



Project no. 027391

Project acronym: LT4eL

Project title: Language Technology for eLearning

Instrument Specific Targeted Research Project

Thematic Priority Information Society Technology

D2.3 Documented glossary candidate detector and integration report (1st cycle)

Due date of deliverable: 30-11-2006

Actual submission date: 21-12-2006

Start date of project: 1-12-2005

Duration: 30 Months

Organisation name of lead contractor for this deliverable: University of Tübingen (UTU)

Revision [1]

Project co-funded by the European Commission within the Sixth Framework Programme (2002-2006)		
Dissemination Level		
PU	Public	x
PP	Restricted to other programme participants (including the Commission Services)	
RE	Restricted to a group specified by the consortium (including the Commission Services)	
CO	Confidential, only for members of the consortium (including the Commission Services)	

D2.3 Documented glossary candidate detector and integration report

Contents

- 1 Introduction
- 2 Approach
- 3 Architecture
- 4 A tool for the processing of the grammars
- 5 The grammars for definitory contexts
- 6 Development and implementation cycles
- 7 Integration of the tool
- 8 Evaluation strategy
- 9 References
- 10 Appendices
 - 10.1 Use Cases
 - 10.1.1 Author generates glossary for a learning object
 - 10.1.2 User searches definitions in learning objects
 - 10.2 User Manual
 - 10.3 Related deliverables

Introduction

In the *Language Technology for eLearning* (LT4eL) project, we address one of the major problems users of LMSs will be confronted with: how to retrieve learning content from an LMS.

With the glossary candidate detector, we address the semi-automatic identification and extraction of definitions, including the defined terms. These data are useful for glossary compilation as well as input to the ontology construction which is the main target of WP3.

Glossaries are an important kind of secondary index to a text. They can be seen as small lexical resources which support the reader in decoding the text and understanding the central concepts which are conveyed. A glossary can be built on the definitory contexts which are presented in the learning objects themselves.

It is therefore necessary to identify definitory contexts in new texts. This is the major task of the glossary candidate detector. This software will serve:

- authors of learning objects who want to compile a glossary
- users of learning objects who look for a definition of a term which they come across (see the Use Cases in the Appendix for further details)

Approach

Our approach to the detection and extraction of definitory contexts is

1. rule-based
2. language specific

3. based on linguistically annotated documents (see Deliverable D2.1 for further details)

We assume that definitory contexts instantiate a limited number of grammatical patterns. The prototypical definition consists of a defined term (definiendum) and a definitory text (definiens), like in the following simple example:

A window is an opening in an otherwise solid and opaque surface through which light and, sometimes, air can pass

Our approach is to define rules which match grammatical patterns that are supposed to be definitions (e.g. NP 'is' NP, as in the above example). It is language dependent, because grammatical patterns for definitory contexts differ from language to language. In the project, we are therefore developing a grammar for each individual language. In the literature about extraction of definitory contexts, good results of the rule-based approach have been reported for English (cf. [1]). The strength of this project is that we develop a processing tool and grammars for all languages of the project (Bulgarian, Czech, Dutch, English, German, Polish, Portuguese and Romanian) and evaluate the results for all of these languages.

The challenges of this task are:

1. to find the relevant patterns for definitory contexts in all languages;
2. to find a good tradeoff between recall and precision (the software should extract as many definitions as possible and as few non-definitions as possible);
3. to identify the defined term of each definitions, even if this term is remote from the defined text.

We reply to these challenges with an empirical approach:

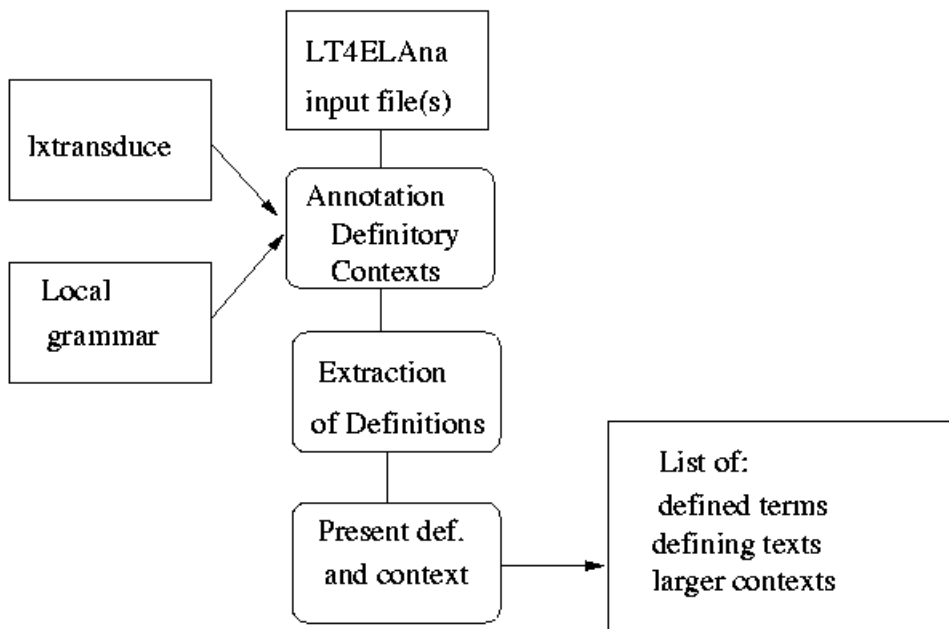
1. First, definitions have been marked in learning objects of all languages
2. Based on these data, grammar rules have been drafted
3. With these grammars, definitory contexts are identified and extracted automatically
4. These automatically extracted definitions are compared with the manually annotated definitions in the same texts
5. Based on the results of this comparison, the grammars are refined
6. The refined grammars are applied to new learning objects.

In the following, we describe the components of the software and the format of the local grammars. We will describe the way in which the tool is integrated into the learning management system and we will outline an evaluation strategy. In the appendix, we explain how the tool in its current stage can be used.

Architecture

The software we describe in this report consists of the following pieces:

1. A grammar for definitory contexts for each language
2. A language independent module for the application of the grammars to texts
3. A language independent module for the extraction of the definitions



The software consists of the following components: a) a module which handles and analyses the input files. Currently, these files must be linguistically annotated in the format which has been defined within the project (LT4ELAna). Annotation comprises part of speech and morpho-syntactic description for the tokens, and, optionally, annotation of noun chunks; b) a tool which processes the local grammars. This tool and the format of the local grammars are described in the following section; c) a module which extracts the definitions which are marked in the text, together with a context of a defined length.

The data flow is as follows: The user of the tool provides a file or a set of files and, for the second use case, a search term and a language tag. The grammar processor matches the grammar against the annotated learning objects. If a suitable grammatical pattern is detected, the corresponding XML subtree is enclosed in a *definingText* element. The defined term is enclosed in a *definedTerm* element. Following this step, the defined term and the defining text, together with the surrounding context of a defined length, are extracted and presented to the user.

It has been decided to implement the main components with Java.

A tool for the processing of the grammars

Given the constraints of the project and its primary task, i.e. to develop useful NLP functions for Learning Management Systems, the development of a grammar processing engine within the project was not feasible. Instead, a tool had to be selected which fulfils at least the following requirements:

- it should be XML-aware, i.e., it should recognise and mark up XML document trees and subtrees,
- the tool should be able to identify those (sub)trees on the basis of (sequences of) particular elements and particular attribute / text values of these elements,
- the tool should be sufficiently fast,
- it should support the Unicode character set,
- it should be freely available or be distributed under the GPL or one of its derivatives,
- it should be easy to integrate into an architecture which is mainly built on Java code.

Three alternative choices have been evaluated on the background of the project requirements:

1. XQuery engines as pure and general-purpose XML processing tools. On first look XQuery seems to be capable of matching local grammars to sequences of elements in an XML (sub)tree. On second look, it turned out that the formulation of constraints on sequences of elements is too complicated and not always possible to the degree which is needed. General-purpose XML processing tools are not well suited for linguistic analyses;
2. JAPE as an engine for the processing of regular expressions within GATE (URL: [http://www.gate.ac.uk/jape/](#)). The tool however is not well documented and adequate grammars for this task soon turned out to be too complicated;
3. Finally we choose to adapt *lxtransduce* for our project (cf [3] for a documentation of the tool): "lxtransduce is an XML transducer, intended for use in NLP (natural language processing) applications. XPath-based rules are matched against elements in the input document, and when one matches a corresponding rewrite is done." Our experiments revealed that the tool is powerful enough to detect and match sequences of linguistically annotated elements, to apply constraints on these sequences, and to wrap these sequences in the analysed text.

The grammars for definitory contexts

Lxtransduce supplies a format for the development of grammars which are matched against either pure text or XML documents. The grammars are XML documents which must conform to a DTD (*lxtransduce.dtd* is part of the software).

With lxtransduce, grammars can be developed which express constraints over sequences and trees of linguistic elements in the input files. In each grammar, there is one "main" rule which calls other rules by referring to them.

As an example, we cite the main rule of the Dutch grammar:

```
<rule name="main" wrap="def_context">
<first>
  <ref name="is_are_def"/>
  <ref name="other_defining_words_def"/>
  <ref name="bij_NP_def"/>
</first>
</rule>
```

If this rule matches a subtree of the XML document, this subtree is wrapped into a "def_context" element. The rule must match one of three alternatives, where the element "first" means that the shortest match of the three rules is taken.

The following example is one of the rules which are referred to in the main rule:

```
<rule name="is_are_def">
<seq>
  <ref name="defined_term"/>
  <query match="tok[@ctag='V' and @base='zijn' and @msd[starts-with(.,'hulpofkopp,ott')]]"/>
  <ref name="adv" mult="?"/>
  <first>
    <ref name="noun_phrase" />
    <query match="tok[@ctag='Pron' and @base='iemand']" />
    <ref name="art" />
  </first>
  <ref name="tok_or_chunk" mult="*"/>
</seq>
</rule>
```

This rule matches a sequence of elements which are further constrained. XQuery style queries are used to match single elements.

Rules can be extended by references to lexical entries. Grammatical classes can thus be constrained to a fixed set of lexical items. The following rule from the Bulgarian grammar is an example of a rule which refers to a lexical resource in its constraint section.

```
<rule name="other-verbs">
  <lookup lexicon="lex" match="w" phrase="true">
    <constraint test="cat='other-verbs'"/>
  </lookup>
</rule>
```

Development and implementation cycles

The development of the glossary candidate detector proceeds in the following steps: a) definitory contexts (i.e., defined terms and defining text) are annotated in the reference LOs of each language. This work is divided into three cycles. In each cycle, at least 150 definitory contexts are to be annotated for each language. b) based on the examples from the first step, local grammars for definitory contexts in each language are drafted; c) the output of these grammars, as applied to the annotated learning objects, is compared to the manual annotation. The results of this comparison give hints to the improvement of the local grammars (see also the section about the evaluation below); d) The grammars will be applied to new learning objects, and the results of the extraction will be evaluated by humans.

Integration of the tool

One of the tasks is to make the tool available within the ILIAS Learning Management System which is the reference LMS in the project. The coupling of the tools however should be such that they can also be integrated into other learning management systems and other application. The strategy is therefore to organize the communication between the learning management system and the natural language processing tools via web services. The interfaces are open and will be published, so that other applications can use these services.

The full picture is given in the integration report. We therefore give a rough outline of the integration of the tool with the ILIAS Learning Management System (and other potential software clients)

- The tool will be placed on the language technology server of the project, provided with a webservice interface
- The Learning Management System (and possible other clients) will communicate with the tool using this interface
- The tool will receive a document, a language code which specifies the language of the document, and, in the context of the second use case, a search term
- the tool will return a list of definitory contexts and defined terms.

Evaluation strategy

It is essential to evaluate the software and its impact to eLearning in general. Therefore, we need a tool-centered evaluation and a user-centered evaluation. In the following we will outline the tool-centered evaluation. The user-centered evaluation will take place once the tools are intergrated into the Learning Management System. The evaluation of the tool will be analogous to the evaluation of the keyword extractor (cf. Deliverable report 2.2). It will comprise of two methods. The first method is an automatic procedure: first, the extracted definitory contexts are matched against the manually annotated defining texts. A tool for the

automatic evaluation of the glossary candidate extractor produces the following results:

- it marks the overlaps between each manual sequence and each automatic sequence.
- as a summary, it computes the share of automatically extracted sequences which map manually extracted sequences, and vice versa. From these figures, one can estimate an approximation to recall and precision.

The results of this automatic procedure are used by the grammar designers for optimizing their grammars.

The second method is based on human evaluation of the quality of the extracted sequences:

- new learning objects are used to extract definitory contexts from them
- humans are confronted with these sequences and a span of the preceding and following context. They rate the quality of the sequence and, if they approve of it as being a definition, they mark those parts of the context which they would like to include in a glossary

This second evaluation helps to assess the quality of the extracted sequences and to measure the relation between the extracted sequences and the surrounding context. The automatic evaluation is currently being performed by the language resource providers. The evaluation tool is part of this deliverable.

References

- [1] Ismail Fahmi and Gosse Bouma. 2006. *Learning to Identify Definitions using Syntactic Features*. In: Proceedings of the EACL 2006 workshop on Learning Structured Information in Natural Language Applications.
- [2] Judith Klavans and Smaranda Muresan. 2001. *Evaluation of the DEFINDER System for Fully Automatic Glossary Construction*. In: *Proc. of AMIA Symposium 2001*.
- [3] Richard Tobin. 2005. *Lxtransduce, a replacement for fsgmatch*. URL: <http://www.cogsci.ed.ac.uk/~richard/lxml2/lxtransduce-manual.html>

Appendices

Use Cases

Author generates glossary for a learning object

Brief Description of Context and Goal: An author wants to provide a glossary for an existing learning object. ILIAS+LTTools or webservices will assist him to find candidates for terms and definitions of the glossary.

Related LT-Functionality: Glossary term and definition detection

Primary Actor: Author

Preconditions: 1) Author is within ILIAS repository and has permission to create glossaries; 2) author has access to webservice

Postconditions: A new glossary has been created, the author has chosen terms and definitions from a list of candidate terms and definitions provided by ILIAS+LTTools or webservice.

Main Success Scenario

1. Author sends an existing learning object to the glossary candidate detection service.

2. ILIAS+LTTools display learning object in Edit mode including a feature Generate Glossary
3. Author hits *Generate Glossary*
4. ILIAS+LTTools or webservice display a list of candidate terms and definitions with larger surrounding context
5. Author selects a set of terms and definitions, makes some manual changes, adds own terms and definitions, enters a glossary name and saves results

User searches definitions in learning objects

Brief Description of Context and Goal: A user wants to find a definition of a term he encounters in a learning object. ILIAS+LTTools or webservices will assist him to find definitory contexts for this term.

Related LT-Functionality: Definition detection for a term.

Primary Actor: User

Preconditions: user has access to webservice

Main Success Scenario

1. Author sends a set of learning object and a search term to the glossary candidate detection service.
2. ILIAS+LTTools or webservice display a list of definitions with larger surrounding context for the search term.
3. User finds a suitable definition.

User Manual

The glossary candidate detector:

- manages the input files
- calls Ixtransduce with the local grammar which should be applied to the input file
- extracts form the Ixtransduce output files the defining texts and the defined terms
- writes these extracted definitions to a file
- in a later version, the definitions will be passed on to the user via a user interface

The programme take needs the following parameters (in that order!)

- the file in which the definitions should be marked. Note that a) this file must conform to the LT4ELAna specification b) the DTD which is referenced in the file, in the DOCTYPE element, must be there;
- the grammar file. Note that a) if in this file you refer to the Ixtransduce DTD, this DTD must be in place;
- a language identifier. This argument is currently not used, but will be useful in a later version of the programme;
- the path to the Ixtransduce programme
- the output file; make sure that this file does not already exist. If it exists, it is overwritten (which might be sth you want the programme to do, in particular in the testing phase).

The programme runs currently only on Linux which is due to the fact that the Ixtransduce programme is only available for Linux. Therefore, you must have a newer linux version and a Java 1.5 Runtime Environment on your computer.

To run the programme, you have to do the following:

In the directory where the jar file (dfex01.jar) resides, run

```
java -jar dfex01.jar <path_and_name_of_your_linguistically_annotated_file> \  
<path_and_name_of_your_local_grammar> \  
&Language_Identifier <path_to_lxtransduce> \  
&path_and_name_of_your_output_file>
```

An example call:

```
java -jar dfex01.jar calimera3-01.ana.xml \  
german.gr de /home/lothar/bin Results
```

Now you can check the *Results* file and assess the results. In the webservice version the records in this file (at least parts of them) will be presented to the user through a web form.

Related deliverables

- Guidelines for the annotation of definitory contexts (on the portal, additional documents section)
- Grammars describing patterns for definitory contexts for Bulgarian, Czech, Dutch, English, German, Polish, Portuguese and Romanian (on the portal, grammars section)
- The software for the detection and extraction of glossary candidates (on the portal, tools section)
- Tools for the automatic evaluation of the glossary candidate detector (on the portal, tools section)