SUPPLE: A Practical Parser for Natural Language Engineering Applications

Robert Gaizauskas, Mark Hepple, Horacio Saggion,
Mark A. Greenwood and Kevin Humphreys[*]
Department of Computer Science
University of Sheffield, Sheffield, UK
{robertg|hepple|saggion|m.greenwood|-}@dcs.shef.ac.uk

Abstract

We describe SUPPLE, a freely-available, open source natural language parsing system, implemented in Prolog, and designed for practical use in language engineering (LE) applications. SUPPLE can be run as a stand-alone application, or as a component within the GATE General Architecture for Text Engineering. SUPPLE is distributed with an example grammar that has been developed over a number of years across several LE projects. This paper describes the key characteristics of the parser and the distributed grammar.

1 Introduction

In this paper we describe SUPPLE[1] — the Sheffield University Prolog Parser for Language Engineering — a general purpose parser that produces both syntactic and semantic representations for input sentences, which is well-suited for a range of LE applications. SUPPLE is freely available, and is distributed with an example grammar for English that was developed across a number of LE projects. We will describe key characteristics of the parser and the grammar in turn.

2 The SUPPLE Parser

SUPPLE is a general purpose bottom-up chart parser for feature-based context free phrase structure grammars (CF-PSGs), written in Prolog, that has a number of characteristics making it well-suited for use in LE applications. It is available both as a language processing resource within the GATE General Architecture for Text Engineering (Cunningham et al., 2002) and as a standalone program requiring various preprocessing steps to be applied to the input. We will here list some of its key characteristics.

Firstly, the parser allows multiword units identified by earlier processing components, e.g. named entity recognisers (NERs), gazetteers, etc, to be treated as non-decomposable units for syntactic processing. This is important as the identification of such items is an essential part of analyzing real text in many domains.

The parser allows a layered parsing process, with a number of separate grammars being applied in series, one on top of the other, with a "best parse" selection process between stages so that only a subset of the constituents constructed at each stage is passed forward to the next. While this may make the parsing process incomplete with respect to the total set of analyses licensed by the grammar rules, it makes the parsing process much more efficient and allows a modular development of sub-grammars.

Facilities are provided to simplify handling feature-based grammars. The grammar representation uses flat, i.e. non-embedded, feature representations which are combined used Prolog term unification for efficiency. Features are predefined and source grammars compiled into a full form representation, allowing grammar writers to include only relevant features in any rule, and to ignore feature ordering. The formalism also permits disjunctive and optional right-hand-side constituents.

The chart parsing algorithm is simple but very

[*] At Microsoft Corporation since 2000 (Speech and Natural Language Group). Email: kevinhum@microsoft.com.

[1] In previous published materials and in the current GATE release the parser is referred to as buChart. This is name is now deprecated.

Proceedings of the Ninth International Workshop on Parsing Technologies (IWPT), pages 200–201,
Vancouver, October 2005. ©2005 Association for Computational Linguistics

efficient, exploiting the characteristics of Prolog to avoid the need for active edges or an agenda. In informal testing, this approach was roughly ten times faster than a related Prolog implementation of standard bottom-up active chart parsing.

The parser does not fail if full sentential parses cannot be found, but instead outputs partial analyses as syntactic and semantic fragments for user-selectable syntactic categories. This makes the parser robust in applications which deal with large volumes of real text.

3 The Sample Grammar

The sample grammar distributed with SUPPLE has been developed over several years, across a number LE projects. We here list some key characteristics.

The morpho-syntactic and semantic information required for individual lexical items is minimal — inflectional root and word class only, where the word class inventory is basically the PTB tagset.

A conservative philosophy is adopted regarding identification of verbal arguments and attachment of nominal and verbal post-modifiers, such as prepositional phrases and relative clauses. Rather than producing all possible analyses or using probabilities to generate the most likely analysis, the preference is to offer a single analysis that spans the input sentence only if it can be relied on to be correct, so that in many cases only partial analyses are produced. The philosophy is that it is more useful to produce partial analyses that are correct than full analyses which may well be wrong or highly disjunctive. Output from the parser can be passed to further processing components which may bring additional information to bear in resolving attachments.

An analysis of verb phrases is adopted in which a core verb cluster consisting of verbal head plus auxiliaries and adverbials is identified before any attempt to attach any post-verbal arguments. This contrasts with analyses where complements are attached to the verbal head at a lower level than auxiliaries and adverbials, e.g. as in the Penn TreeBank. This decision is again motivated by practical concerns: it is relatively easy to recognise verbal clusters, much harder to correctly attach complements.

A semantic analysis, or simplified quasi-logical form (SQLF), is produced for each phrasal constituent, in which tensed verbs are interpreted as referring to unique events, and noun phrases as referring to unique objects. Where relations between syntactic constituents are identified in parsing, semantic relations between associated objects and events are asserted in the SQLF.

While linguistically richer grammatical theories could be implemented in the grammar formalism of SUPPLE, the emphasis in our work has been on building robust wide-coverage tools — hence the requirement for only minimal lexical morphosyntactic and semantic information. As a consequence the combination of parser and grammars developed to date results in a tool that, although capable of returning full sentence analyses, more commonly returns results that include chunks of analysis with some, but not all, attachment relations determined.

4 Downloading SUPPLE Resources

SUPPLE resources, including source code and the sample grammar, and also a longer paper providing a more detailed account of both the parser and grammar, are available from the supple homepage at:

`http://nlp.shef.ac.uk/research/supple`

5 Conclusion

The SUPPLE parser has served as a component in numerous LE research projects, and is currently in use in a Question Answering system which participated in recent TREC/QA evaluations. We hope its availability as a GATE component will facilitate its broader use by NLP researchers, and by others building applications exploiting NL technology.

Acknowledgements

The authors would like to acknowledge the support of the UK EPSRC under grants R91465 and K25267, and also the contributions of Chris Huyck and Sam Scott to the parser code and grammars.

References

H. Cunningham, D. Maynard, K. Bontcheva, and V. Tablan. GATE: A framework and graphical development environment for robust NLP tools and applications. *Proceedings of the 40th Anniversary Meeting of the Association for Computational Linguistics*, 2002.