LIRICS

Deliverable 5.2.C

LIRICS Reference Implementation

Project reference number	e-Content-22236-LIRICS
Project acronym	LIRICS
Project full title	Linguistic Infrastructure for Interoperable Resource and Systems
Project contact point	Laurent Romary, INRIA-Loria 615, rue du jardin botanique BP101. 54602 Villers lès Nancy (France) romary@loria.fr
Project web site	http://lirics.loria.fr
EC project officer	Erwin Valentini
Document title	LIRICS Reference Architecture
Deliverable ID	**D5.2.C**
Document type	Report
Dissemination level	Public
Contractual date of delivery	M30
Actual date of delivery	24 August 2007
Status & version	Draft
Work package, task & deliverable responsible	WP5, USFD
Author(s) & affiliation(s)	Adam Funk (USFD)
Additional contributor(s)	Julien Nioche, Niraj Aswani (USFD), Gil Francopolou (INRIA) Thierry Declerck (DFKI)
Keywords	LIRICS Reference Architecture

Document evolution

version	date	version	date
1.0	30/06/06		
1.9	24/08/07		
2.0	13/09/07		

1 Introduction

This brief document accompanies and explains the demonstrator-prototype software deliverable D5.2.C, which consists of the reference implementations of the MAF and SynAF services for various languages.

This document covers the services for Bulgarian, English and French on the one hand, and for German, Spanish and Italian on the other hand. It has been decided internally to the project, that DFKI would take over those last three language languages.

But at some time in the development, we discovered/remembered at DFKI that the Technical Annex of LIRICS specified the delivery of open source software. Not having been aware of this before, we had been using licensed tools for generating the Morpho-Syntactic annotation, and so cannot provide (at least for the time being) for the same kind of services as has been done for the other language, but can generate he required annotation on licensed products. This is also the reason for having a separated section. Thierry Declerck (DFKI) takes the responsibility for this misunderstanding. In the next future we will re-implement the MAF implementation using open source tools, which we already do for the SynAF implementation. The MAF-SynAF annotation can be demonstrated at the review, and request.

2 Requirements

The delivered software is packed into two web application archive file, **maf-service.war** and **synaf-service.war**, which require Apache Tomcat but contain all the relevant GATE libraries.

The Bulgarian MAF service additionally requires the third-party TreeTagger software, installed on the server with the configuration files for the Bulgarian language.

For the MAF services, we have been using a combination of statistical POS Tagging (the TNT software, see http://www.coli.uni-saarland.de/~thorsten/tnt/), which is trainable on virtually any tagsets, and the Mmorph engine for morphological analysis. (see http://www.lt-world.org/onto/onto?sorted=Division1.3_Technologies&GroupLanguageAnalysis&Originally_Ascending=MORPHOLOGICALANALYSIS)

3 Installation of the MAF service for Bulgarian, English and French

The software is delivered as one web-service archive file, **maf-service.war**, which should be placed in the **tomcat/webapps/** directory. When Tomcat is restarted it will automatically extract the application into the **tomcat/webapps/maf-service/** directory, which will contain all the necessary resources for the English and French MAF services and most of the resources for the Bulgarian service.

The TreeTagger software and its Bulgarian data files can be obtained from this website

http://www.ims.uni-stuttgart.de/projekte/corplex/TreeTagger/DecisionTreeTagger.html

and should be installed according to the instructions in the GATE manual

http://gate.ac.uk/sale/tao/index.html#x1-2040009.7

but with respect for the relative paths specified in the GATE application file, **maf-service/webapp/WEB-INF/MAF-bg.gapp**; in particular the directory **webapps/maf-service/WEB-INF/plugins/TreeTagger/** should contain the TreeTagger's **bin/** and **lib/** directories. It is important to ensure that the shell script **tree-tagger-bulgarian** and the three files in **bin/** have the correct (readable and executable) permissions for Tomcat to execute them. The file **bulgarian.par** should be installed in **lib/**.

For internal use within the LIRICS project, the maf-service.war file includes the TreeTagger. It is still necessary, however, to set the executable permissions on the four files listed above, *after* Tomcat unpacks the archive.

The working MAF service will have the local URL

http://localhost:8080/maf-service/services/MAFservice

although the port number will vary according to the Tomcat installation.

4 Implementation of the MAF service for Bulgarian, English and French

This MAF service consists of a GATE-based web service which contains three GATE applications (or pipelines) as follows.

The pipeline for English consists of the following processing resources (PRs):

- ANNIE Tokeniser,
- ANNIE Sentence Splitter,
- ANNIE POS Tagger,
- GATE Morphological analyser (lemmatiser), and
- GateAnnots2MafAnnotsPR.

The first four are standard GATE components which tokenise and split the document text and add POS (part-of-speech) tags and lemmata. The final PR converts the preceding PRs' annotations into MAF-compliant ones in a separate annotation set.

The Bulgarian MAF service consists of a similar pipeline, but the tagging is carried out by the University of Stuttgart's TreeTagger using a Bulgarian data file produced for the LIRICS project from the Bulgarian Treebank (http://www.bultreebank.org/).

The French MAF pipeline contains a PR developed for the LIRICS project which carries out tokenisation, sentence-splitting and POS-tagging and produces MAF-compliant output. It shares most of its code with the French SynAF PR explained in Section 6.2 below.

5 Implementation of the MAF service for German, Spanish and Italian

As mentioned in the introduction, DFKI took over for those languages, since we had a unified set of POS tagging and morphological analysers Unfortunately we have done this work suing tools that need a license (forgetting about the open sources requirements). So that we cannot provide for an open service right now.

The tools used are a combination of the statistical Tagger TNT and extended version of the Mmorph morphological analyzer developed originally at ISSCO in Switzerland.

6 Installation of the SynAF service for Bulgarian, English and French

The software is delivered as one web-service archive file, **synaf-service.war**, which should be placed in the **tomcat/webapps/** directory. When Tomcat is restarted it will automatically extract the application into the **tomcat/webapps/synaf-service/** directory, which will contain all the necessary resources for the Bulgarian, English and French services.

The working SynAF service will have the following local URL

http://localhost:8080/synaf-service/services/SYNAFservice

although the port number may vary according to the Tomcat installation.

7 Implementation of the SynAF services

The SynAF service for Bulgarian, English and French consists of a GATE web service containing three processing resources (PRs). Each time the service is called, it executes the appropriate PR according to the language code specified in the web-service call.

7.1 Implementation for Bulgarian and English

The English and Bulgarian PRs consist of wrappers around the Stanford Parser, a probabilistic parser implemented in Java. (http://nlp.stanford.edu/software/lex-parser.shtml)

The English PR uses the lexicalised parser dataset provided by Stanford NLP with the parser code and compliant with the tagset used by the Penn treebank.

The Bulgarian PR uses a dataset produced for the LIRICS project from a version of the Bulgarian Treebank transformed into a format similar to that of the Penn treebank.

The GATE PR also reads a data file listing the relevant treebank's tags and their equivalents in SynAF and uses it to map the parser's tags accordingly.

7.2 Implementation for French

The MAF (i.e. ISO-24611) and SynAF (i.e. ISO-24615) services for French are based on TagParser that is fully described at www.tagmatica.com.

TagParser comprises the following main steps:

1. morphological analysis for simple words and multi-words expressions,
2. chunking based on machine learning techniques,
3. syntactic relation computation based on 14 local grammars.

The MAF service is basically a two phases process where first a call to the sequence morphological analysis and chunking is made. Secondly, the result is mapped onto morpho-syntactic marks that are the ones recorded in the ISO data category registry (see morpho-syntactic profile in http://syntax.inist.fr/).

Let's note that for French, it has been proved since the French GRACE evaluation campaign in 1998, that a good robust parser is better than a good tagger for tagging. The rationale made by the winner was that in French:

1. many phrases are longer than a three word window (the window size of taggers like TreeTagger or Brill's one).
2. long distance attachment is rather frequent.

The SynAF service comprises a full parsing. The results conform to SynAF for the structure and to syntactic values taken from the ISO data category registry (see syntactic profile in http://syntax.inist.fr/).

The whole code is written in Java version 1.5. It has been tested on Java version 1.5 and version 1.6 on Windows XP. TagParser is not a prototype written in a lab. TagParser is written according to professional methods and requirements. TagParser is currently used in several industrial contexts, in fact mainly in the airplane industry. TagParser has been evaluated within the Technolangue/Easy campaign (see www.limsi.fr/Recherche/CORVAL/easy) and, currently (Fall 2007) is one of the parsers that is selected for the National ANR/PASSAGE campaign that compares all modern French parsers in the Giga-words range (see http://atoll.inria.fr/passage).

7.3 Implementation for German, Spanish and Italian

For those languages we used For the SynAF services we adapted the SCHUG shallow parser, developed at DFKI. Work was devoted to build an interface to the output of the MAF services. SCHUG is a tool written in Perl that can be parameterized for various input types (distinct tagsets of morpho-syntactic analysis), and apply on the top of this constituency and dependency analysis for the 3 languages concerned. SCHUG implements a cascade of chunk analysis, building clauses out of phrases and sentences out of clauses. Parallel to the construction of constituents, SCHUG builds the relational framework we call dependency in SynAF.

The multilingual aspects are covered by a lattice of agreement features, which have been translated to the MAF data categories, and specialised grammars.