

The Linguistic Feature Labeler (LFL) 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. 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.

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

Further details on the three modules can be found in [4, 3].

2.1. Lexicon and Neural Architecture

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. The number and selection of microfeatures are domain dependent and must be made manually. For the English Spontaneous Scheduling Task (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. Further information on the lexicon can be found in [3].

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[9], where learning starts using a small context, which is increased later in the learning process. This causes local dependencies to be learned first. Further techniques for improving performance are described in [3]. For the neural networks, the average test set performance is 95.4 %.

3. Consistency Checking 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 [4, 3].

4. Performance Comparison

To show the learning ability of FeasPar, it is compared with the GLR* parser[7, 11], applying ESST as domain, and the JANUS speech translation system[7] as environment. An ESST GLR* semantic grammar only exists for English. Hence, FeasPar is trained with English.

4.1. PM 1: Parse Quality

The first performance measure, PM 1, expresses the parse quality. PM1 is also called ILT feature accuracy and is defined as:

$$\frac{C_{corr}-M}{C_{corr}}$$

where:

- M is the number of mismatches made, while checking all features in the correct ILTs *and* the suggested ILTs from the parser.
- C_{corr} is the number of considerations of existing features in the correct ILTs.

It is important to notice that this number can become negative: if a suggested ILT contains a feature A not present in the correct ILT, M is counted up, but C_{corr} is not. The latter is only incremented if feature A occurs in the correct ILT. The advantage of PM 1 is that the measure is computed automatically and is independent of human judgement. Its disadvantage is that it is only an indirect indicator for translation quality, since not all ILT features are equally important for the generator.

4.2. PM 2: Translation Quality

The second performance measure, PM 2, is also called the end-to-end comparison, where the quality of the translated sentences is measured: A translated sentence is graded as 'acceptable' if all relevant information is conveyed and the sentence is natural (i.e. perfect), or slightly unnatural, but clear enough to understand the meaning (i.e. ok). It is graded as 'not acceptable' if incorrect or not all information is conveyed, or (with speech data only) an irrecoverable recognition error occurs. Furthermore, trivial sentences, whose translations can easily be retrieved by lookup, are excluded, so that only truly translated sentences are counted. Two variants exist for PM2: PM 2E is used when the parser is coupled with an English generator and PM 2G, when the parser is combined with a German generator. The English generator was developed longer than the German generator, which is notable in performance. The advantage of measure 2 is that translation quality is measured directly. Its disadvantage is that the grading must be done by humans, i.e. it depends on human judgement, and is therefore subjective. Also, grading becomes a fairly time consuming task.

5. Evaluation

For the parsing GLR* grammars for ESST, two samples are selected: The first after 4 months of development, and the second after two years of development. We compare the first one with FeasPar by applying the first evaluation set. Results are shown in Table 1. As one can see, FeasPar performs better than the GLR* parser developed for 4 months in all six comparison performance measurements that are made.

	FeasPar	GLR* parser (4 months)
PM1 - T	71.8 %	51.6 %

	FeasPar	GLR* Parser (4 months)
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 %

Table 1: Comparing FeasPar with a GLR* parser hand modeled for 4 months, S=speech data, T=transcribed data).

In a second comparison, FeasPar is compared with a GLR* parser developed for 2 years. Evaluation material is another,

	FeasPar	GLR* Parser (2 years)
PM2E - T	75.1 %	78.6 %
PM2E - S	60.5 %	60.8 %

Table 2: Comparing FeasPar (old ILT) with a GLR* parser developed for 2 years.

larger evaluation set. For technical reasons, a comparison fair to the GLR* parser could only be made with performance measure 2E (see [3] for a discussion). All output is graded by an independent person, being native speaker and not involved in parser research or development. Grading results are shown in Table 2. On the larger evaluation set, FeasPar has a similar performance (60.5 % versus 60.8 %) as the GLR* grammar developed for 2 years, when measuring the performance for acceptable translations into English (PM2E) with speech input. One sees that for speech data, which is the situation for being used in an actual speech-to-speech translation system, FeasPar and the GLR* parser have practically the same performance.

6. 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 trained, tested and evaluated with the Spontaneous Scheduling Task, and compared with two samples of a handmodeled GLR* parser, developed for 4 months and 2 years, respectively. The handmodeling effort for FeasPar is 2 weeks. FeasPar performes better than the GLR* parser developed 4 months in all six comparisons that are made and has a similar performance as the GLR* parser developed for 2 years.

7. REFERENCES

1. George Berg. 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, 1991.
2. 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. A Statistical Approach To Machine Translation. *Computational Linguistics*, 16(2):79–85, June 1990.
3. Finn Dag Buø. *FeasPar - A Feature Structure Parser Learning to Parse Spontaneous Speech*. PhD thesis, University of Karlsruhe, upcoming 1996.
4. Finn Dag Buø and Alex Waibel. Search in a Learnable Spoken Language Parser. In *Proceedings of the 12th European Conference on Artificial Intelligence*, August 1996.
5. J. Dowding, J. M. Gawron, D. Appelt, J. Bear, L. Cherny, R. Moore, and D. Moran. Gemini: A Natural Language System for Spoken-Language Understanding. In *Proceedings ARPA Workshop on Human Language Technology*, pages 43–48, Princeton, New Jersey, March 1993. Morgan Kaufmann Publisher.
6. G. Gazdar, E. Klein, G. K. Pullum, and I. A. Sag. A theory of syntactic features. In *Generalized Phrase Structure Grammar*, chapter 2. Blackwell Publishing, Oxford, England and Harvard University Press, Cambridge, MA, USA, 1985.
7. P. Geutner, B. Suhm, F. D. Buø, T. Kemp, L. Mayfield, A. E. McNair, I. Rogina, T. Schultz, T. Sloboda, W. Ward, M. Woszczyna, and A. Waibel. Integrating Different Learning Approaches into a Multilingual Spoken Language Translation System. In *Workshop on New Approaches to Learning for Natural Language Processing, International Joint Conference on Artificial Intelligence*, Montreal, Canada, August 1995.
8. Sunil Issar and Wayne Ward. CMU's robust spoken language understanding system. In *Proceedings of Eurospeech*, 1993.
9. Ajay N. Jain. *A Connectionist Learning Architecture for Parsing Spoken Language*. PhD thesis, School of Computer Science, Carnegie Mellon University, Dec 1991.
10. R. Kaplan and J. Bresnan. 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, 1982.
11. A. Lavie and M. Tomita. GLR* - An Efficient Noise-skipping Parsing Algorithm for Context-free Grammars. In *Proceedings of Third International Workshop on Parsing Technologies*, pages 123–134, 1993.
12. R. Miikkulainen and M. Dyer. Natural Language Processing With Modular PDP Networks and Distributed Lexicon. *Cognitive Science*, 15:343–399, 1991.
13. C. Pollard and I. Sag. Formal Foundations. In *An Information-Based Syntax and Semantics*, chapter 2. CSLI Lecture Notes No.13, 1987.
14. Hinrich Schütze. Translation by Confusion. In *Spring Symposium on Machine Translation*. AAAI, 1993.
15. Stephanie Seneff. TINA: A Natural Language System for Spoken Language Applications. *Computational linguistics*, 18(1), 1992.
16. Stefan Wermter and Volker Weber. Learning Fault-tolerant Spreech Parsing with SCREEN. In *Proceedings of Twelfth National Conference on Artificial Intelligence*, Seattle, 1994.