A Method to Improve the Efficiency of Deep Parsers with Incremental Chart Pruning

Pierre Lison
Language Technology Lab,
DFKI GmbH,
Saarbrücken, Germany

Abstract

The use of deep parsers in spoken dialogue systems is usually subject to strong performance requirements. Real-time dialogue applications must be capable of responding quickly to any given utterance, even in the presence of noisy, ambiguous or distorted input. The parser must therefore ensure that the number of analyses remains bounded at every processing step.

The paper presents a practical approach to address this issue in the context of deep parsers designed for spoken dialogue. The approach is based on a word lattice parser for Combinatory Categorial Grammar combined with a discriminative model for parse selection. Each word lattice is parsed incrementally, word by word, and a discriminative model is applied at each incremental step to prune the set of resulting partial analyses. The model incorporates a wide range of linguistic and contextual features and can be trained with a simple perceptron. The approach is fully implemented as part of a spoken dialogue system for human-robot interaction. Evaluation results on a Wizard-of-Oz test suite demonstrate significant improvements in parsing time.

1 Introduction

Developing robust and efficient parsers for spoken dialogue is a difficult and demanding enterprise. This is due to several interconnected reasons.

The first reason is the pervasiveness of *speech recognition errors* in natural (i.e. noisy) environments, especially for open, non-trivial discourse domains. Automatic speech recognition (ASR) is indeed a highly error-prone task, and parsers designed to process spoken input must therefore find ways to accomodate the various ASR errors that may (and will) arise.

Next to speech recognition, the second issue we need to address is the *relaxed grammaticality* of spoken language. Dialogue utterances are often incomplete or ungrammatical, and may contain numerous disfluencies like fillers (err, uh, mm), repetitions, self-corrections, etc.

Finally, the vast majority of spoken dialogue systems are designed to operate in *real-time*. This has two important consequences. First, the parser should not wait for the utterance to be complete to start processing it – instead, the set of possible semantic interpretations should be gradually built and extended as the utterance unfolds. Second, each incremental parsing step should operate under strict time constraints. The main obstacle here is the high level of ambiguity arising in natural language, which can lead to a combinatorial explosion in the number of possible readings.

The remaining of this paper is devoted to addressing this last issue, building on an integrated approach to situated spoken dialogue processing previously outlined in (Lison, 2008; Lison and Kruijff, 2009). The approach we present here is similar to (Collins and Roark, 2004), with some notable differences concerning the parser (our parser being specifically tailored for robust spoken dialogue processing), and the features included in the discriminative model.

An overview of the paper is as follows. We first describe in Section 2 the cognitive architecture in which our system has been integrated. We then discuss the approach in detail in Section 3. Finally, we present in Section 4 the quantitative evaluations on a WOZ test suite, and conclude.

2 Architecture

The approach we present in this paper is fully implemented and integrated into a cognitive architecture for autonomous robots (Hawes et al.,

2007). It is capable of building up visuo-spatial models of a dynamic local scene, and continuously plan and execute manipulation actions on objects within that scene. The robot can discuss objects and their material- and spatial properties for the purpose of visual learning and manipulation tasks. Figure 1 illustrates the architecture schema for the communication subsystem.

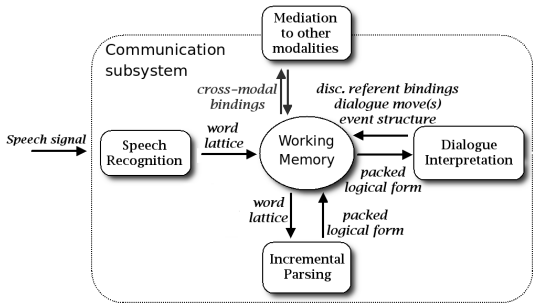

Figure 1: Architecture schema of the communication subsystem (only for comprehension).

Starting with ASR, we process the audio signal to establish a *word lattice* containing statistically ranked hypotheses about word sequences. Subsequently, parsing constructs grammatical analyses for the given (partial) word lattice. A grammatical analysis constructs both a syntactic analysis of the utterance, and a representation of its meaning. The analysis is based on an incremental chart parser[1] for Combinatory Categorial Grammar (Steedman and Baldridge, 2009). These meaning representations are ontologically richly sorted, relational structures, formulated in a (propositional) description logic – more precisely in the HLDS formalism (Baldridge and Kruijff, 2002). The incremental build of derivational structures is realised within the parser via type-raising and composition rules.

Once all the possible (partial) parses for a given (partial) utterance are computed, they are filtered in order to retain only the most likely interpretation(s). This ensures that the number of parses at each incremental step remains bounded and avoid a combinatorial explosion of the search space. The task of selecting the most likely parse(s) among a set of possible ones is called *parse selection*. We describe it in detail in the next section.

At the level of dialogue interpretation, the logical forms are resolved against a dialogue model to establish co-reference and dialogue moves.

Finally, linguistic interpretations must be associated with extra-linguistic knowledge about the environment – dialogue comprehension hence needs to connect with other subarchitectures like vision, spatial reasoning or planning.

[1]Built using the OpenCCG API: http://openccg.sf.net

3 Approach

3.1 Parse selection

As we just explained, the parse selection module is responsible for selecting at each incremental step a subset of "good" parses. Once the selection is made, the best analyses are kept in the (CKY-like) parse chart, while the others are discarded and pruned from the chart.

To achieve this selection, we need a mechanism to discriminate among the possible parses. This is done via a (discriminative) statistical model covering a large number of features.

Formally, the task is defined as a function $F : \mathcal{X} \rightarrow \mathcal{Y}$ where the domain $\mathcal{X}$ is the set of possible inputs (in our case, $\mathcal{X}$ is the set of possible *word lattices*), and $\mathcal{Y}$ the set of parses. We assume:

1. A function $\mathbf{GEN}(x)$ which enumerates all possible parses for an input x. In our case, the function represents the admissibles parses according to the CCG grammar.

2. A d-dimensional feature vector $\mathbf{f}(x, y) \in \Re^d$, representing specific features of the pair (x, y). It can include various acoustic, syntactic, semantic or contextual features.

3. A parameter vector $\mathbf{w} \in \Re^d$.

The function F, mapping a word lattice to its most likely parse, is then defined as:

$$F(x) = \underset{y \in \mathbf{GEN}(x)}{\operatorname{argmax}} \ \mathbf{w}^T \cdot \mathbf{f}(x, y) \qquad (1)$$

Given the parameters $\mathbf{w}$, the optimal parse of a given word lattice x can be therefore easily determined by enumerating all the parses generated by the grammar, extracting their features, computing the inner product $\mathbf{w}^T \cdot \mathbf{f}(x, y)$, and selecting the parse with the highest score.

3.2 Learning

3.2.1 Training data

To estimate the parameters $\mathbf{w}$, we need a set of training examples. Since no corpus of situated dialogue adapted to our task domain is available to this day – let alone semantically annotated – we

followed the approach advocated in (Weilhammer et al., 2006) and *generated* a corpus from a hand-written task grammar.

We first designed a small grammar covering our task domain, each rule being associated to a HLDS representation and a weight. Once specified, the grammar is then randomly traversed a large number of times, resulting in a large set of utterances along with their semantic representations.

3.2.2 Perceptron learning

The algorithm we use to estimate the parameters **w** using the training data is a **perceptron**. The algorithm is fully online - it visits each example in turn, in an incremental fashion, and updates **w** if necessary. Albeit simple, the algorithm has proven to be very efficient and accurate for the task of parse selection (Collins and Roark, 2004; Zettlemoyer and Collins, 2007).

The pseudo-code for the online learning algorithm is detailed in [**Algorithm 1**].

3.3 Features

As we have seen, the parse selection operates by enumerating the possible parses and selecting the one with the highest score according to the linear model parametrised by $\mathbf{w}$.

The accuracy of our method crucially relies on the selection of "good" features $\mathbf{f}(x, y)$ for our model - that is, features which help *discriminating* the parses. In our model, the features are of four types: semantic features, syntactic features, contextual features, and speech recognition features.

3.3.1 Semantic features

What are the substructures of a logical form which may be relevant to discriminate the parses? We define features on the following information sources: the nominals, the ontological sorts of the nominals, and the dependency relations (following (Clark and Curran, 2003)).

These features therefore help us handle various forms of lexical and syntactic ambiguities.

3.3.2 Syntactic features

Syntactic features are features associated to the *derivational history* of a specific parse. Alongside the usual CCG rules (application, composition and type raising), our parser also uses a set of non-standard (type-changing) rules designed to handle disfluencies, speech recognition errors, and combinations of discourse units by selectively relaxing the grammatical constraints (see (Lison and

Algorithm 1 Online perceptron learning

Require: - Set of n training examples $\{(x_i, z_i) : i = 1...n\}$
- For each incremental step j with $0 \leq j \leq |x_i|$, we define the partially parsed word lattice x_i^j and its gold standard semantics z_i^j
- T: number of iterations over the training set
- $\texttt{GEN}(x)$: function enumerating possible parses for an input x, according to the CCG grammar.
- $\texttt{GEN}(x, z)$: function enumerating possible parses for an input x and which have semantics z, according to the CCG grammar.
- $L(y)$ maps a parse tree y to its logical form.
- Initial parameter vector $\mathbf{w_0}$

% Initialise
$\mathbf{w} \leftarrow \mathbf{w_0}$
% Loop T times on the training examples
for $t = 1 \ldots T$ **do**
  for $i = 1 \ldots n$ **do**
    % Loop on the incremental parsing steps
    for $j = 0...|x_i|$ and if x_i not already updated **do**
      % Compute best parse according to model
      Let $y' = \text{argmax}_{y \in \mathbf{GEN}(x_i^j)} \mathbf{w}^T \cdot \mathbf{f}(x_i^j, y)$
      % If the decoded parse $\neq$ expected parse, update the parameters of the model
      if $L(y') \neq z_i^j$ **then**
        % Search the best parse for the partial word lattice x_i^j with semantics z_i^j
        Let $y^* = \text{argmax}_{y \in \mathbf{GEN}(x_i^j, z_i^j)} \mathbf{w}^T \cdot \mathbf{f}(x_i^j, y)$
        % Update parameter vector $\mathbf{w}$
        Set $\mathbf{w} = \mathbf{w} + \mathbf{f}(x_i^j, y^*) - \mathbf{f}(x_i^j, y')$
      end if
    end for
  end for
end for
return parameter vector **w**

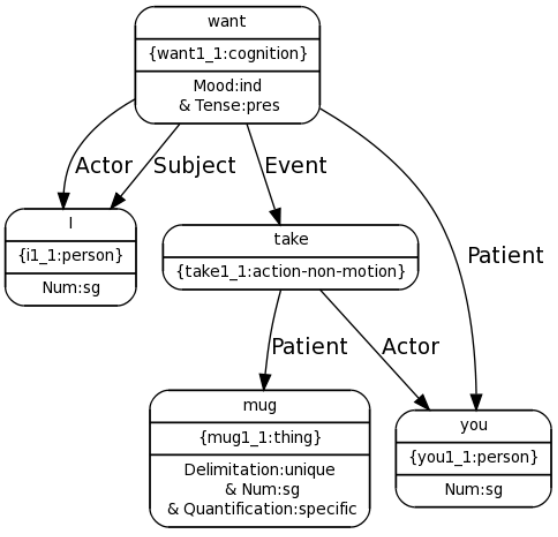

Figure 2: graphical representation of the HLDS logical form for "*I want you to take the mug*".

Kruijff, 2009) for details). In order to "*penalise*" to a correct extent the application of these non-standard rules, we include in the feature vector $\mathbf{f}(x, y)$ new features counting the number of times these rules are applied in the parse. In the derivation shown in the Figure 3, the rule *corr* (correction of a speech recognition error) is for instance applied once.

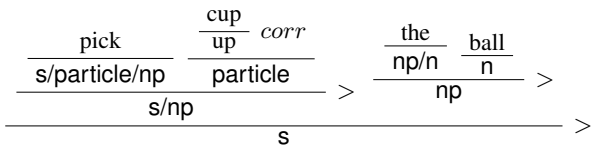

Figure 3: CCG derivation of *"pick cup the ball"*.

In the usual case, the perceptron will learn to assign negative weights to the syntactic features during the training process. In other words, these features can be seen as a *penalty* given to the parses using these non-standard rules, thereby giving a preference to the "normal" parses over them. This ensures that the grammar relaxation is only applied "as a last resort" when the usual grammatical analysis fails to provide a parse.

3.3.3 Contextual features

One striking characteristic of spoken dialogue is the importance of *context*. Understanding the visual and discourse contexts is crucial to resolve potential ambiguities and compute the most likely interpretation(s) of a given utterance.

The feature vector $\mathbf{f}(x, y)$ therefore includes various contextual features. Our dialogue system notably maintains in its working memory a list of contextually activated words (Lison and Kruijff, 2008). This list is continuously updated as the dialogue and the environment evolves. For each context-dependent word, we include one feature counting its occurrence in the utterance.

3.3.4 Speech recognition features

Finally, the feature vector $\mathbf{f}(x, y)$ also includes features related to the *speech recognition*. The ASR module outputs a set of (partial) recognition hypotheses, packed in a word lattice (Figure 4).

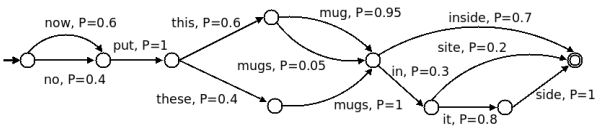

Figure 4: Example of word lattice

We want to favour the hypotheses with high confidence scores, which are, according to the statistical models incorporated in the ASR, more likely to reflect what was uttered. To this end, we introduce in the feature vector several acoustic features measuring the likelihood of each recognition hypothesis.

3.4 Incremental chart pruning

In the previous subsections, we explained how the parse selection was performed, and on basis of which features. We now briefly describe how it can be used for incremental chart pruning.

The main idea is to specify a *beam width* parameter in the parser. This beam width defines the maximal number of analyses which can be kept in the chart at each incremental step. If the number of possible readings exceeds the beam width, the analyses with a lower parse selection score are removed from the chart. Practically, this is realised by removing the top signs associated in the chart with the set of analyses to prune, as well as all the intermediate signs which are included in these top signs *and* are not used in any of the analyses retained by the parse selection module.

The combination of incremental parsing and incremental chart pruning provides two decisive advantages over classical, non-incremental parsers: first, we can start processing the spoken inputs as soon as a partial analysis can be outputted by the ASR. Second, the pruning mechanism ensures that each parsing step remains time-bounded. It is therefore ideally suited for spoken dialogue systems used in human-robot interaction.

4 Experimental evaluation

We performed a quantitative evaluation of our approach, using its implementation in a fully integrated system (cf. Section 2). To set up the experiments, we gathered a Wizard-of-Oz corpus of human-robot spoken dialogue for our task-domain, segmented and annotated manually with their expected semantic interpretation. The data set contains 195 individual utterances [2] along with their complete logical forms.

The results are shown in the Table 1. We tested our approach for five different values of the beam width parameter. The results are compared against a baseline, which is the performance of our parser

[2] More precisely, word lattices provided by the ASR, containing up to 10 alternative recognition hypotheses.

	Beam width	Average parsing time (in s.)	Exact-match			Partial-match		
			Precision	Recall	F_1-value	Precision	Recall	F_1-value
(Baseline)	*(none)*	*10.1*	*40.4*	*100.0*	*57.5*	*81.4*	*100.0*	*89.8*
	120	5.78	40.9	96.9	57.5	81.9	98.0	89.2
	60	**4.82**	41.1	92.5	56.9	81.7	94.1	87.4
	40	4.66	39.9	88.1	54.9	79.6	91.9	85.3
	30	4.21	41.0	83.0	54.9	80.2	88.6	84.2
	20	4.30	40.1	80.3	53.5	78.9	86.5	82.5

Table 1: Evaluation results (in seconds for the parsing time, in % for the exact- and partial-match).

without chart pruning. For each configuration, we give the average parsing time, as well as the exact-match and partial-match results (in order to verify that the performance increase is not cancelled by a drop in accuracy). We observe that the choice of the beam width parameter is crucial. Above 30, the chart pruning mechanism works very efficiently – we observe a notable decrease in the parsing time without significantly affecting the accuracy performance. Below 30, the beam width is too small to retain all the necessary information in the chart, and the recall quickly drops.

5 Conclusions

In this paper, we presented an original method to improve the efficiency of deep parsers (in particular, parsers for categorial grammars) with an incremental chart pruning mechanism used to limit at every processing step the number of analyses retained in the parse chart.

The incremental chart pruning mechanism is based on a discriminative model exploring a set of relevant semantic, syntactic, contextual and acoustic features extracted for each parse. At each incremental step, the discriminative model yields a score for each resulting parse. The parser then only retains in its chart the set of parses associated with a high score, the others being pruned.

As forthcoming work, we shall examine the extension of our approach in new directions, such as the introduction of more refined contextual features or the use of more sophisticated learning algorithms such as Support Vector Machines.

6 Acknowledgements

This work was supported by the EU FP7 ICT Integrated Project "CogX" (FP7-ICT- 215181).

References

J. Baldridge and G.-J. M. Kruijff. 2002. Coupling CCG and hybrid logic dependency semantics. In *ACL'02: Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 319–326, Philadelphia, PA. Association for Computational Linguistics.

S. Clark and J. R. Curran. 2003. Log-linear models for wide-coverage ccg parsing. In *Proceedings of the 2003 conference on Empirical methods in natural language processing*, pages 97–104, Morristown, NJ, USA. Association for Computational Linguistics.

M. Collins and B. Roark. 2004. Incremental parsing with the perceptron algorithm. In *ACL '04: Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics*, page 111, Morristown, NJ, USA. Association for Computational Linguistics.

N. A. Hawes, A. Sloman, J. Wyatt, M. Zillich, H. Jacobsson, G.-J. M. Kruijff, M. Brenner, G. Berginc, and D. Skocaj. 2007. Towards an integrated robot with multiple cognitive functions. In *Proc. AAAI'07*, pages 1548–1553. AAAI Press.

P. Lison and G.-J. M. Kruijff. 2008. Salience-driven contextual priming of speech recognition for human-robot interaction. In *Proceedings of the 18th European Conference on Artificial Intelligence*, Patras (Greece).

P. Lison and G.-J. M. Kruijff. 2009. An integrated approach to robust processing of situated spoken dialogue. In *Proceedings of the International Workshop on Semantic Representation of Spoken Language (SRSL'09)*, Athens, Greece. (to appear).

P. Lison. 2008. Robust processing of situated spoken dialogue. Master's thesis, Universität des Saarlandes, Saarbrücken. http://www.dfki.de/∼plison/pubs/thesis/main.thesis.plison2008.pdf.

M. Steedman and J. Baldridge. 2009. Combinatory categorial grammar. In Robert Borsley and Kersti Börjars, editors, *Nontransformational Syntax: A Guide to Current Models*. Blackwell, Oxford.

K. Weilhammer, M. N. Stuttle, and S. Young. 2006. Bootstrapping language models for dialogue systems. In *Proceedings of INTERSPEECH 2006*, Pittsburgh, PA.

L. S. Zettlemoyer and M. Collins. 2007. Online learning of relaxed CCG grammars for parsing to logical form. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 678–687.