Open Resources for Language Technology

Lars Degerstedt and Arne Jönsson

Department of Computer and Information Science
Linköping University, SE-581 83, LINKÖPING, SWEDEN
larde@ida.liu.se arnjo@ida.liu.se

Abstract

NLPFARM is an Open Source code repository for development and sharing of language technology resources. NLPFARM hosts a number of projects covering various language technology needs, providing possibilities to develop more robust and well-formed applications. NLPFARM has been in use for more than a year and our experience is that it has facilitated co-operation and sharing of resources but that there are still issues to consider.

1. Introduction

Ideally, the research community of language technology should work towards one and the same goal, concerning its software – a shared code library that can be used for sharing research ideas and serve as a platform for commercial efforts and applications. Furthermore, new research results must be readily available to industry in order for new techniques to be useful. Often such dissemination has been made through the development of prototypes showing the possibilities and prospects of the current research frontier. Such prototypes illustrate the potential, but very often the code itself has been fragile and unreliable and often lack a proper API (Olsson and Gambäck, 2000).

Another possibility is to provide a repository of frameworks, tools, and linguistic resources in the form of publicly available facility software ready to be used and re-used in industry projects of today and tomorrow. This puts new demands on both design and robustness of the research software. In order to be useful the design must have a generic character, i.e. it must be possible to use new software without too much extra effort. Furthermore, the development strategy must be chosen carefully so that robustness of code is eventually achieved. Finally, the design is preferably open and the code well documented.

To push the level of research software forward it is important to stimulate interactions between different projects, its usage and methods of development. By putting more emphasis on project evolution, research systems and language technology modules will eventually evolve beyond a certain point into artifacts useful for industry (Lehman and Ramil, 2001). In fact, the evolutionary perspective on systems, designs and resources also place new issues on the language technology research agenda.

Language technology systems and frameworks tend to have a proprietary character (with a few important exceptions such as the DARPA initiative cf. (Aberdeen et al., 1999)). This state of affairs means that it is difficult to distribute research results to industry and also that it is difficult for researchers to keep in touch with parallel development of ideas of software in industry, and other parts of the society.

We find ourselves in a situation where we want to:

- hand over existing preliminary prototypes into a phase of more robust software constructions
- keep our research software compliant with the network-based software technology to be useful for industry
- find competent computing competence that are willing to engage in our projects
- use our systems in an industrial environment and find users that are truly interested in our results and therefore give us feedback that help us direct our own research work.

At this point we turn to the open source community and see that they seem to solve several of these problems for us, if we only join their activities.

Definitions of Open Source are somewhat shallow and need more fundamental analysis of the phenomena. In particular, the following aspects of Open Source are interesting from the perspective of development methodology:

- publicity: development is open to public scrutiny since communication is done through open channels such as mailing lists, discussion forums and web sites.
- community cooperation: development is conducted by a community of independent groups that jointly develop software as the result of pursuing their different goals within the same framework (technical or application purpose).
- user-driven design: the members of the user community essentially become the developers of the content. Development tends to become just-in-time, relevance-based and bottom-up.

Moreover, Open Source projects are not really planned but rather grow as a result of the emerging needs of the many different users and the desires of individuals. Resulting from the directional flexibility is adaptation to changing requirements. In an Open Source community the ability to change is built into the individuality of development. Everyone is a change-prone individual making decisions on their own, going their own way. In its extreme, Open Source is a non-directional development process. Though most Open Source projects have an application vision, total freedom of choice is present within the boundaries of that vision and the vision may also change over time.

2. nlpFarm

NLPFARM, is an open source community that has been developed as a powerful means for dissemination of research results through new evolving facility software developed following the open-source criteria. We suggest focusing on the software development process and means to facilitate this process instead of "reuse via a standardised language data storage, management and visualisation" (Cunningham, 1999). With useful facility software modules available, i.e. code that can be used without too much effort, NLPFARM encourage development of language technology that has previously been only available by exemplification design in research prototypes. The goal is to iteratively develop more and more robust and effective facility software, which in turn makes the end-application thinner and thus leaves room for more powerful features to be added.

The NLPFARM site consists of two parts: a web resource for users of the facility software, and a project page for facility software developers and administration[1]. The site contains dowloadable release files, project documentation, and the latest source code available on-line[2].

2.1. Infrastructure

The NLPFARM is organised as a collection of Java software projects, each with their own independent piece of software, download files and documentation. We have developed a project work model that contains straight-forward rules for project management, development, and the community life. Moreover, efforts have been made to simplify a uniform treatment of development and project builds. In particular, a tool for automatically creating all required project files, projectConfig, has been created. Using the tool simplifies compilation, debugging, as well as release file and documentation generation. In this way, project evolution is facilitated, and thus also the ability to follow up new needs coming in from client applications that use the facility software of NLPFARM.

2.2. Resources

NLPFARM is constantly growing and changing. Currently (February 2004) the site hosts 11 projects:

Tools and Frameworks Tools and frameworks comprise projects that are building blocks for a variety of language technology applications and situations.

- QUAC: QUAC is a framework for the interface between a q/a, or dialogue, system and background knowledge sources. Currently QUAC supports access to relational databases and to HTTP resources. In addition, for query-based resources it offers a template-based query generation scheme.
- GUIDIA: The GUIDIA project provides high-level dialogue system components which works under different widget libraries. Currently the components work on both Java AWT and Swing.
- MODI: The MODI project enables users to interact wireless with dialog systems. This is achieved by utilising the Java MIDP libraries, which are supported on modern cell phones and PDA:s.
- JAVACHART: JAVACHART is a chart parser with the following features: 1) it uses syntax similar to the well-known PATR-II parser. 2) it supports partial parsing of a sentence 3) it contains both a command-line mode and a server Java API for use in applications. 4) it supports compilation of resources where syntax-checking is done once and intermediate code is placed on file.
- MOLINC: a collection of dialogue system components for re-use of computational phases, and strategies of a dialogue system.

Demo System This project is used to exemplify how other projects at the farm can be used in this sample application.

- TVGUIDE: TVGUIDE is an information-providing system where users can type natural language queries about movies. The current information source is the Internet Movie Database (http://us.imdb.com), and (from version 0.2.0 and on) Swedish TV channel tableau (such as http://www.tv.nu). The focus of the project is not so much on dialogue coverage but rather on principal designs and use of tools.

Libraries Fundamental NLP-modules that form a platform that can be used both by the more advanced projects and directly by applications.

- NLPLIB: The NLPLIB project contains basic NLP packages useful for development of NLP tools and applications. The library contains a Factory based implementation of feature structures.
- JBRICKSLIB: The library contains various useful utility snippet classes, implementations of well-known design patterns, and general support classes such as configuration management. The aims of JBRICKSLIB are: 1) to facilitate for other projects to use well-known design patterns in their code. 2) to encourage to use generic solutions in all NLPFARM projects for similar problems. 3) to become a fairly complete Java platform tailored for the needs of the NLP libraries and applications of study at the NLPFARM.
- JGRLIB: The project contains Java implementations of various sorts of grammars. Right now, ordinary context-free grammars and their probabilistic friends, PCFG, are implemented. The grammars come with parsers and, in the case of PCFG, an implementation of the Inside-Outside training algorithm.
- JAVACONLIB: JAVACONLIB is a set of library functions useful for test and development of algorithms using word contexts.

[1] See http://nlpfarm.sourceforge.net and http://sourceforge.net/projects/nlpfarm.

[2] The latest version of the source code is available at all times by the SourceForge file handling system, i.e. CVS.

Support A kind of meta activity containing projects for support of development of NLP facility and application software.

- ADMIN: The project contains various resource files, scripts and programs that supports management of the other projects at the NLPFARM. In particular, it contains the configuration tool projectConfig for easy handling of the administrative files for development and release of open source projects.

As the software modules at NLPFARM become more mature we intend to also package selected projects as an official NLPFARM toolkit release. The user will then be able to download the resources both as a unit, and as separate parts.

3. Using nlpFarm

We have been using NLPFARM for more than a year, and have developed a variety of projects, (Johansson, 2004, provides an overview) based on the methodology of incremental and iterative development (Degerstedt and Jönsson, 2001). In this section we present one application being developed using NLPFARM resources.

The BIRDQUEST system is an NLP question-answering system about birds (Andén et al., 2004). The main focus of BIRDQUEST is on utilising information extraction techniques for dialogue systems and on using an ontology. The system have full support for basic dialogue capabilities, such as clarification sub-dialogues and focus management, based on empirically-collected dialogue corpora. The current information source is a MySQL Database. The system is the first documented dialogue system application, besides the demo system TVGUIDE, that uses the NLPFARM facility software.

Initially BIRDQUEST was developed in parallel with the activities on NLPFARM, and consequently, some frameworks were not available from the beginning. BIRDQUEST used the JavaChart-parser for parsing user requests from the start. The grammar and lexicon needed for BIRDQUEST were new but a variety of constructions from the demo system TVGUIDE could be adopted.

The experiences from the BIRDQUEST project gave feedback that led to further development of the NLPFARM facility software. The BIRDQUEST system uses a design pattern called phase graph processor (PGP) for the NLP part of the system (Degerstedt and Johansson, 2003). The PGP facility software module was further refined during the development BIRDQUEST. Focus resolution and generation are based on extendable object-oriented facility modules. The modules could be strengthened by gradual generalization of ideas from the BIRDQUEST application code. The database lookup module used facility software. The combination of ordinary database processing with tests and combination of results based on an ontology module gave birth to future extensions in facility software. In particular, the projects NLPLIB, QUAC, and MOLINC have been further refined and extended, as a consequence of this work. BIRDQUEST was also re-factored to utilise these new, more stable and robust, modules.

4. Experience

The NLPFARM resource has been in use for more than a year and our experience is that it has facilitated cooperation and sharing of resources but that there are still issues to consider, especially regarding various types of NLP software developers.

Design and implementation We note that finding the right conceptualization has been the key to good interface design. The use of design patterns has been very helpful as means for this, especially since our approach is incremental and bottom-up. Modularisation has been done on several levels, most importantly perhaps the choice to encapsulate the software in "projects", i.e. stand-alone release entities. This forces the development and the code to behave in a structural way. The major re-use factor so-far is the use of design patterns, as discussed above. For knowledge representation we have so far focused on representation on the Java-object-level (i.e. not the XML level etc). Extensive use of (common) interfaces for encapsulation of concrete classes which makes the representation robust, e.g. as in the case for structures where we use an (adjusted) interface from the OpenNLP initiative[3]

Methodology Our research work related to the NLPFARM has used an evolutionary development process, developing applications and facility software as individual projects but with frequent interaction between the projects. No facility software at the NLPFARM has yet reached a more stable and mature status, but we expect to reach at least 1.0 versions of the most important projects at the end of the year 2004. The basic iterative cycle of the facility software at NLPFARM is that each phase comprises a new release of a project. Functionality is incrementally added to the system in a cumulative way in each release. Following the evolutionary philosophy of the agile methodology (including Open Source), the contents of NLPFARM have been built bottom-up in separate independent pieces. Dependencies between facility software projects themselves are also encouraged, but should be one way, and libraries are only allowed to depend on other libraries.

Community An evolutionary process such as NLPFARM also means a community activity. In the case of a language technology resource such as the NLPFARM process it has been a mix of software builders, linguistic and interface experts and domain experts. The SourceForge community has given the NLPFARM community its major channels for communication. This has led to download of our results and email contacts with other interested parties. The NLPFARM has become a part of the NLP community formed under the OpenNLP initiative, and we will integrate our efforts under the forthcoming OpenNLP Java platform. We have initiated discussions of and made initial attempts for re-using common APIs within the OpenNLP forum. OpenNLP is still immature, though there are currently 14 member projects. The organisation is informal in its character, since it is run in the bottom-up style of Open Source.[4]

[3] See http://opennlp.sourceforge.net for more details on the OpenNLP umbrella project.

[4] In particular, one can note that there has been suggestions within OpenNLP to learn from the both positive and negative ex-

Usage-feedback has come mostly from application projects in direct relation with the NLPFARM and exchange between the NLPFARM projects themselves, so far. However, there has also been concrete suggestions of improvements coming from independent parties. The re-use between projects is a suitable initial test bench, since these projects are developed as separate release-entities. In terms of research results, our method for dialogue system development takes a step towards incorporating Open Source ideas, by further examination of the incremental aspects of the method (Johansson et al., 2002).

5. Discussion

During the work with NLPFARM we have gained new insights that will be used to refine our working process and software designs further.

That application artefacts and facility software are kept distinct is crucial as the basis of the software refinement process. For an application, the functional coverage relative to a domain and language models are the essential issues. For the facility software, representation formats and generic design are the focus points during development. Thus, the needs of the NLPFARM projects themselves are different from the needs of their clients, the NLP applications – on all three levels of development, methodology and community.

Differences in professional background tend to determine the order of how developers prefer to work. Thus, in a multi-disciplinary environment the method must support a multitude of iterative strategies to exist in parallel. We have also learnt the importance of discrimination of beginners (typically last year undergraduates in our case) with more experienced system developers (senior graduates or people with industrial experiences). The beginner must be able to work with minimal overhead from methodology and other support routines. Their strength lies in the fresh eyes on the problems at hand, and they often have more time to spend but in a shorter interval. The senior developer must take care of leverage of a platform on which the beginners can work, as well as re-package results in more robust formats.

Just as pair programming has been found to be a success in Extreme Programming (Beck, 2000), we have found that development should be done jointly in small groups (two-three persons is often optimal) working with NLP aspects of a system. Working alone with development tends to diverge the work and thereby decrease re-use, and too large groups makes the overhead of communication too heavy. The two-three group size is especially important for more technically complex phenomena, such as insuring that a dialogue system is coherently built in all its executional phases. Note that programming in groups does not mean that the developers must be located in the same room. Through NLPFARM we have means that facilitate geographically distributed programming.

Initially it has been natural to work bottom-up with the software of the NLPFARM. However, recently we have placed more and more focus on how to handle language resources, in order to avoid too much divergence between the projects. This indicates that the NLPFARM process will eventually become an intertwined approach consisting of both a bottom-up software development and a top-down knowledge representation.

periences of the GATE project.

Acknowledgment

This research is financed by Vinnova, Swedish Agency for Innovation Systems.

6. References

Aberdeen, John, Sam Bayer, Sasha Caskey, Lauire Damianos, Alan Goldschen, Lynette Hirschman, Dan Loehr, and Hugo Trappe, 1999. Implementing practical dialogue systems with the darpa communicator architecture. In *Proceedings of IJCAI-99 Workshop on Knowledge and Reasoning in Practical Dialogue Systems, August, Stockholm*.

Andén, Frida, Lars Degersedt, Annika Flycht-Eriksson, Arne Jönsson, Magnus Merkel, and Sara Norberg, 2004. Experiences from Combining Dialogue System Development with Information Access Techniques. In Mark T. Maybury (ed.), *New Directions in Question Answering*. AAAI/MIT Press.

Beck, Kent, 2000. *Extreme Programming Explained*. Addison-Wesley.

Cunningham, Hamish, 1999. A Definition and Short History of Language Engineering. *Natural Language Engineering*, 5(1):1–16.

Degerstedt, Lars and Pontus Johansson, 2003. Evolutionary Development of Phase-Based Dialogue Systems. In *Proc. of the 8th Scandianvian Conference on Artificial Intelligence*. Bergen, Norway.

Degerstedt, Lars and Arne Jönsson, 2001. Iterative Implementation of Dialogue System Modules. In *Proceedings of Eurospeech 2001, Aalborg, Denmark*.

Johansson, Pontus, 2004. Design and development of recommender dialogue systems. Licentiate Thesis 1079, Linköping Studies in Science and Technology, Linköping University.

Johansson, Pontus, Lars Degerstedt, and Arne Jönsson, 2002. Iterative Development of an Information-Providing Dialogue System. In *Proceedings of 7th ERCIM Workshop*.

Lehman, M. M. and J. F. Ramil, 2001. An Approach to a Theory of Software Evolution. In *Proc. of the 4th int. workshop on Principles of software evolution*. Vienna, Austria.

Olsson, Fredrik and Björn Gambäck, 2000. Composing a general-purpose toolbox for swedish. In *Proceedings of Using Toolsets and Architectures to Build NLP Systems a workshop held in conjunction with The 18th International Conference on Computational Linguistics (COLING 2000). Centre Universitaire, Luxembourg*.