R-parse: A Natural Language Processing Application of R

Over the last several years, the fields of Natural Language Processing and Computational Linguistics have shifted their focus toward statistical methods, emphasizing probabilistic grammars, Expectation Maximization, Maximum Entropy, and Log-linear modeling frameworks. These developments make R an attractive environment in which to explore the development of new computational linguistic models. One such model involving a straightforward application of R is Latent Semantic Analysis/Indexing (LSA/LSI), which employs Singular Value Decomposition (SVD) to perform a principal components-style analysis of term frequencies in documents. This application is readily handled by writing wrapper functions that perform specialized handling of the built-in matrix analysis and plotting functions of R.

Yet, an impediment to the widespread use of R in NLP/CL research is the lack of good facilities for the handling of traditional symbolic processing tasks such as language parsing and generation. The fact that R is a member of the LISP family of programming languages suggests that this could be done, since most symbolic NLP work has used declarative languages like LISP and Prolog. At the same time, it remains to be seen whether R itself is flexible enough to permit useful parsers to be written without resorting to external language extensions.

This paper reports on efforts to develop a simple but flexible chart-based parser in R called R-parse. Chart parsers are preferred in NLP applications, as they minimize the redundant work of the parser by employing a data structure for recording attempted parses. R-parse is implemented as a recursive function that returns a parsed string in the form of a chart. Charts and grammar rules are both implemented as data frames, hence both parser output and grammar are amenable to statistical analysis using functions such as `glm()` (e.g. for log-linear modeling of the parsed output). Furthermore, a probabilistic generator function R-gen accepts the same form of grammar as R-parse. Together, the parser and generator allow one to model and study all aspects of the relationship between probabilistic grammars and the languages they generate entirely within the resources of R. The properties of R-parse and R-gen can presently be demonstrated on modest but realistic grammars and language training corpora.

An obstacle for the development of R-parse has turned out to be R's relatively impoverished meta-programming facilities. Such features are necessary for implementing exotic execution control schemes needed in parsers (e.g. backtracking) without resorting to an entirely redesigned stack. Much of this work might be handled with some form of environment-passing, but unfortunately, in its present implementation, R's environment-passing is not powerful enough as it provides no way to selectively prevent the modification of variables in an environment. Hence intervening computations can "spoil" the outcome of a computing on an environment that has since been inadvertently modified. An improved meta-programming environment in R, perhaps made possible by exposing more of R's internal workings in the fashion of `delay()`, might permit more elegant solutions to these problems. In the mean time, a working version of R-parse exists which correctly records in the chart all possible parses of a string, including failed and incomplete parses, for both context-free and regular grammars.