Using Toolsets and Architectures To Build NLP Systems

Centre Universitaire, Luxembourg
5 August 2000

Table of Content

Author Index

Program Committee

Rémi Zajac (Chair), CRL, New-Mexico State University, USA: zajac@crl.nmsu.edu.

Jan Amtrup, CRL, New-Mexico State University, USA: jamtrup@crl.nmsu.edu.

Stephan Busemann, DFKI, Saarbrücken: busemann@dfki.de.

Hamish Cunningham, University of Sheffield: hamish@dcs.shef.ac.uk.

Guenther Goerz, IMMD VIII, University of Erlangen: goerz@immd8.informatik.uni-erlangen.de.

Gertjan van Noord, University of Groningen: vannoord@let.rug.nl.

Fabio Pianesi, IRST, Trento: pianesi@irst.itc.it.

Using Toolsets and Architectures To Build NLP Systems

Centre Universitaire, Luxembourg
5 August 2000

Many toolsets have been developed to support the implementation of single NLP components (taggers, parsers, generators, dictionaries) or complete Natural Language Processing applications (Information Extraction systems, Machine Translation systems). A source for available toolkits is the Natural Language Software Registry, an initiative of the Association for Computational Linguistics hosted by DFKI at `http://registry.dfki.de`. These tools aim at facilitating and lowering the cost of building NLP systems. Since the tools themselves are often complex pieces of software, they require a significant amount of effort to be developed and maintained in the first place. Is this effort worth the trouble? It is to be noted that NLP toolsets have often been originally developed for implementing a single component or application. In this case, why not build the NLP system using a general programming language such as Lisp or Prolog? There can be at least two answers. First, for pure efficiency issues (speed and space), it is often preferable to build a parameterized algorithm operating on a uniform data structure (e.g., a phrase-structure parser). Second, it is harder, and often impossible, to develop, debug and maintain a large NLP system directly written in a general programming language.

It has been the experience of many users that a given toolset is quite often unusable outside its environment: the toolset can be too restricted in its purpose (e.g. an MT toolset that cannot be used for building a grammar checker), too complex to use, or even too difficult to install. There have been, in particular in the US under the Tipster program, efforts to promote instead common architectures for a given set of applications (primarily IR and IE in Tipster; see also the Galaxy architecture of the DARPA Communicator project). Several software environments have been built around this flexible concept, which is closer to current trends in main stream software engineering.

The workshop aims at providing a picture of the current problems faced by developers and users of toolsets, and future directions for the development and use of NLP toolsets. It includes reports of actual experiences in the use of toolsets as well as presentation of toolsets and application development.

Rémi Zajac, Computing Research Laboratory, New Mexico State University
`zajac@crl.nmsu.edu`