An Open Source Environment for Compiling Typed Unification Grammars into Speech Recognisers

Manny Rayner, Beth Ann Hockey and John Dowding
RIACS, Mail Stop T27A–2
NASA Ames Research Center
Moffett Field, CA 94035-1000, USA
{mrayner, bahockey, jdowding}@riacs.edu

Abstract

We present REGULUS, an Open Source environment which compiles typed unification grammars into context free grammar language models compatible with the Nuance Toolkit. The environment includes a large general unification grammar of English and corpus-based tools for creating efficient domain-specific recognisers from it. We will demo applications built using the system, including a speech translator and a command and control system for a simulated robotic domain, and show how the development environment can be used to edit and extend them.

1 Introduction

This demo presents REGULUS, an Open Source environment that supports efficient compilation of typed unification grammars into speech recognisers. The basic intent is to provide a set of tools to support rapid prototyping of spoken dialogue applications in situations where little or no corpus data exists. The environment has already been used to build over half a dozen applications with vocabularies of between 100 and 500 words. We will be demoing some of them here, along with examples of how the development environment can be used to edit and extend them.

The rest of this document is organised as follows. Section 2 describes the grammar formalism and the grammar-to-recogniser compiler process. This includes a tool which allows construction of domain-specific grammars by corpus-based specialisation of a large general unification grammar for English. Section 3 describes the interactive development environment. Section 4 describes how we intend to structure the actual demo. The final section contains instructions for downloading and installing REGULUS.

2 Compiling Unification Grammars into Recognisers

The core functionality provided by the REGULUS environment is compilation of typed unification grammars into annotated context-free grammar language models expressed in Nuance Grammar Specification Language (GSL) notation (Nuance, 2002). GSL language models can be converted into runnable speech recognisers by invoking the Nuance Toolkit compiler utility, so the net result is the ability to compile a unification grammar into a speech recogniser.

The REGULUS unification grammar formalism is closely related to the one used in the SRI Core Language Engine (CLE) and Gemini systems (Alshawi, 1992; Dowding et al., 1993), and it is most reasonable to compare it with Gemini, which offers a broadly similar range of functionalities. One important difference relates to the treatment of semantics. CLE and Gemini support a general unification-based semantics; REGULUS, however, does not permit the use of unification when constructing semantic representations. Although this involves a slight loss of expressive power, it offers the very significant advantage of allowing the

semantics of the original grammar to be compiled into semantic annotations on the target GSL rules. The resulting recogniser can thus be used as a stand-alone system, which both recognises spoken utterances and produces semantic representations for them. In contrast, recognisers compiled by the Gemini system only produce word-string output, which then has to be parsed by a separate process.

The basic algorithm used by REGULUS to compile unification grammars into CFG form is described in (Rayner et al., 2001). The central idea is simply to perform an enumerative expansion of the unification grammar by non-deterministically instantiating each feature to all of its possible values; the resulting grammar is then filtered to remove non-reachable rules. As described in (Rayner et al., 2001), the efficiency of the naive algorithm can be greatly improved by adding a pre-processing step which performs a suitable factoring of the grammar. The current version of REGULUS further refines the naive method by iteratively alternating the expansion and filtering stages, non-deterministically expanding each feature in turn and then filtering the result before proceeding to the next feature. On large grammars, this "iterative expansion" technique can reduce time and space requirements of the compilation algorithm by several orders of magnitude. Use of iterative expansion has allowed REGULUS successfully to compile several grammars which exceeded resource bounds for the Gemini compiler (Moore et al., 1997; Moore, 1998).

2.1 Using Grammar Specialisation

Experience with grammar-based spoken dialogue systems shows that there is usually a substantial overlap between the structures of grammars for different domains. This is hardly surprising, since they all ultimately have to model general facts about the linguistic structure of English and other natural languages. It is consequently natural to consider strategies which attempt to exploit the overlap between domains by building a single, general grammar valid for a wide variety of applications. A grammar of this kind will probably offer more coverage (and hence lower accuracy) than is desirable for any given specific application. It is however feasible to address the problem using corpus-based techniques which extract a specialised version of the original general grammar.

REGULUS implements a version of the grammar specialisation scheme which extends the Explanation Based Learning method described in (Rayner et al., 2002). There is a general unification grammar, loosely based on the Core Language Engine grammar for English (Pulman, 1992), which has been developed over the course of about ten individual projects. The current version of the grammar contains 145 unification grammar rules, 465 function word entries, and 72 features. The grammar for each individual domain also includes a domain-specific content word lexicon, which typically contains 100 to 500 lexical entries. We have intentionally used a variety of widely differing domains, including a command and control system for a simulated mobile robot, a speech enabled home automation system, a travel planning system and a medical speech translator.

The semantic representations produced by the grammar are in a simplified version of the Core Language Engine's Quasi Logical Form notation (van Eijck and Moore, 1992). Thus for example the representation of "turn on the fan" is

```
[[imp,
  form(imperative,
       [[turn,
         term(pro,you,[]),
         term(the_sing,fan,[]),
         on]]
       )
]]
```

and that for " how long have you had headaches" is

```
[[whq,
  form([present,perfect],
       [[have_symptom,
         term(pro,you,[]),
         term(null,headache,[])],
        [duration,how_long]]
       )
]]
```

A grammar built on top of the general grammar is transformed into a specialised Nuance grammar in the following processing stages:

1. The training corpus is converted into a "treebank" of parsed representations. This is done using a left-corner parser representation of the grammar.

2. The treebank is used to produce a "raw" specialised grammar in REGULUS format, using the EBL algorithm (van Harmelen and Bundy, 1988; Rayner, 1988). The granularity of the learned rules is determined by a user-supplied parameter. This parameter can currently be set to the following values:

   Lexical The learned grammar is completely "flat", with each training example producing exactly one rule.

   NP The learned grammar has two levels, with only NP nodes between root nodes and pre-lexical nodes.

   NP_PP The learned grammar has three levels, with NP and PP nodes between root nodes and pre-lexical nodes.

   S_NP_PP The learned grammar has four levels, with S, NP and PP nodes between root nodes and pre-lexical nodes.

3. The "raw" specialised grammar is post-processed into the final specialised grammar. The post-processing stage consists of three steps:

   (a) Duplicate rules are merged, keeping the different training examples as documentation.

   (b) Specialised rules are sorted by number of training examples.

   (c) If there are enough training examples for a rule, it is further constrained to unify with the the least common generalisation of all the contexts in which it has occurred. The threshold which determines when this happens is defined by another user-supplied parameter.

4. The final specialised grammar is compiled into a Nuance GSL grammar.

For the applications we have so far worked with, output GSL grammars vary in size from about 500 to about 15000 GSL rules. Compilation times on a 2GHz PC vary from a few seconds to about 10 minutes, mainly depending on the size of the training corpus. Word error rates on in-coverage material for the resulting recognisers vary from about 5% to about 15%. Interestingly, it appears that there is no very strong correlation between recognition performance and either vocabulary size or size of the training corpus, with performance depending rather more heavily on average utterance length and the types of constructions covered by the specialised grammar. We are actively investigating these issues at the moment.

3 The Development Environment

All the functionalities in the REGULUS environment are available via a command-line interface, and also from within a top-loop designed primarily for interactive grammar debugging. In this mode, the grammar is compiled into an efficient left-corner parser, using the algorithm of (Moore, 2000), and also into a Definite Clause Grammar (DCG) form; the advantage of the DCG representation is that it can often be used to help diagnose grammar bugs by attempting to parse non-top constituents. Each separate domain-specific grammar is defined though a config file, which also specifies settings for the various user-defined parameters relevant to the compilation process.

4 Structure of the demo

We will demo two recognisers built using REGULUS, one for a command and control application and one for a medical speech translation application. The command and control recogniser is an extended version of the one for the Personal Satellite Assistant, a speech enabled mobile robot currently being developed at the NASA Ames Research Center (PSA, 2002). The domain-specific lexicon contains 313 entries; coverage includes commands ("go to flight deck"), Y-N and WH-questions ("has the temperature increased during the last five minutes", "when did you measure the pressure at storage lockers"), conjunctions ("what were oxygen and carbon dioxide five minutes ago") and pronouns ("go to crew hatch and open it"). Word error rates for this application on in-coverage data are around 10–11%.

The medical speech translator recogniser is an extended English-language version of the system described in (Rayner and Bouillon, 2002). The domain-specific lexicon for this system contains 607 entries. Coverage includes Y-N and WH-questions ("does the headache usually start suddenly", "what makes your headache better") and elliptical phrases ("insomnia", "in the left chest", "severe"). Although the medical speech application has nearly twice as large a vocabulary, recognition performance is in fact noticeably better than for the command and control application; preliminary results suggest a word error rate on in-coverage data of around 6–7%.

We will demo speech recognition for each recogniser, and show how the development environment can be used to make rapid changes to coverage by adding lexical entries and/or training examples.

5 Downloading and installation

REGULUS requires the Nuance Toolkit (Nuance, 2002) and SICStus Prolog (Programming Systems Group, 1995), and runs under Windows 2000 and NT. As a minimum hardware configuration, we recommend a 400MHz machine with 256M of RAM. The whole system, including source code, documentation and examples, is publicly available and can used freely under the terms of the Lesser General Public Licence (LGPL). The current release can be obtained as a zip-file by mailing the authors, or by download from SourceForge at http://sourceforge.net/projects/leonlp/.

References

H. Alshawi, editor. 1992. *The Core Language Engine.* MIT Press, Cambridge, Massachusetts.

J. Dowding, M. Gawron, D. Appelt, L. Cherny, R. Moore, and D. Moran. 1993. Gemini: A natural language system for spoken language understanding. In *Proceedings of the Thirty-First Annual Meeting of the Association for Computational Linguistics.*

R. Moore, J. Dowding, H. Bratt, J. Gawron, Y. Gorfu, and A. Cheyer. 1997. CommandTalk: A spoken-language interface for battlefield simulations. In *Proceedings of the Fifth Conference on Applied Natural Language Processing*, pages 1–7.

R. Moore. 1998. Using natural language knowledge sources in speech recognition. In *Proceedings of the NATO Advanced Studies Institute.*

R. Moore. 2000. Improved left-corner chart parsing for large context-free grammars. In *Proceedings of the 6th International Workshop on Parsing Technologies*, pages 171–182.

Nuance, 2002. http://www.nuance.com. As of 15 November 2002.

Programming Systems Group, 1995. *SICStus Prolog User's Manual.* Swedish Institute of Computer Science, Kista, Sweden.

PSA, 2002. *Personal Satellite Assistant (PSA) Project.* http://ic.arc.nasa.gov/ic/projects/psa/. As of 1 Feb 2002.

S.G. Pulman. 1992. Syntactic and semantic processing. In Alshawi (Alshawi, 1992), pages 129–148.

M. Rayner and P. Bouillon. 2002. A phrasebook style medical speech translator. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics (demo track)*, Philadelphia, PA.

M. Rayner, J. Dowding, and B.A. Hockey. 2001. A baseline method for compiling typed unification grammars into context free language models. In *Proceedings of Eurospeech 2001*, pages 729–732, Aalborg, Denmark.

M. Rayner, B.A. Hockey, and J. Dowding. 2002. Grammar specialisation meets language modelling. In *Proceedings of the 7th International Conference on Spoken Language Processing (ICSLP)*, Denver, CO.

M. Rayner. 1988. Applying explanation-based generalization to natural-language processing. In *Proceedings of the International Conference on Fifth Generation Computer Systems*, pages 1267–1274, Tokyo, Japan.

J. van Eijck and R. Moore. 1992. Semantic rules for English. In H. Alshawi, editor, *The Core Language Engine*, pages 83–116. MIT Press.

T. van Harmelen and A. Bundy. 1988. Explanation-based generalization = partial evaluation (research note). *Artificial Intelligence*, 36:401–412.