INTERNATIONAL WORKSHOP

Natural Language Processing and Knowledge Representation for eLearning Environments

held in conjunction with the International Conference
RANLP - 2007, September 27-29, 2007, Borovets, Bulgaria

PROCEEDINGS

Edited by

Paola Monachesi, Lothar Lemnitzer and Cristina Vertan

Borovets, Bulgaria

26th September 2007
Program Committee

Antonio Branco (University of Lisbon, Portugal)
Dan Cristea (University of Iași, Romania)
Diane Evans (Open University, United Kingdom)
Walther v. Hahn (University of Hamburg, Germany)
Erhard Hinrichs (University of Tübingen, Germany)
Susanne Jekat (Zürich Winterthur Hochschule, Switzerland)
Alex Killing (ETHZ, Switzerland)
Steven Krauwer (University of Utrecht, the Netherlands)
Vladislav Kubon (Charles University Prague, Czech Republic)
Petya Osenova (Bulgarian Academy of Sciences, Bulgaria)
Adam Prezpiórkowski (Institute of Computer Science, Polish Academy of Sciences)
Anne de Roeck (Open University, United Kingdom)
Mike Rösner (University of Malta, Malta)
Paul Buitelaar (DFKI, Germany)
Lothar Lemnitzer (University of Tübingen, Germany)
Paola Monachesi (University of Utrecht, the Netherlands)
Marco Ronchetti (University of Trento, Italy)
Cristina Vertan (University of Hamburg, Germany)

Organizing Committee

Paola Monachesi, University of Utrecht, The Netherlands
Lothar Lemnitzer, University of Tübingen, Germany
Cristina Vertan, University of Hamburg, Germany

The workshop is partially supported by the European Community under the Information Society and Media Directorate, Learning and Cultural Heritage Unit via the LT4eL project, STREP-IST 027391.
Foreword

Several initiatives have been launched in the area of Computational Linguistics, Language Resources and Knowledge Representation both at the national and international level aiming at the development of resources and tools. Unfortunately, there are few initiatives that integrate these results within eLearning. The situation is slightly better with respect to the results achieved within Knowledge Representation since ontologies are being developed which describe not only the content of the learning material but crucially also its context and the structure. Furthermore, knowledge representation techniques and natural language processing play an important role in improving the adaptivity of learning environments even though they are not fully exploited, yet. eLearning environments constitute valuable scenarios to demonstrate the maturity of computational linguistic methods as well as of natural language technologies and tools. This kind of task-based evaluation of resources, methods and tools is a crucial issue for the further development of language and information technology.

The goal of this workshop is to discuss:

- the use of language and knowledge resources and tools in eLearning
- requirements on natural language resources, standards, and applications originating in eLearning activities and environments
- the expected added value of natural language resources and technology to learning environments and the learning process
- strategies and methods for the task based evaluation of Natural Language Processing applications.

The eight accepted papers describe different functionalities developed in the framework of the EU-Project Language Technology for eLearning, (www.lt4el.eu) from the keyword and definition extractor that can be adapted to several languages up to the language independent ontological search engine.

The first paper ALPE as LT4eL processing chain environment by Dan Cristea and Ionut Pistol describes an Automated Linguistic Processing environment used to build linguistic processing chains involving the annotation formats and tools developed in the LT4eL project.

In the paper Combining pattern-based and machine learning methods to detect definitions for eLearning purposes Eline Westerhout and Paola Monachesi discuss how the performance of a pattern-based glossary candidate detector capable of extracting definitions in eight languages can be improved by machine learning techniques. The discussion is based on results for Dutch.

Language specific definition extractors and their performance are presented in the following three papers: Automatic Extraction of Definitions in Portuguese: A Rule-Based Approach by Rosa del Gaudio and Antonio Branco, Grammar-based Automatic Extraction of Definitions and Applications for Romanian by Adrian Iftene, Diana Trandabat, and Ionut Pistol and On the evaluation of Polish definition extraction grammars by Adam Przepiórkowski, Łukasz Degórski, Beata Wójtowicz. The last one addresses also the issue of the appropriate methodology for the evaluation of the developed functionality.

One of the functionalities developed within the LT4eL project is the possibility to annotate learning objects semi-automatically with keywords that describe them. To this end, a keyword extractor has been created which can deal with documents in 8 languages. The keyword extractor is presented in the paper Keyword extraction for metadata annotation of Learning Objects by Lothar Lemnitzer and Paola Monachesi.

The paper Applying Ontology-Based Lexicons to the Semantic Annotation of Learning Objects by Kiril Simov and Petya Osenova discusses the role of the ontology in the definition of domain lexicons in several languages and its usage for the semantic annotation of Learning Objects.

Finally in the paper Crosslingual Ontology-Based Document Retrieval, Eelco Mossel describes the crosslingual search engine.

The programme will be complemented by two invited talks by Iryna Gurevych on Educational Natural Language Processing and by Roberto Basili on Cross Lingual Retrieval.

We hope that the workshop will provide a forum for interaction among members of different research communities, and a means for attendees to increase their knowledge and understanding of the potential of Language Technology resources in eLearning.

The workshop is partially supported by the European Community under the Information Society and Media Directorate, Learning and Cultural Heritage Unit via the LT4eL project, STREP-IST 027391.

Lothar Lemnitzer, Paola Monachesi, Cristina Vertan.
<table>
<thead>
<tr>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>INVITED TALKS</td>
<td>1</td>
</tr>
<tr>
<td>Dan CRISTEA, Ionuț PISTOL and Corina FORĂSCU</td>
<td>3</td>
</tr>
<tr>
<td><em>ALPE as LT4eL Processing Chain Environment</em></td>
<td></td>
</tr>
<tr>
<td>Rosa Del GAUDIO and António BRANCO</td>
<td>11</td>
</tr>
<tr>
<td><em>Supporting e-Learning with Automatic Glossary Extraction: Experiments with Portuguese</em></td>
<td></td>
</tr>
<tr>
<td>Adrian IFTENE, Diana TRANDABĂȚ and Ionuț PISTOL</td>
<td>19</td>
</tr>
<tr>
<td><em>Grammar-Based Automatic Extraction of Definitions. Applications for Romanian</em></td>
<td></td>
</tr>
<tr>
<td>Lothar LEMNITZER and Paola MONACHESI</td>
<td>27</td>
</tr>
<tr>
<td><em>Keyword Extraction for Metadata Annotation of Learning Objects</em></td>
<td></td>
</tr>
<tr>
<td>Eelco MOSSEL</td>
<td>35</td>
</tr>
<tr>
<td><em>Crosslingual Ontology-Based Document Retrieval</em></td>
<td></td>
</tr>
<tr>
<td>Adam PRZEPIÓRKOWSKI, Łukasz DEGÓRSKI and Beata WÓJTOWICZ</td>
<td>43</td>
</tr>
<tr>
<td><em>On the Evaluation of Polish Definition Extraction Grammars</em></td>
<td></td>
</tr>
<tr>
<td>Kiril SIMOV and Petya OSENOVA</td>
<td>49</td>
</tr>
<tr>
<td><em>Applying Ontology-Based Lexicons to the Semantic Annotation of Learning Objects</em></td>
<td></td>
</tr>
<tr>
<td>Eline WESTERHOUT and Paola MONACHESI</td>
<td>57</td>
</tr>
<tr>
<td><em>Combining Pattern-Based and Machine Learning Methods to Detect Definitions for eLearning Purposes</em></td>
<td></td>
</tr>
</tbody>
</table>
INVITED TALKS

Educational Natural Language Processing, Iryna Gurevych (Technische Universität Darmstadt)

The talk aims at defining Educational Natural Language Processing (e-NLP) as a field of research exploring the use of NLP techniques in educational contexts. Current renaissance of interest in e-NLP is due to eLearning 2.0 which leads to the creation of large repositories with user generated discourse and user generated metadata. This user generated knowledge can be employed for creating structured knowledge bases to improve NLP, but it needs advanced information management capabilities and NLP to be efficiently accessed. The talk will present snapshots from several ongoing e-NLP research projects in the Ubiquitous Knowledge Processing (UKP) Lab at the Technische Universität Darmstadt to illustrate some of the claims and challenges in e-NLP.

From multimedia semantic indexing to cross-lingual retrieval: the Prestospace approach to cultural heritage preservation and dissemination., Roberto Basili, university of Roma, Tor Vergata

Digital archives in large European TV broadcasters constitute an immense resource for cultural, historical and scientific education. In the Prestospace project, a framework for the acquisition and delivery of semantic metadata (MAD) from the multimedia repositories of the major European TV broadcasters (e.g. BBC, RAI) has been defined and a platform for annotation, indexing, publication and conceptual retrieval has been realised. An innovative feature of the MAD system is the semantic annotation ability able to derive ontology-based metadata through speech recognition and natural language processing. The specific industrial requirements of the project pushed for the adoption of robust, efficient and portable models for language processing. In the talk, I will discuss some research aspects of the semantic analysis process applied in Prestospace. A particular emphasis will be given to the unsupervised learning approach adopted for cross-lingual retrieval from the annotated multimedia material. It integrates geometrical learning models (like Latent Semantic Analysis) with large scale ontological and lexical resources (e.g. Wordnet): the resulting semantic disambiguation process supports robust unsupervised query translation. The overall MAD approach supports effective multimedia annotation and indexing and it is easily applicable to large scale archives and Web scenarios (e.g. Web 2.0).
Abstract
This paper briefly describes the concept, initial implementation and usage of the ALPE\(^1\) system for natural language processing. A hierarchy connecting annotation schemas, processing tools and resources is used as working environment for the system, which can perform various complex NL processing tasks. ALPE will be used to build linguistic processing chains involving the annotation formats and tools developed in the LT4eL\(^2\) project. The particularities and advantages of such an endeavor are the main topics of this paper.

Keywords
XML annotation, processing architectures, e-learning, linguistic processing systems, multilinguality

1. Introduction
One of the latest developments in Computational Linguistics, and one which promises to have a significant impact for future linguistic processing systems, is the emerging of linguistic annotation meta-systems, that make use of existing processing tools and implement some sort of processing path, pipelined or otherwise.

The “wild” diversity of formats, tools and resources scares off a newcomer or less informed user who needs to configure an NLP (Natural Language Processing) architecture to solve a given task. Configuration and parametrisation of processing chains is also very time consuming due to the heavy documentation the component modules come with. The more sophisticated the task, the more likely it is that it requires complex pre-processing steps involving several other NLP systems, which have to be chosen, documented and interfaced.

The newly emerged linguistic processing metasystems make use of existing modules in building LP chains, use existing linguistic resources and allow the user to add/build new ones, and also allow the user to compare and choose among the available modules. The two most prominent systems of this type are GATE\(^3\) and IBM's UIMA\(^4\).

GATE [3,4] is a versatile environment for building and deploying NLP software and resources, allowing for the integration of a large amount of built-ins in new processing pipelines that receive as input single documents or corpora. The user can configure new architectures by selecting from a repository pool the desired modules, as parts of a chain. The configured chain of processes may be put to work on an input file and the result is an output file, XML annotated.

UIMA [7] is a new promising release of IBM Research (first freely available version – June 2007). It offers the same general functionalities as GATE, but once a processing module is integrated in UIMA it can be used in any further chains without any modifications (GATE requires wrappers to be written to allow two new modules to be connected in a chain). Also, UIMA allows the user to work with various annotation formats and perform various additional operations on annotated corpora. Since the release of UIMA, the GATE developers have made available a module that allows GATE and UIMA processing modules to be interchangeable, basically merging the “pool” of modules available.

ALPE is another approach to the task of developing an LP meta-system, offering more flexibility than existing systems. ALPE is based on the hierarchy of annotation schemas described in [1]. In this model, XML annotation schemas are nodes in a directed acyclic graph, and the hierarchical links are subsumption relations between schemas. In [2] it is described the way the graph may be augmented with processing power by marking edges linking parent nodes to daughter nodes with processors names, each realizing an elementary NL processing step. On the augmented graph, three operations are defined: simplification, pipeline and merge. A navigation algorithm is described in this hierarchy, which computes paths between a start node, corresponding to an input file, and a

---

1 Automated Linguistic Processing Environment
2 Language Technology for e-Learning, IST 027391, http://www.lt4el.eu/
3 http://gate.ac.uk/
4 http://www.research.ibm.com/UIMA/
As such, a hierarchical relation between a node \( x \) and a tag-name of the hierarchy (we will say also that \( x \) is a descendent of \( A \)). We will agree to use the term descendent of \( A \) with a node \( B \) in the hierarchy (we will say also that \( B \) is a descendent of \( A \)) then the following conditions hold simultaneously:

- any tag-name of \( A \) is also in \( B \);
- any attribute in the list of attributes of a tag-name in \( A \) is also in the list of attributes of the same tag-name of \( B \).

As such, a hierarchical relation between a node \( A \) and one descendant \( D \) describes \( A \) as an annotation schema which is more informative than \( A \). In general, either \( B \) has at least one tag-name which is not in \( A \), and/or there is at least one tag-name in \( B \) such that at least one attribute in its list of attributes is not in the list of attributes of the homonymous tag-name in \( A \). We will agree to use the term path in this DAG with its meaning from the support graph, i.e. a path between the nodes \( A \) and \( B \) in the graph is the sequence of adjacent edges, irrespective of their orientation, which links nodes \( A \) and \( B \). As we will see later, the way this graph is being built triggers its property of being fully connected. This means that, if edges are seen undirected, there is always at least one path linking any two nodes.

2.2 The hierarchy augmented with processing power

In NLP, the needs for reusability of modules, and language and application independence impose the reuse of specific modules in configurable architectures. In order for the modules to be interconnectable, the module’s inputs and outputs must observe the constraints expressed as annotation schemas.

When we place processes on the edges of the graph of linguistic metadata, the hierarchy of annotation schemas becomes a graph of interconnecting modules. More precisely, if a node \( A \) is placed above a node \( B \) in the hierarchy, there should be a process which takes as input a file observing the restrictions imposed by the schema \( A \) and produces as output a file observing the restrictions imposed by the schema \( B \).

We will call a graph (or hierarchy) of annotation schemas on which processing modules have been marked on edges as being augmented with processing power (or simply, augmented) [2]. The null process, marked \( O \), is a module that leaves an input file unmodified.

2.3 Building the hierarchy

Three hierarchy building operations are used in our model: initialize-graph, classify-file and integrate-process. They are described below.

The initialize-hierarchy operation receives no input and outputs a trivial hierarchy formed by a \( \text{ROOT} \) node (representing the empty annotation schema).

Once the graph is initialised, its nodes and edges are contributed by classifying documents in the hierarchy.

The classify-file operation takes an existing hierarchy and a document marked with metadata observing a certain schema and classifies the schema of the document within the hierarchy. The operation results in an updated hierarchy and the location of the input schema as a node of the hierarchy. If the input document fully complies with a schema described by a node of the hierarchy, the latter remains unchanged and the output indicates this existing node; otherwise a new node, corresponding to the annotation schema of the input document, is inserted in the proper place within the hierarchy.

Integrate-process is an operation aiming to properly attach processes to the edges of a hierarchy of annotation schemas, mainly by labeling edges with processors, but sometimes also by adding nodes and edges and labeling the connecting edges.

An ALPE type hierarchy can either be defined completely manually, or partially manually and partially completed with operations described above, or only using the above operations. The ALPE hierarchy variant developed for the LT4eL project is fully constructed manually, as its purpose is testing the functionalities offered, and less the building procedure.
2.4 Operations on the augmented graph

Three main operations can be supported by the model, as follows.

If an edge linking a node A to a node B (therefore B being a descendant of A) is marked with a process p, we say that A pipelines to B by p. Equally, when a file corresponding to the schema A is pipelined to B by p, it will be transformed by the process p onto a file that corresponds to the restrictions imposed by the schema B. This arises in augmenting the annotation of the input file (observing the restrictions of the schema A) with new information, as described by schema B.

For any two nodes A and B of the graph, such that B is a descendant of A, we will say that B can be simplified to A. When a file corresponding to the schema B is simplified to A, it will lose all annotations except those imposed by the schema A. Practically, a simplification is the opposite of a (series of) pipeline(s) operation(s).

The merge operation can be defined in nodes pointed by more than one edge on the hierarchical graph. It is not unusual that the edges pointing to the same node are labelled by empty processors. The merge operation applied to files corresponding to parent nodes combines the different annotations contributed by these nodes onto one single file corresponding to the schema of the emerging node.

With these operations, the graph augmented with processing power is useful in two ways: for goal-driven, dynamic, configuration of processing architectures and for transforming metadata attached to documents. Automatic configuration of a processing architecture is a result of a navigation process within the augmented graph between a start node and a destination node, the resulted processes being combinations of branching pipelines (serial simplifications, processing and merges). The difference with respect to GATE and UIMA, both allowing only pipeline processing in which the whole output of the preceding processor is given as input to the next processor, is that in our model the required processing may result in a combination of branching pipelines. This is due to the introduction of the merge operation which is able to combine two different annotations on the same file. Once the process is computed, then it can be applied on an input file displaying a certain metadata in order to produce an output file with the metadata changed as intended. These two files comply with the restrictions encoded by the start node and, respectively, the destination node of the hierarchy.

Since the graph is fully connected, there should always be at least one path connecting any two nodes. The paths found are made up of oriented edges and, depending on whether the orientation of the edges is the same as that of the path or not, we will have pipeline operations or simplification operations. The flow of paths between the start and destination node configures the processing combination that transforms any file observing the specifications of the start node (schema) onto a file observing the specifications of the destination node (schema).

Once the entry and exit points in the hierarchy have been determined and translation links have been devised, all the rest is done by the hierarchy itself augmented with the processing power in the manner described above. This way, the processing needed to arrive from the input to the output is computed by the hierarchy as sequences of serial and parallel processing steps, each of them supported in the hierarchy by means of specialised modules. Then the process itself is launched on the input file. It includes an initial translation phase, followed by a sequence of simplifications, pipelines and/or merges, as described by the computed path, and followed by a final translation, which is expected to produce the output file.

2.5 Features

In this section we will describe a set of features, important for environments working with linguistic resources and tools, that emerge from the proposed model. These features are important especially when considering the proposed integration of resources and tools belonging to the LT4eL project in an ALPE type hierarchy.

Multilinguality

Usually the adaptation of a module to process a certain natural language is given by the specific set of resources it accesses. For instance, a POS-tagger runs the same algorithms on different sets of language models in order to tag documents for part-of-speeches in different languages. To take another example, a shallow parser applies a set of regular expressions, which are language dependent, in order to identify chunks. In both cases the processing modules are language independent and only the specific language model or the specific set of rules make them applicable to the language L1 or L2.

To realise multilinguality within the proposed model means to map the edges of the augmented graph on a collection of repositories of configuring resources (language models, sets of grammar rules, etc.) which are specific to different languages. This can be achieved if the edges of the graph labeled with processes are indexed with indices corresponding to languages. This way, to each particular language an instance of the graph can be generated, in which all edges keep one and the same index – the one corresponding to that particular language. This means that all processors of that particular language should access the configuring resources, specific to that language, in order for the hierarchy to work properly. For instance, in the graph instance of language $L_x$, the edge corresponding to a POS-tagger has as index $L_x$, meaning that it accesses a configuring resource file that is specific to language $L_x$ – the $L_x$ language model (Figure 1).

It is a fact that different languages have different sets of processing tools developed, English being perhaps the richer, presently. Ideally, the lack of a tool in a specific
language should be put on to the lack of the corresponding configuring resource, once a language independent processing module is available for that task. It is also the case that differences exist in processing chains among languages. For instance one language could have a combined POS-tagger and lemmatizer while another one

realizes these operations independently, pipelining a POS-tagger with a lemmatization module. These differences are reflected in particular instances of sections of the graph, which, although reproduce the same set of nodes, do not allow but for certain edges linking them. The missing edges inhibit pipelining operations along them, but are suited for simplification operations.

**Distributivity and access**

Edges, as recorders of processors, can be seen as Web services, therefore can be physically supported by servers anywhere on the virtual space. Similarly, documents (the files attached to nodes) could be physically located in different locations than the hierarchy itself. This way, the whole augmented graph of annotation schemas could be distributed over the Web. However, as the unique accessing gate, a portal holding a representation of the entire graph, on which classification and navigation operations can be performed, must exist. By manual configuration and/or repeated classification accesses, the graph grows. Also navigation accesses are initiated by users and run on the portal. They leave the graph constant while returning the computed architectures, to be executed mainly remotely from the portal, by activating chains of processors which are not all located on the same machine, but which are pointed to by edges of the graph.

**Versioning of language resources**

Each document can have multiple annotations, in correspondence with the nodes of the hierarchy. While some of the language resources may have been created by human annotators (therefore being taken as gold standards), others can be automatically created, some even using the augmented hierarchy. More than that, different versions of the same hub document may correspond to just one node in the hierarchy, mainly by being created both ways (manually and automatically). The versioning problem could be accommodated by the model through an indexing mechanism similar to the language indexing of edges, by allowing the attachment of different versions of the same document to nodes of the hierarchy. When a computed architecture is run over an input file (corresponding to a start node), the output file (corresponding to the destination node) will be indexed identically with the input file.

**Manual versus automatic annotation**

While automatic annotation is supported by the graph, how can manual annotation be accommodated by the approach?

Usually, in order to train processing modules in NLP, developers use manually annotated corpora. To create such corpora, they make use of annotation tools configured to help placing XML elements over a text, and to decorate them with attributes and values. As such, if annotation tools do, although in a different way, the same jobs which can be performed by processing modules, it is most convenient to associate them with edges in the graph in the same way in which processing modules are associated with these edges.

Meanwhile, it is clear that manual annotation cannot be chained in complex processing architectures in the same way in which automatic annotation can. In order to differentiate between automatic and manual processes, as encumbered by pairs of schemas observing the descendent relation, it results that edges should have facets, for instance AUT and MAN. Under the AUT facet of a POS-tagging edge, for instance, the automatic POS-tagger should be placed, while under the MAN facet – the POS-tagging annotation tool should be placed.

The configuration files of these tools can usually be separated from the tools themselves. We can say that the corresponding configuration files particularize the annotation tools, which label edges of the graph, in the same way in which language specific resources particularize processing modules.

**IPR and cost issues**

Intellectual property rights can be attached to documents and modules as access rights. Only a user whose profile corresponds to the IPR profile of a resource/tool can have access to it. As a result, while computation of processing chains within the hierarchy is open to anybody, the actual access to the dynamically computed architectures could be banned to users not corresponding to certain IPR profiles of certain component modules or resources they need.

More than that, some price policies can be easily implemented within the model. For instance, one can imagine that the computation of a path results also in a computation of a cost, depending on particular fees the chained Web servers charge for their services, on the load of some service providers, etc.

Out of this, it is also imaginable the graph as including more than one edge between the same two nodes in the hierarchy. This can happen when different modules performing the same task are reported by different contributors. When these modules charge fees for their services, it is foreseeable also an optimization calculus over
the set of paths that can be computed for a transformation with respect to the overall price.

**Facing the diversity of annotation styles**

It is a fact that nowadays a huge diversity of annotation variants circulates and is being used in diverse research communities. It is far from us to believe that a Procustes’ Bed policy could ever be imposed in the CL or NLP community, that would aim for a strict adoption of standards for the annotated resources. On the other hand, it is also true that efforts towards standardization are continually being made (see the TEI, XCES, ISLE, etc. initiatives [8]). Moreover, Semantic Web, with its tremendous need for interconnection and integration of resources and applications on communicating environments, boosts vividly the appeal for standardization. It is therefore foreseeable that more and more designers will adopt recognized standards, in order to allow easy interoperability of their applications. A realistic view on the matter would bring into the focus the standards while also providing means for users to interact with the system even if they do not rigorously comply with the standards.

We have seen already that, by classification, any schema could be placed in the hierarchy. Of course, classification could increase in an uncontrollable way the number of nodes of the hierarchy. The proliferation could be caused not so much by the semantic diversity of the annotations, as by the differences in name spaces (names of tags and attributes). Suppose one wants to connect a new file to the hierarchy in order to exploit its processing power. What s/he has to do is to first classify the metadata scheme of the file. If the system reports the result as being a new node in the hierarchy, then its position gives also indications of its similarity/dissimilarity with the neighboring schemas. A visual inspection of the names used can reveal, for instance, that a simple translation operation can make the new node identical to an existing one. This means that the new schema is not new for the hierarchy, although the set of conventions used, which make it different from those of the hierarchy, are imposed by the restrictions of the user’s application.

Technically, this can be achieved by temporarily creating links between the new schema classified by the hierarchy, as a new node, and its corresponding schema in the hierarchy. Processing along such a link is different than the usual behavior associated to the edges of the graph and is specific to wrappers. It describes a translation process, in which the annotation is not enriched, but rather names of tags and attributes are changed. Ideally, the processing abilities of the hierarchy should include also the capability to automatically discover the wrapping procedure. This task is not trivial since it would require that the hierarchy “understands” the intentions hidden behind the annotation, displaying an intelligent behavior which is not easy to implement, but could make an interesting topic for further research.

2.6 **ALPE vs GATE and UIMA**

Since ALPE, GATE and UIMA are systems capable of performing similar tasks, the significant differences and, most important, the advantages of ALPE over the other two are presented below.

First of all, ALPE is intended primarily to facilitate user interaction with the system, allowing the common user to access integrated resources and tools. As a standalone linguistic processing environment, the user is presented with a visual representation of a hierarchy of annotation formats and has basically three main choices: he can either add a new resource to the hierarchy, a new processing tool or create and use a processing chain by specifying start and end nodes in the hierarchy and providing the input document. In comparison, GATE offers a user interface just for creating and using processing chains, and these have to be built manually, requiring at least a well informed user. UIMA is even more oriented to the CL specialist, offering very little in terms of visual user interaction.

Every one of the three main functionalities is easier to perform using ALPE. Both UIMA and GATE require some formal description to be written for each new resource integrated into the system, but ALPE generates these formal descriptions automatically. When adding a new processing tool, ALPE has much more permissive restrictions with regards to what tool can be integrated: it basically has to be either a webservice or a command line executable under Windows or Linux. GATE allows the user to integrate just Java and Perl based tools, and this is done by writing some dedicated code. UIMA allows only Java based tools to be integrated, and only after significant implementations and changes to the original code.

When creating and using processing chains, the most significant advantage of ALPE is the automatically creation of processing chains, and the fact they can be created between any two formats in the existing hierarchy (if the required modules are available). GATE and UIMA offer relatively simple ways to create and use processing chains, but the user has to be sure the required modules exist and have compatible input/output formats. Also, ALPE deals much easier with multilinguality, as it has a module that performs language identification automatically for each input file, then selects the corresponding tools and language resources, if available. GATE and UIMA are mainly focused on English, GATE having some modules for Romanian, but the user has to make sure to select those and not the English ones when building a processing chain for a Romanian document.

Let us consider a simple use-case: the user has two processing tools he wants to use on the same input file and merge the results in an output file. Using ALPE he just has to use the available functionality to integrate the two tools in any hierarchy (even if all the annotation formats involved are not currently available: new nodes will be created automatically), then input the file and specify the required output format (node). Using GATE, the user has to
implement the integration of the tools to make them available to the processing chain building interface, to build and run two processing chains, one for each tool, then merge the results manually (GATE does not allow parallel processing and merging of annotations). UIMA performs this task basically the same as GATE, requiring even more implementation when integrating the new tools, but can perform annotations merging.

3. ALPE and LT4eL

The model presented in the previous sections is partially implemented and will be used (in an intermediate version) in the framework of the LT4eL European FP6 project [5]. As the main objective of the project is to provide functionalities based on language technologies and to integrate semantic knowledge in Learning Management Systems (LMS), the first step was to create an environment for collecting and (semi) automatic exploitation of language resources and tools. For the 9 languages involved (Bulgarian, Czech, Dutch, English, German, Maltese, Polish, Portuguese and Romanian), a multilingual corpus, partially parallel, of almost 9 million words was collected, annotated and uploaded on the project’s portal. There are about 30 linguistic tools on the portal, corresponding to various processing steps, hence to edges of the hierarchy of annotation schemas. The following sub-sections will briefly describe the LT4eL formats and tools which will be involved in the adaptation of ALPE for LT4eL.

3.1 The resources in the hierarchy of schemas

The linguistic resources - called learning objects (LO) - were first collected as documents corresponding to formats on the first layer of the ALPE LT4eL hierarchy (see Fig. 2), according to their language, format (doc, pdf, plain text, html or other), domain (broadly: the use of computers in education), and IPRs. In figure 2 an ALPE hierarchy, as described in this paper, includes the sxml node and all others below it. After their automatic conversion to XML, using a specially created converter [6], the objects were linguistically annotated to mark (tokens, part-of-speech, lemma), hence placed on the second layer of the hierarchy – Figure 2. After another conversion to the specific format used as input for the keyword extractor developed within the project, the resources are taken to the third layer, corresponding to the annotation of keywords and definitory contexts. This specific hierarchy was used in LT4eL as a repository form for the projects LOs. Practically by clicking on the node the users have access to all files observing the specific annotation. A dedicated portal was built that includes functionalities for upload and download the projects’ LOs.

The formats (ALPE schema definitions) shown in figure 2 are:
- doc: Format of MS Word and OpenOffice text documents;
- pdf: Portable Document Format generated by any of the available software;
- latex: LaTeX document format;
- html: HTML format for web pages and documentations;
- txt: simple text format, without markups and viewable/editable with basic editors;
- other: formats other than the ones nominated above;
- sxml: XML format with basic formatting information extracted from the txt and-or html formats;
- morpho: annotated format with added morphological information;
- tok: annotated format with added tokenisation;
- pos: annotated format with added part of speech information;
- lemma: annotated format with marked lemmas for words;
- NP: annotated format with marked Noun Phrases;
- wp2xml: XML format with morphological and syntactical information merged from the above formats;
- awk: XML with automatically generated keywords annotation;
- adef: XML with automatically generated definitions annotation;
- axml: XML format combining the automatically generated keywords and definitions annotations.

3.2 The tools of LT4eL

The LT4eL corpora required extensive processing. The tools used for working with the LT4eL corpora are either existing tools, which were adapted for LT4eL purposes, or tools developed as part of the project.

---

5 http://consilr.info.uaic.ro/uploads_lt4el/
6 http://ufal.mff.cuni.cz/~spousta/lt4el/html2xml/LT4ELBase.dtd
Initially, all collected documents were in various formats and they were converted to an XML format preserving some of the visual formatting (for further processing). This step involved the development of a conversion tool from an intermediate HTML or TXT version of the original document (obtained using existing tools) to an XML format observing the LT4eL format. Basically, this tool, combined with the original (automated) conversion from the original format to the intermediate format, allows the user to input almost any kind of file in the processing hierarchy and obtain a required XML format. The conversion tool is configurable for various output formats, its source code is available, so this will become one of the core tools in the ALPE system.

For the next step, adding basic linguistic annotation to the XML corpora, existing tools were employed. Each partner language identified, adapted and used its own tools, and produced various types of annotated XML. All these formats were transformed according to a common DTD, to include linguistic information such as token markings, lemma information, POS tags, morpho-syntactical characteristics and noun phrase identifiers. Two tools were implemented to mark the keywords and definitions in the corpora using this common XML standard for input files. The keywords and definitions were considered with respect to the LT4eL domain: teaching computer science and e-learning. All processing modules are under a continuous process of improvement. One of the final goals of the project is to fully develop these technologies with modularity and language-independence as two of the main characteristics, hence making LT4eL an ideal environment for practical testing a system such as ALPE.

### 3.3 ALPE-LT4eL

The LT4eL hierarchy represents the first significant deployment of an ALPE environment. The nature and requirements of the project impose the following characteristics of the system:

- Has to be able to handle resources/tools for 8 languages (the initial version includes tools for 3 languages, the others having to be added later).
- Has to work with files in 12 different formats. ALPE automatically identifies the file format and language.
- Has to be able to handle a wide variety of processing modules. Preliminary inquiries showed that most
modules are either Java or Perl based. Some modules are available only as executables.

- Has to be able to handle diverse processing configurations in different languages, from strictly serial to a combination between serial and parallel. In average, a processing chain involves 4 processing modules (and several of the ALPE core modules, like language identification and annotations merging).

The ALPE-LT4eL hierarchy can be constructed manually, first by defining the schema definitions graph, then adding LT4eL modules to the edges. The fact that the formats and modules involved are already available and are not subject to significant changes allowed the hierarchy to be fully build prior to its actual use. The ALPE-LT4eL hierarchy allows a user to input a document in any LT4eL format and automatically obtain any of the other formats in the hierarchy, except those above the \texttt{txt} and \texttt{html} level.

The integration of LT4eL tools and formats in an ALPE hierarchy makes processing and adding resources to the LT4eL corpus a much simpler and quicker task. All the transitions are performed automatically, as well as the detection of the input file, required resources and modules.

The manual configuration works faster only for simple processing chains. When the number of integrated modules increases, the advantage of the automated system becomes visible. Using the ALPE type of processing is more justified in complex projects involving large numbers of data formats and processing modules. Also, this type of processing can give access for non-specialist users easy to resources and tools, being able to test any LT4eL module using any file available in the many possible input formats.

Moreover, ALPE-LT4eL automatically identifies the language and checks whether a XML file (input for one of the modules) conforms to the required DTD assuring the correct execution of the processing flows.

The conceptual design of the ALPE hierarchy makes possible, in a later stage, to include new nodes (formats) and modules, as well as modules and resources for processing files in other languages.

4. Conclusions
In this paper we have argued for augmenting the theoretical model of an automatic configuration of NLP architectures, introduced in [1] and [2], with new features that can accommodate multilinguality, distributivity, versioning of language resources, manual versus automatic annotation, IPR and cost issues, as well as the diversity of annotation styles. For the first time, the ALPE environment has found an application field in an European project dedicated to applying linguistic processing to e-learning. Although only a part of the functionality of the ALPE framework has been exploited in this context, since the hierarchy itself was considered given and therefore not dynamically built, the integration of ALPE in the LT4eL LMS can bring versatility for the user with respect to the file format of the input and easiness to expand the functionalities of the system for other languages.

Other envisaged deployments of ALPE hierarchies will be used in Question Answering, Textual Entailment and Anaphora Resolution systems. Eventually, ALPE will be deployed as a global NLP hierarchy dynamically built and usable on the net as a webservice.

5. Acknowledgments
Part of the work described in this paper was funded through the LT4eL project, STREP 027391 in FP6-2004-IST-4.

6. References
Supporting e-learning with automatic glossary extraction: Experiments with Portuguese

Rosa Del Gaudio  
University of Lisboa  
rosa@di.fc.ul.pt  

António Branco  
University of Lisboa  
antonio.branco@di.fc.ul.pt  

Abstract
This paper reports a preliminary work on automatic glossary extraction for e-learning purpose. Glossaries are an important resource for learners, in fact they not only facilitate access to learning documents but also represent an important learning resource by themselves. The work presented here was carried out within the project LT4eL which aim is to improve e-Learning experience by the means of natural language and semantic techniques. This work will focus on a system that automatically extract glossary from learning objects, in particular the system extract definitions from morpho-syntactic annotated documents using a rule-based grammar. In order to develop such a system a corpus composed by a collection of Learning Object covering three different domain was collected and annotated. A quantitative evaluation was carried out comparing the definition retrieved by the system against the definitions manually marked. On average, we obtain 14% for precision, 86% for recall and 0.33 for $F_2$ score.

Keywords
automatic definition extraction, glossary candidate detector, Portuguese, e-learning management systems

1 Introduction
The main focus of this work is supply Learning Management Systems (LMS) with a tool allowing an easy and quick glossary building. The definition extraction system presented here was developed with the practical objective of supporting the functioning of the module of Glossary Candidate Detector (GCD), in a Learning Management System (LMS).

The research underlying the extraction system presented here was carried out within the LT4eL project ¹ funded by European Union (FP6). The main goal of this project is to improve LMSs by using language technology, in order to make more effective the retrieval and the management of learning materials and information.

In particular, the ILIAS² Learning Management Systems is being extended with new functionalities to support the different actors in e-learning environments. ILIAS is a fully fledged web-based learning management system that allows users to create, edit and publish learning and teaching material in an integrated system with their normal web browsers. At present, it is being extended with a automatic keyword extractor [8] and a GCD.

This GCD module permits the extraction of candidates to definitory contexts from learning objects. Such a module is thus meant to help content providers and teachers to speed up the process of building glossaries corresponding to the learning objects they are making available via the LMS to their users and students.

In particular the final user, the content provider or the teacher, will take advantage, in terms of time saving, in generating the glossary using the tool; instead of scrolling the entire document looking for terms and definition the system will do it for him presenting the result in a page (see Figure 1), and then the user can accept or discard each definition as well as modify it.

Beside the advantage in term of time saving this approach allows the construction of specific glossary for each learning object, and this positively influence the learning process in two different way. First, these glossaries can be used as a quick index to the information contained in the original document, second, the learned will have access not to a general definition of a concept but to the specific acception that the concept takes in that particular context.

In this paper we will focus on the module of the system that allow the extraction of definition from document written in Portuguese. Modules for Bulgarian, Czech, Dutch, English, German, Polish and Romanian are being developed by other partners. We present the methodology used to develop the system and its performance by comparing the results of the system against a test data made of texts belonging to the domains of computer science, information society and e-learning.

In this work, a *definition* (also called definitory context) is assumed to be a sentence containing an expression (the *definiendum*) and its definition (the *definiens*) and a connector between them. We identify three different connector: the verb “to be”, all other verbs other than “to be” and punctuation mark such as “:.”. Here, we will be calling copula definition all those definitions where the verb “to be” acts as a connector, verb definition to all those definitions

---

¹ http://www.lt4el.eu/
² http://www.ilias.de/ios/index-e.html
that are introduced by a verb other than “to be”, and punctuation definitions to the ones introduced by punctuation marks.

In the next Section we present a brief review of researches focused on automatic glossary building and definition extraction, with some references to question answering systems.

In Section 3 we present the corpus collected in order to develop and test our system. The corpus is composed by learning objects in three different domains. Part of the corpus, referred to in the remainder of this paper as the development corpus, was retained to identify the lexical and syntactic patterns to be taken into account by the system. A grammar for each definition type was developed, which are described in Section 4.

In Section 5, the results of the evaluation of the grammar, in terms of recall, precision and F2-score, are presented and discussed.

In Section 6, we provide an analysis of errors and discuss possible alternative methods to evaluate our system.

Finally in Section 7 conclusions are presented as well as possible ways to improve the system in future work.

2 Previous Work

Glossary building is often considered as an extension of term extraction, many systems start with the identification of relevant term in a certain domain [7] and then try to apply different techniques in order to build glossary for that specific domain. For instance [1] present a methodology in order to construct a glossary, supporting knowledge dissemination, in a collaborative research project, where the semantic unification represents a critic factor for the success of the project. Different resources are used in order to construct the glossary: a term extractor based on statistical techniques, corpora and other glossaries where definition are extracted using a context free grammar. Then non relevant definitions are filtered out using information about the domain covered by the glossary. The use of information about the domain is also used in [7] in order to improve results. In our work we don’t have such previous domain knowledge, the system is supposed to work with documents belonging to different domains.

DEFINDER [6] is based on a methodology that use just lexical and syntactical information. This is an automatic definition extraction system targeting the medical domain, where the data is composed of consumer-oriented medical articles. In terms of quantitative evaluation, this system presents 87% precision and 75% recall. This very high values are probably due to the nature of the corpus.

Turning more specifically to the Portuguese language, there is only one publication in this area. Pinto and Oliveira [11] present a study on the extraction of definitions with a corpus from a medical domain. They first extract the relevant terms and then extract definition for each term. An evaluation is carried out for each term; for each term recall and precision are very variable ranging between 0% and 100%.

By using the same methodology for Dutch as the one used here, Westerhout and Monachesi [15] obtained 0.26 of precision and 0.72 of recall, for copula definitions, and 0.44 of precision and a 0.56 of recall, for other verbs definition.

Hearst [5] proposed a method to identify a set of lexico-syntactic patterns to extract hyponym relations from large corpora and extend WordNet with them. This method was extended in recent years to cover other types of relations[10].

In particular, Malaise and colleagues [9] developed a system for the extraction of definitory expressions containing hyperonym and synonym relations from French corpora. They used a training corpus with documents from the domain of anthropology and a test corpus from the domain of dietetics. The evaluation of the system using a corpus of a different domain, makes results more interesting as this put the system under more stressing performance. Nevertheless, it is not clear what is the nature and purpose of the documents making this corpora, namely if they are consumer-oriented, technical, scientific papers, etc. These authors used lexical-syntactic markers and patterns to detect at the same time definitions and relations. For the two different, hyponym and synonym, relations, they obtained, respectively, 4% and 36% of recall, and 61% and 66% of precision.

Answering questions asking for a definition is a particular dimension of the broad task of question answering that is very related to the main focus of this paper. The objective is that given an expression, a definition for it should be retrieved in a corpus or in the entire web. The main difference between this task and our work resides in the fact that we do not know beforehand the expressions that should receive definitions. This lack of information makes the task more difficult because it not possible to use the term as a clue for extracting its definitions.

Saggion [13] presents results of the TREC QA 2003 competition, where he tested his QA system against 50 definition questions. He just reports F5 score, where the recall is 5 times more important precision. His system obtained a F-score of 0.236, where the best score in the same competition was of 0.555 and the median was of 0.192.
The sentence O cabo é a parte mais básica duma rede. (The cable is the most basic component of a network) in final XML format.

Fig. 2: The sentence O cabo é a parte mais básica duma rede. (The cable is the most basic component of a network) in final XML format.

3 The Data Set

In order to develop and test our grammars, a corpus of around 270,000 tokens was collected. The corpus is composed of 33 different documents, which can be integrated in the an LMS as learning objects. These documents are mainly tutorials, PhD thesis, and research papers. They cover three different domains: Information technology for non-experts, e-learning, and information society. This last part is composed by the Section 3 of Calimera guidelines. These guidelines have been compiled by the CALIMERA⁴ Co-ordination Action, funded by the European Commission’s, with the goal of explaining in an accessible way how technologies can be deployed to support digital services designed to meet real user needs.

These three domains are evenly represented in the corpus. One third is composed of documents that are mainly tutorials focusing on basic notions and tools in the domain of computer technology (tutorials on using text editors, HTML, Internet, etc.). The second third of the corpus is composed of documents (mainly articles and PhD thesis) on e-learning concepts, experiments, and governmental policies. The last third is the Section 3 of the Calimera guidelines.

Table 1 shows the composition of the corpus. All the documents were originally in several different file formats (.pdf, .html, etc.). They were processed in order to be converted into a common XML format, conforming to a DTD derived from the XCES DTD for linguistically annotated corpora.

The corpus was then automatically annotated with morpho-syntactic information using the LX-Suite [14].

Portuguese with state of the art performance. This pipeline of modules comprises several tools, namely a sentence chunker (99.94% F-score), a tokenizer (99.72%), a POS tagger (98.52%), and nominal and verbal featureizers (99.18%) and lemmatizers (98.73%).

In Figure 2, we present a sample of the final result. Each sentence is delimited by tags $s$ and each token by the tag $tok$. Of particular interest for the development of our grammars are the attribute base, containing the lemma of each word, the attribute ctag, containing the POS information, and the msd with the morpho-syntactic information on inflection.

Subsequently, the definitory contexts in the corpus were manually annotated. This involves the explicit mark up of the sentence containing the definition (definingText tag), of the definiens (markedTerm tag), and also of the type of definition (defType1 attribute). Figure 3 shows an example of the mark-up of a definitory context.

Besides the three definition types referred to in the beginning of this paper, a fourth category was also introduced for those that are not captured under any of the other three types. Accordingly, the definition typology is made of four different classes whose members were tagged with $is\_def$, for copula definitions, $verb\_def$, for verbal non copula definition, $punct\_def$, for definition whose connector is a punctuation mark, and finally $other\_def$, for all the remaining definitions. Table 2 displays the distribution of the different types of definitions in the corpus.

The domains of Information Society and Information Technology present a higher number of definitions, in particular of copula definitions. This is due to the fact that most documents belonging to these domains were conceived to serve as tutorials for

<table>
<thead>
<tr>
<th>Type</th>
<th>IS</th>
<th>IT</th>
<th>e-Learning</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>is_def</td>
<td>80</td>
<td>62</td>
<td>24</td>
<td>166</td>
</tr>
<tr>
<td>verb_def</td>
<td>85</td>
<td>93</td>
<td>92</td>
<td>270</td>
</tr>
<tr>
<td>punct_def</td>
<td>4</td>
<td>84</td>
<td>18</td>
<td>106</td>
</tr>
<tr>
<td>other_def</td>
<td>30</td>
<td>54</td>
<td>23</td>
<td>107</td>
</tr>
<tr>
<td>Total</td>
<td>199</td>
<td>295</td>
<td>137</td>
<td>661</td>
</tr>
</tbody>
</table>

Table 2: The distribution of types of definitions in the corpus

Table 1: Corpus domain composition (IS: Information Society; IT: Information Technology)

<table>
<thead>
<tr>
<th>Domain</th>
<th>Tokens</th>
</tr>
</thead>
<tbody>
<tr>
<td>IS</td>
<td>92825</td>
</tr>
<tr>
<td>IT</td>
<td>90688</td>
</tr>
<tr>
<td>e-Learning</td>
<td>91225</td>
</tr>
<tr>
<td>Total</td>
<td>274000</td>
</tr>
</tbody>
</table>

³ http://www.calimera.org
non experts, and have thus a more didactic style. The part of the corpus concerning the e-learning domain is mostly composed by research papers and PhD thesis, where the goal is less didactic. In Section 5, we will see how the difference in the objectives of the documents, irrespective of the domain they belong to, may affect the performance of the system.

4 The Rule-Based System

In order to take advantage of the XML format of the corpus, a regular expression based tool for pattern matching that was XML aware was convenient. The tool opted for was the LXtransduce\(^4\). It is a transducer which adds or rewrites XML markup on the basis of the rules provided. Lxtransduce is an updated version of fsgmatch, the core program of LT TTT[4].

LxTransduce allows the development of grammars containing a set of rules, each of which may match part of the input. In case of successful match of the rule with some part of the input, it is possible to replace the matched text or wrap it with a xml tag. Rules may contain simple regular-expression, or they may contain references to other rules in sequences or in disjunctions, hence making it possible to write complex procedures on the basis of simple rules. The outcome of a rule may be used to instantiate variables that can be used later on by other rules. A grammar is thus composed by several rules, where a main rule calls the remaining ones.

All the grammars we developed present a similar structures, they start with simple rules for matching basic expressions, such as conjunctions, articles or nouns. Then further rules may proceed on the basis of the outcome of those rules aiming at matching for complex noun phrases, for instance. As expected, special focus is given to verbs and syntactic patterns surrounding verbs and possibly other connectors in definitory contexts.

A development corpus, consisting of 75% of the whole 270 000 token corpus, was inspected in order to obtain generalizations helping to concisely delimit lexical and syntactic patterns entering in definitory contexts. This sub-corpus was used also for testing the successive development versions of each grammar.

The held out 25% of the corpus was thus reserved for testing the system and to obtain the evaluation results reported below, and was not used in the development phase.

At the present stage of development, three grammars were developed, one for each of the three major types of definitions, namely copula, other verbs, and punctuation definitions.

Copula definitions

In order to develop a grammar for definitions based on the copula verb “to be”, the information contained in the developing corpus was exploited. The copula definitions manually marked in the developing corpus were gathered. Then all information was removed except for the information on part-of-speech, in order to abstract possible useful syntactic patterns for definitory contexts organized around this type of connector. Every such pattern was listed, one per line, and sorted. This permitted to construct a list of different syntactic patterns and associate them with corresponding frequency. Patterns occurring more than three times in the development corpus were implemented in this sub-grammar. The following rules are a sample of the rules in copula sub-grammar.

\[\text{<rule name="Serdef"> <!-- To Be 3rd person pl and s -->} \]
\[\text{<query} \]
\[\text{match="tok[@ctag = 'V' and @base='ser' and (@msd[starts-with(.,'fi-3' )or @msd[starts-with(.,'pi-3' )])]} \]
\[\text{</query}\]
\[\text{.... \}
\[\text{<rule name="copulal"> \}
\[\text{<seq}> \]
\[\text{<ref name="SERdef"/> } \]
\[\text{<best}> \]
\[\text{<seq}> \]
\[\text{<ref name="Art"/> } \]
\[\text{<ref name="adj|adv|prep|" mult="*"/> } \]
\[\text{<ref name="Noun" mult="*"/> } \]
\[\text{</seq>} \]
\[\text{<ref name="tok" mult="*"/> } \]
\[\text{</end/> } \]
\[\text{</seq>} \]
\[\text{</rule>} \]

The second rule is a complex rule composed by other that use other rules, defined in the grammar. This rule matches a sequence composed by the verb “to be” followed by an article and one or more nouns. Between the article and the noun an adjective or an adverb or a preposition can occur. The rule named SERdef matches the verb “to be” only if it occur in the third person singular and plural of the present and future past.

Verbs definitions

To extract definitions whose connector is a verb other than “to be”, first all such verbs appearing in the developing corpus were collected, to which some synonyms were also added. We exploited the possibility provided by Lxtransduce to have a separate lexicon where the verbs, were listed with some information about the use of the verb in definitory contexts. For example, it is possible to specify whether a particular verb occurs in a definition only in passive form, or only in reflexive form, etc.

We decide to exclude some verbs initially collected from the final list because their occurrence in the corpus were very high, but their occurrence in definitions were very low. Their introduction in the final list would not improve recall and would have a detrimental effect on the precision score. The following rule is a sample of how verbs are listed in the lexicon.

\[\text{<rule name="Serdef">} \]

\[\text{<lex word="significar"}> \]
\[\text{<cat>act</cat> } \]
\[\text{</lex>} \]

In this example the verb significar ("to mean") is listed, in his infinitive form that correspond to

\[\text{http://www.ltg.ed.ac.uk/˜richard/ltxml2/lxtransduce- manual.html}\]
5 Outcomes

In the present section we report on the results obtained for the sub-grammars, and for the larger grammar made of the composition of them. Scores for Recall, Precision and F2-measure, for developing corpus (dv) and for test (ts) corpus are indicated. These scores were calculated at the sentence level, that is a sentence (manually or automatic annotated) is considered a true positive of a definition. Recall is the proportion of the sentences correctly classified by the system with respect to the sentences (manually annotated) containing a definition. Precision is the proportion of the sentences correctly classified by the system with respect to the sentences automatically annotated. The option for an F2 measure instead of F1 one is justified by the context in which these grammars are expected to operate.

Since the goal is to help the user in the construction of a glossary, it is important that the system retrieve as many definition candidates as possible. These candidates will be presented to the user in a graphical interface that will allow him to very quickly delete the bad candidates and keep a list with the accepted definition (instead of manually scanning the document to find each one of those definitions, in a much more time consuming way). Hence, to obtain a good recall is more important that obtain a good precision, in case the latter can be traded for the first.

We presented results for the different domains in the corpus, because we expected some difference in the performance due to the different nature of the material belonging to each domain.
Copula definitions

Table 3 displays the results of the copula grammar. These results can be put in contrast with those obtained with a grammar that provides values that can be seen as baseline scores. This simple grammar was developed to extract all sentences as definitive context provided they contain the verbal form of the verb “to be”, in the third person singular and plural of the present and future past and in gerundive and infinitive form (no further syntactic patterns were taken in account). In future experiments more patterns will be introduced in order to improve recall.

Table 3: Results for copula grammar

<table>
<thead>
<tr>
<th></th>
<th>Precision</th>
<th>Recall</th>
<th>F2</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>dv</td>
<td>ts</td>
<td>dv</td>
</tr>
<tr>
<td>IS</td>
<td>0.40</td>
<td>0.33</td>
<td>0.80</td>
</tr>
<tr>
<td>IT</td>
<td>0.26</td>
<td>0.51</td>
<td>0.56</td>
</tr>
<tr>
<td>e-Learning</td>
<td>0.13</td>
<td>0.16</td>
<td>0.54</td>
</tr>
<tr>
<td>Total</td>
<td>0.30</td>
<td>0.32</td>
<td>0.69</td>
</tr>
</tbody>
</table>

Verbs definitions

The results obtained with the grammar for other verbs are not so satisfactory as the ones obtained with the copula grammar. This is probably due to the larger diversity of patterns and meaning for each such verb. In order to improve these results a deeper analysis of each verb pattern is required.

Punctuation definitions

As can be seen in Table 2, only 4 definitions of this type occur in the documents of the IS domain and 18 in e-learning domain. Consequently, this grammar for punctuation definitions ended up by scoring very badly in these documents. Nevertheless, the global evaluation result for this sub-grammar is better than the results obtained with the grammar for other verb definitions.

All-in-one

Finally, Table 7 presents the results obtained by a grammar that combines all the other three sub-grammars below. This table gives the overall performance of the system based on the grammars developed so far, that is this result represents the performance the end user will face when he will be using the glossary candidate detector.

To obtain the precision and recall score for this grammar, it is not anymore necessary to take into account the type of definition. Any sentence that is correctly tagged as a definitive context (no matter which definition type it receives) will be brought on board.

As can be seen, the recall value remains quite high, 86%, while it is clear that for the precision value (14%), there is much room for improvement yet.

As expected, the results obtained with documents from the Information Society and Information Technology domains are better than the results obtained with documents from the e-Learning domain. This confirms our expectation drawn from the style and purpose of the material involved. Documents with a clear educational purpose, like those from IS and IT sub-corpora, are more formal in the structure and are more directed towards explaining concepts, many times via the presentation of the associated definitions. On the other hand, documents with a less educative purpose present less explicit definitions and for this reason it is more difficult to extract definitive contexts from them using basic patterns. More complex pattern and a deep grammar are likely to be useful in dealing with such documents.

Also worth noting is the fact that though the linguistic annotation tools used score at the state of the art level, the above results can be improved with the improvement of the annotation of the corpus. A few errors in the morpho-syntactic annotation were discovered during the development of the grammars that may affect the performance of the grammars.

Another issue about evaluation of our result is that determining the performance of a definition extraction system is not a trivial task. Many authors have pointed out that a quantitative evaluation as the one we carried out in this work may not be completely appropriate [12]. A qualitative approach to evaluation has to be taken in account (see for example [6]). For this reason we are planning to evaluate the effectiveness and the usefulness of the system with real users, by the means of user scenario methodology.

7 Conclusions and Future Work

In this work, we presented preliminary results of a rule-based system for the extraction of definitions from corpora. The practical objective of this system is to support the creation of glossaries in e-learning environments, and it is part of the LT4eL project aiming at improving e-learning management systems with the human language technology.
Table 6: Results for punctuation grammar

<table>
<thead>
<tr>
<th></th>
<th>Precision</th>
<th>Recall</th>
<th>F2</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>dv</td>
<td>ts</td>
<td>dv</td>
</tr>
<tr>
<td>Calimera</td>
<td>0.00</td>
<td>00</td>
<td>0.00</td>
</tr>
<tr>
<td>IS</td>
<td>0.48</td>
<td>0.43</td>
<td>0.60</td>
</tr>
<tr>
<td>e-Learning</td>
<td>0.05</td>
<td>00</td>
<td>0.58</td>
</tr>
<tr>
<td>Total</td>
<td>0.19</td>
<td>0.28</td>
<td>0.64</td>
</tr>
</tbody>
</table>

Table 7: The combined result

The better results was obtained with the system running over documents that are tutorials on information technology, where it scored a recall of 69% and a precision of 33%. For less educational oriented documents, 59% and 11%, respectively, was obtained.

We also studied its performance on different types of definitions. The better results were obtained with copula definitions, with 67% of recall and 51% of precision, in the Information Technology domain.

Compared to work and results reported in other publications concerning related research, our results seem thus very promising. Nevertheless, further strategies should be explored to improve the performance of the grammar, in particular its precision.

In general, we will seek to take advantage of a module that allows deep syntactic analysis, able to deal with anaphora and apposition, for instance. At present, a grammar for deep linguistic processing of Portuguese is being developed in our group [3]. We plan to integrate this grammar working in our system.

It is also worth noting that some authors [2, 9] noticed that some verbs are better clues for definitions than others, and that it is possible to measure this property in order to decide which verbs include in the searching patterns and which to exclude. For example, verbs that occur in many sentences, but only in few cases introduce a definition may perhaps be ignored. A more permissive approach is to parametrize the system in order to let the user choose whether he wants to use all the patterns in the grammar, obtaining a good recall but a worst precision, or to restrict the extraction only to the more promising patterns, thus improving precision at the expense of a worst recall. The same strategy can be applied to all type of definitions, not only to verb definitions.

Regarding punctuation definition, the pattern in the actual grammar can also be extended. At present, the pattern can recognize sentences composed by a simple noun followed by a colon plus the definition. Other rules with patterns involving brackets, quotation marks, dashes will be integrated.

Finally, in this work we ignored an entire class of definitions that we called “other definition”, which represents 16% of all definitions in our corpus. These definitions are introduced by lexical clues such as that is, in other words, etc. This class also contains definitions spanning over several sentences, where the terms to be defined appears in the first sentence, which is then characterized by a list of features, each one of them conveyed by expressions occurring in different sentences. These patterns need thus also to be taken into account in future efforts to improve the grammar and its results reported here.

References


Grammar-based Automatic Extraction of Definitions. Applications for Romanian

Adrian Iftene
Faculty of Computer Science
“Al. I. Cuza” University of Iaşi
Romania
adiftene@info.uaic.ro

Diana Trandabăț
Faculty of Computer Science
“Al. I. Cuza” University of Iaşi
Romania
dtrandabat@info.uaic.ro

Ionuț Pistol
Faculty of Computer Science
“Al. I. Cuza” University of Iaşi
Romania
ipistol@info.uaic.ro

Abstract
This paper presents part of our work in the LT4eL project [1] regarding the grammar developed by the Romanian team in order to extract definitions from texts. Some qualitative results come in order to evaluate our grammar rules. Among the applications of this kind of grammar we will discuss the possible inclusion of the grammar rules into a question answering system in order to extract answers for definition type questions. Another possible usage of those rules envisages the extraction of supplementary knowledge from linguistic resources like Wikipedia. The benefits of such an extra-knowledge resource are evident in textual entailment systems, where some resources like WordNet, Acronyms database or DIRT cannot cover all the requirements of the system.

Keywords
Definition Extraction, Question Answering, Textual Entailment.

1. Introduction
Under the framework of the FP6 European project LT4eL¹ (Language Technology for e-Learning), an environment for collecting and (semi)automatic exploiting language resources has been created, as the main objective of the project is to provide functionalities based on language technologies and to integrate semantic knowledge in Learning Management Systems. The first step was to create, for the 9 languages involved (Bulgarian, Czech, Dutch, English, German, Maltese, Polish, Portuguese and Romanian), a multilingual corpus, partially parallel, of almost 5.5 million words, annotated and uploaded on the project’s portal² [2].

In order to improve the management, distribution and retrieval of the learning material by automatically attaching metadata (such as keywords and definitions) to any text, a necessary step was the observation of those metadata in the annotated corpus. Therefore, the corpus was manually annotated to keywords (the words or expressions that a user of the Learning Management System would use to retrieve documents referring that notion), definitions of various terms and semantic concepts. Using the manual annotated documents, a grammar was created for the automatic identification of definitions in texts. Apart for the use in this project, we present also two other envisaged applications.

After briefly describing the formats of the learning materials used on the LT4eL project and the annotation of the definitions, Section 3 will describe the Romanian grammar. Section 4 presents several possible applications of the grammar in order to improve the quality of complex systems like a Question Answering system and a Textual Entailment system, before drawing some conclusions and further directions.

2. The learning material
The linguistic resources - called learning objects (LO) - were selected according to their language, format (.doc, .pdf, plain text, .html or other), domain (broadly: the use of computers in education), or Intellectual Properties Rights. After their automatic conversion to XML [2], the objects were linguistically annotated (tokens, part-of-speech, lemma, chunks) and converted into a unitary XML format³ with basic formatting information extracted from the .txt and/or .html versions of the document, called basic XML. This version of the document was used for the manually annotation of keywords and definitions. The corpus collected for the Romanian language contains 56 documents summing approx. 700,000 words.

For the annotation process, we understood by a definition a concise explanation, description of a concept’s meaning or type. A definition has two parts: the defined term and the defining context. An example of definition extracted from the Romanian corpus is:

Ro: [Def.prim] Def.part1 prevăzută în tratatul de la Roma și mai apoi în cel de la Maastricht (este caracterizată de drepturi, de obligații și de implicarea în viața politică) Def.part2.

¹http://www.lt4el.eu/
²http://consilr.info.uaic.ro/uploads_lt4el/
³http://ufal.mff.cuni.cz/~spousta/lt4el/html2xml/LT4ELBase.dtd
where the defined term is Cetățenia Uniunii Europene (En: European Union citizenship), and the definition is marked between brackets [ ]. One can see that not the entire sentence was considered to be the definition, since an explanatory attributive sentence can be considered outside the definition scope. In order to obtain this split, the definition is marked as continuing, and the parts of the defining context are marked successively.

An example of annotated learning material, in basic XML format, is presented in figure 1, for the definition discussed above.

![Figure 1. Example of manually annotated definition](image)

### 3. Romanian grammar

For the automatic annotation of the definitions found in the learning objects, the approach throughout the LT4eL consortium was to develop local grammars for the 9 represented languages (English, Dutch, German, Polish, Bulgarian, Maltese, Czech, Romanian, and Portuguese) to extract definition patterns. The main difficulties addressed were due to the different manner of expressing the definitions, especially if the lexicalization of the introducing words (like the verbs “is”, “represents” etc.) were to be kept minimal. Other problems were raised by interrupted definitions or by the ending point of a definition, if the definition ends before the sentence punctuation marks.

The linguistic information from the manually annotated definitions is used as starting point in identifying possible grammar patterns that could form a definition. Previous work within this area shows that the use of local grammars which match syntactic structures of defining contexts are really useful when deep syntactic and semantic analysis is not present [3, 4].

The creation of the Romanian grammar started with some simple rules and their application over the manual annotated files. Observing repeatedly the cases left aside, we improved the grammar with more complex rules and different lexical items. The drawback of this approach is that the results are corpus dependent.

#### 3.1 Categorization of Definitions

Definitions have been categorized in six types in order to reduce the search space and the complexity of rules. The types of definitions observed in Romanian texts have been classified as follows:

1. `is_def` – Definitions containing the verb “este” (En: is).

Example: “Prescurtare pentru Hyper Text Mark Up Language, HTML este tot un protocol folosit de World...”

![Definition](image)
Wide Web.” (En: Abbreviation for Hyper Text Mark Up Language, HTML is also a protocol used by World Wide Web).


Example: “Poșta electronică reprezintă transmisia mesajelor prin intermediul unor rețele electronice.” (En: Electronic mail represents sending messages through electronic networks).

3. “punct_def” – Definitions which use punctuation signs like the dash “—”, brackets“(“), comma “,” etc.

Example: “Bit – prescurtarea pentru binary digit” (En: Bit – shortcut for binary digit)

4. “layout_def” – Definitions that can be deduced by the layout: they can be included in tables when the defined term and the definition are in separate cells or when the defining term is a heading and the definition is the next sentence.

Ro:

<table>
<thead>
<tr>
<th>Organizarea datelor</th>
<th>Cel mai simplu mod de organizare este cel secvențial.</th>
</tr>
</thead>
</table>

En:

Data organizing The simplest method is the sequential one.

5. “pron_def” – Anaphoric definitions, when the defining term is expressed in a precedent sentence and it is only referred in the definition, usually pronoun references.

Example: “…definirii conceptului de baze de date. Acesta descrie metode de modelare ale problemelor reale în scopul definirii unor structuri care să eliminate redundanțele în stocarea datelor.” (En: …defining the database concept. It describes methods of modeling real problems in order to define structures which eliminate redundancy in data collecting.)

6. “other_def” – Other definitions, which cannot be included in any of the previous categories. In this category are constructions which do not use verbs as the introducing term, but a specific construction, such as “i.e.”

Example: “triunghi echilateral, adică cu toate laturile egale” (En: equilateral triangle i.e. having all sides equal).

The distribution of the definition types in Romanian corpus is presented in table 1:

<table>
<thead>
<tr>
<th>Type</th>
<th>Manual</th>
<th>%</th>
<th>Automatic</th>
<th>%</th>
</tr>
</thead>
<tbody>
<tr>
<td>is_def</td>
<td>70</td>
<td>33.8</td>
<td>204</td>
<td>32.8</td>
</tr>
<tr>
<td>verb_def</td>
<td>116</td>
<td>56.0</td>
<td>272</td>
<td>43.8</td>
</tr>
<tr>
<td>punct_def</td>
<td>15</td>
<td>7.2</td>
<td>124</td>
<td>20.0</td>
</tr>
<tr>
<td>layout_def</td>
<td>2</td>
<td>1.0</td>
<td>21</td>
<td>3.4</td>
</tr>
<tr>
<td>pron_def</td>
<td>4</td>
<td>2.0</td>
<td>0</td>
<td>0.0</td>
</tr>
<tr>
<td>Total</td>
<td>207</td>
<td></td>
<td>621</td>
<td></td>
</tr>
</tbody>
</table>

Table 1: Distribution of the definitions into types

The above table states that 33% of the total of definitions (both for manual and automatic definitions) are introduced by the verb a fi (“to be”). An interesting observation is that the definitions introduces by something else than a verb sums approximately 10% of the manually and around 23% of the automatic detected definitions. The big difference indicates the fact that more manual markups are needed in our corpus.

3.2 The grammar

The XML transducer Lxtransduce developed in [5] is used to match a grammar against several files in XML format. Lxtransduce supplies a format for the development of grammars either in pure text or in XML documents. The grammars files are XML documents built according a specific DTD. In our grammar, we created rules for each type of defining context and a “main” rule used to call the rules one-by-one at different runs.

All these rules were observed through the observation of the manually extracted definitions.

3.3 Grammar rules

The grammar for extracting Romanian definitions starts with several simple rules which identify different part of speech.

For instance, the rule presented in figure 2 identifies adverbs by looking in the tag attribute at the first letter. If this first letter is “r” then we have detected an adverb:

<rule name="Adv">
  <query match="tok[@ctag[starts-with(",',r')]]"/>
</rule>

Figure 2: Simple grammar rule

Those rules can be combined in order to obtain more complex rules. Figure 3 presents an example of entities that are actually combination of simple entities:

<rule name="Nominal">
  <seq>
    <ref name="undef" mult="?"/>
    <ref name="Adj" mult="?"/>
    <ref name="Noun"/>
    <ref name="Adj" mult="?"/>
  </seq>
</rule>

Figure 3: Composed grammar rule
After identifying the different structures, the general rules are created. Figure 4 presents the grammar rules for the “is_def” definitions. The lemma for the verb must be “fi” (En: be) and its part of speech label (contained in the <ctag> tag) must be “vmip3” (verb main indicative present third person). Another condition is that we have an entity “DefNominal” or “UndefNominal” (definite or indefinite noun), entity defined through a complex rule as the one presented in figure 3.

![Figure 4: “is_def” grammar rule](image)

Another type of rule is the one that identifies the end of the sentence, thus considered the end of the definition (Figure 5):

![Figure 5: Sentence boundary rule](image)

### 3.4 Grammar evaluation

The XML transducer, lxtransduce, is used to match our grammar, conforming to an XML format specified within the tool by using XPath. When a match is found, a rule is applied and a definition is marked in the file. The tool was run on every type of definitions and the results are presented in table 2 (P = precision, R= recall and F2 = F measure):

<table>
<thead>
<tr>
<th>Definition Type</th>
<th>Result</th>
</tr>
</thead>
</table>
| is_def | Sentence-level matching: 
P: 0.5366, R: 1.0, F2: 0.7765 
Token-level matching: 
P: 0.0648, R: 0.3328, F2: 0.14 |
| verb_def | Sentence-level matching |

**Table 2: Romanian grammar evaluation**

For each definition type, the precision and recall were calculated at two levels: at the token level and at the sentence level [5]. At token level, precision is understood as the number of tokens simultaneously belonging to a manual definition and an automatically found definition, divided by the number of tokens in automatically found definitions. Correspondingly, recall is the ratio of the number of tokens simultaneously in both definition types to the number of tokens in manual definitions. At the sentence level, a sentence is taken as a manual or automatic definition sentence if and only if it contains a (part of a), respectively, manual or automatic definition. Therefore, precision and recall are calculated analogous to the values used in token level.

The best results are obtained for the definitions introduced by verbs (the most common cases). Among those, the definitions introduced by the verb “is” are the most difficult to identify, since the verb appears very frequently in the language and many cases of non-defining contexts are wrongly considered. An example of a wrong annotation is:

<definition>o asemenea practica este recomandata in cadrul documentelor complete</definition> (En: Such a practice is recommended within complex documents.)

For the pron_def and other_def, although there are several manually annotated definitions, the rules considered aren’t accurate enough and require improvement.

### 4. Definition Extraction Applications

#### 4.1 Definitions extraction in a Question Answering system

Question Answering (QA) can be defined as the task which takes a question in natural language and produces one or more ranked answers from a collection of documents. The QA research area has emerged as a result of a monolingual English QA track being introduced at TREC\(^4\).

QA systems normally adhere to the pipeline architecture composed of three main modules: question analysis, paragraph retrieval and answer extraction [5].

The first module is the question analyzer. The input to this module is a natural language question and the output is one or more question representations which will be used at subsequent stages. At this stage most systems identify the semantic type of the entity sought by the question, determine additional constraints on the answer and the question type and extract keywords to be employed by the next module.

The paragraph retrieval module is typically based on a conventional information retrieval search engine in order to select a set of relevant candidate paragraphs/sentences from the document collection.

At the last phase, answer extraction and ranking, the representation of the question and the candidate answer-bearing snippets are compared and a set of candidate answers are produced and then ranked using likelihood measures.

Accordingly to the answer type, we have the following type of questions [7]:

- **Factoid** – The question refers to a single answer, as for instance: “Who discovered the oxygen?” or “When did Hawaii become a state?” or “What football team won the World Cup in 1992?”.

- **List** – The answer to a list question is an enumeration: “What countries export oil?” or “What are the regions preferred by the Americans for holidays?”.

- **Definition** – These questions require a complex processing of the texts and the final answer consist of a text snippet or is obtain after summarization of more documents: “What is quasar?” or “What is a question-answering system?”.

We present below the system developed this year for the QA@CLEF2007 competition and the way we dealt with definition questions. This system is based on the cross-lingual system built last year for the English-Romanian task [8].

In the case of DEFINITION questions, the candidate paragraphs extracted in the information retrieval phase are matched against a set of rules of our Romanian grammar. The rules from the Romanian grammar were translated from lxtransduce format to Perl patterns. The reason of this rule transformation is that the QA system uses an annotation of the corpus (lemma part of speech, name entity, etc.) different form the file format considered by lxtransduce, and the size of the corpus didn’t allow for any format changes.

Thus, each possible definition having as defined term the focus of the question is extracted and added to a set of candidate answers, together with a score revealing the reliability of the pattern it matched.

The set of NPs in the snippet is also investigated to detect those NPs containing the defined term surrounded by other functional words (this operation is motivated by cases like the Atlantis space shuttle, where the correct definition for Atlantis is space shuttle). The selected NPs are added to the set of candidate answers with a lower score.

The set of candidate answers is then ordered according to the score attached to each answer and to the number of

---


---

Figure 6: Architecture of the Romanian Question Answering System

other candidate answers it subsumes. The highest ranked candidate answers are presented as final answers. Figure 6 presents the architecture of the Romanian Question Answering System that competed to the QA@CLEF competition this year.
4.2 Building a Background Knowledge database for Textual Entailment

Within the Textual Entailment competition\(^6\) [9], participants in the evaluation exercise are provided with pairs of small text snippets (one or more sentences in English), which are named Text-Hypothesis (T-H) pairs. The participants must build a system that, for each pair, should say if there is entailment or no (if the text entails the hypothesis). The complexity of this task comes from the complexity of the applications needed in order to assess a semantic relationship between text segments: Information Retrieval (IR), Question Answering (QA), Information Extraction (IE), and Text Summarization (SUM).

Our system architecture is based on a peer-to-peer network design, in which neighboring computers collaborate in order to obtain the global fitness for every text-hypothesis pair [10]. The main idea is to transform the hypothesis making use of extensive semantic knowledge from sources like DIRT, WordNet, Wikipedia, acronyms database, etc. Additionally, we built a system to acquire the required extra background knowledge and applied complex grammar rules for rephrasing. We calculated then the distance between the dependency trees associated to the initial text and to the new hypothesis. Eventually, based on the computed score, we decide for which pairs we have entailment.

There is information that cannot be deduced from the databases and thus we require additional means of gathering extra information such as the one presented in table 3.

| Argentine [is] Argentina |
| Netherlands [is] Holland |
| 2 [is] two |
| Los Angeles [in] California |
| Chinese [in] China |

Table 3: Background knowledge

The background knowledge was built semi-automatically, for the named entities and for numbers from the hypothesis without correspondence in the text. For these named entities, we used a module to extract from Wikipedia snippet with information related to them. In the snippets extracted from Wikipedia we try to identify the defining texts. For each such context:

a) we identify the “core” of the definition (which is either the verb “to be” or another definition introducing verb or a punctuation mark).

b) we extract from the left hand part of the “core”: all the name entities (left NEs)

c) we extract from the right hand side of the “core”: all name entities (right NEs)

d) we compute the Cartesian product between left NEs and right NEs and add the resulting pairs to the existing background knowledge base.

Subsequently, we use this file with snippets (Argentina for instance in Figure 8) and the several patterns in order to identify the relations between the entities in The goal in this endeavor is to identify a known relation between two named entities.

ar | calling_code = 54 | footnotes = Argentina also has a territorial dispute Argentina', , Nacion Argentina (Argentine Nation) for many legal purposes), is in the world. Argentina occupies a continental surface area of Argentina national football team

Figure 8: Snippets extracted for Argentina

If such a relation is found, we make the association and save it to an output file. For our case only line “Argentina [is] Argentine” is added to the background knowledge. Another example of extracted knowledge for the NE “Netherlands” is presented in table 4:

| Netherlands [is] Dutch |
| Netherlands [is] Nederlandse |
| Netherlands [is] Antillen |
| Netherlands [in] Europe |
| Netherlands [is] Holland |
| Antilles [in] Netherlands |

Table 4: Results for Netherlands

All these relations are added to the background knowledge database and will be used at the next run. Not all relations are correct, but the relation “Netherlands [is] Holland” will help us at the next run.

Our patterns identify two kinds of relations between words:

- “is”, when the module extracts information in the form: ‘Argentine Republic’ (Spanish: ‘Republica Argentina’, IPA)’ or when explanations about the word are between brackets, or when the extracted information contains one verb used to define something, like “is”, “define”, “represent”: ‘2’ (‘two’) is a number.

- “in” when information is of the form: ‘Chinese refers to anything pertaining to China’ or in the form Los Angeles County, California, etc.

In the case of the ‘is’ relation, we use the same rules as those in the grammar and for the ‘in’ relation we add more specific rules. Usually, ‘in’ relation appear between specific named entities like LOCATION or COUNTRY or ORGANIZATION, and in these cases it is determined by specific words like: “in”, “located”, “from”, etc.

In order to be able to see each component’s relevance, the system was run in turn with each component removed.

\(^6\) http://www.pascal-network.org/Challenges/RTE/  
\(^7\) http://en.wikipedia.org/wiki/Main_Page
### Table 5: Components relevance

<table>
<thead>
<tr>
<th>System Description</th>
<th>Precision</th>
<th>Relevance</th>
</tr>
</thead>
<tbody>
<tr>
<td>Full system</td>
<td>0.6913</td>
<td>-</td>
</tr>
<tr>
<td>Without DIRT</td>
<td>0.6876</td>
<td>0.54 %</td>
</tr>
<tr>
<td>Without WordNet</td>
<td>0.6800</td>
<td>1.63 %</td>
</tr>
<tr>
<td>Without Acronyms</td>
<td>0.6838</td>
<td>1.08 %</td>
</tr>
<tr>
<td>Without BK</td>
<td><strong>0.6775</strong></td>
<td><strong>2.00 %</strong></td>
</tr>
<tr>
<td>Without Negations</td>
<td>0.6763</td>
<td>2.17 %</td>
</tr>
<tr>
<td>Without NEs</td>
<td>0.5758</td>
<td>16.71 %</td>
</tr>
</tbody>
</table>

We can notice that the Background Knowledge resource is very important, and represent 2 % from total precision of the system.

The system presented here participated this year for first time in the RTE3\(^8\) competition. From 26 competing groups, we obtained the third place with a precision of 69.13%.

### 5. Conclusions and Future work

This paper presented the Romanian grammar used in the European LT4eL project to automatically extract definitions from texts. The definitions were devised in 6 types, and the results of the system for each definition type were presented. The automatic discovery of definitions using a rule-based method can significantly improve a question answering system (for definition type questions) or the background acquisition useful for a textual entailment system.

A necessary further step in the improvement of the Romanian grammar is applying it to a new corpus, in order to verify that all the definitions extracted are real defining contexts.

### 6. Acknowledgements

This paper present the work of the Romanian team in the European project Language Technologies for e-Learning, STREP 027391 in FP6-2004-IST-4, and within the CEEX Rotel project number 29.

Special thanks goes to the other members of the Romanian team in the LT4eL project, Dan Cristea and Corina Forăscu, also involved in developing the Romanian grammar.

We also acknowledge the help provided by Claudia Borg and her analysis program in improving the grammar performances.

### 7. References


\(^8\) http://www.pascal-network.org/Challenges/RTE3/
Keyword extraction for metadata annotation of Learning Objects

Lothar Lemnitzer and Paola Monachesi
Tübingen University and Utrecht University
lothar@sfs.uni-tuebingen.de Paola.Monachesi@let.uu.nl

Abstract
One of the functionalities developed within the LT4eL project is the possibility to annotate learning objects semi-automatically with keywords that describe them. To this end, a keyword extractor has been created which can deal with documents in 8 languages. The approach employed is based on a linguistic processing step which is followed by a filtering step of candidate keywords and their subsequent ranking based on frequency criteria.

Two tests have been carried out to provide a rough evaluation of the performance of the tool and to measure inter annotator agreement in order to determine the complexity of the task and to evaluate its performance with respect to human annotators.

1 Introduction

eLearning aims at replacing the traditional learning style in which content, time and place are predetermined with a more flexible, customized process of learning. While in traditional learning, the instructor plays an intermediate role between the learner and the learning material, this is not always the case within eLearning since learners have the possibility to combine learning material and to create their own courses. However, a necessary condition is that content should be easy to find and metadata plays a crucial role to this end. It provides a common set of tags that can be viewed as data describing data. Metadata tagging enables organizations to describe, index, and search their resources and this is essential for reusing them.

In the eLearning community, various metadata standards have emerged to describe eLearning resources with the IEEE LOM being the most widespread and well-known standard. Providing metadata, however, is a tedious activity and it is not widely accepted by content providers and authors as part of their work. This has, however, the highly undesirable consequence that content becomes less visible and more difficult to retrieve.

One of the goals of the LT4eL project is to show that language technology can provide significant support for this task. The solution we offer is to provide a Language Technology based functionality, that is a keyword extractor which allows for semi-automatic metadata annotation of the learning objects within a Learning Management System (LMS). Keyword extraction is the process of extracting a few salient words or phrases from a given text and using these words to represent the content of the text. Keyword extraction has been widely explored in the natural language processing and information retrieval communities and in our project we take advantage of the techniques and the results achieved in these areas and adapt them to the eLearning context.

More specifically, our approach employs statistical measures in combination with linguistic processing to detect salient words which are good keyword candidates.

It should be noticed, however, that keyword and keyphrase extractors have been provided mainly for English, cf. [13], [5], [16], [18], [17], [14], [10], [6]. One innovative aspect of our project is that we provide this functionality for all the eight languages represented in our project, that is Bulgarian, Czech, Dutch, English, German, Polish, Portuguese and Romanian and we embed this significant result within the eLearning context. Another innovative feature is that keyphrases are extracted in addition to keywords. This responds to findings that users frequently use keyphrases to describe a document, cf. [7].

More generally, the main objective of the LT4eL project is to show that the integration of Language Technology based functionalities and Semantic Web techniques will enhance the management, distribution and retrieval of the learning material within Learning Management Systems.

The rest of the paper is organized as follows. In section 2, we outline the architecture of the keyword extractor, including the methods we are using for ranking keywords and we point out the innovative features of our tool. The quantitative evaluation of the tool is discussed in section 3 and results obtained are analyzed. The keyword extractor has been integrated into the learning management system ILIAS. We show the result in section 4. Finally, section 5 contains our conclusions and plans about future work.

2 The Keyword Extractor

The task of a keyword extractor is to automatically identify a set of terms in a document that best describes its content. Keyword extractors have been employed to identify appropriate entries for building an automatic index for a document collection and have been used to classify texts. Keyword extraction has also been considered in combination with summarization ([16], [10], [18]). An additional use is to identify automatically relevant terms that can be employed...
in the construction of domain-specific dictionaries or more recently of domain ontologies ([13]).

In the LT4eL project, we have adapted current techniques for term extraction in order to develop a keyword extractor which is employed for the semi-automatic metadata annotation of the learning objects. We have privileged a simple approach which is based on a frequency criterion to select the relevant keywords in a document which has been complemented with a linguistic processing step.

This method was found to lead to poor results, as claimed in [10] and consequently alternative methods were explored in the literature. They are mainly based on supervised learning methods, where a system is trained to recognize keywords in a text, based on lexical and syntactic features. However, given the specific application which has been envisaged in our project, that is the extraction of relevant keywords for metadata generation, we have chosen an approach which could be easily adapted to several languages. In the LT4eL project, we use the same algorithm for all the languages under consideration while we encode the language specific differences in the language model. It should be noticed that a machine learning approach didn’t seem a possible option: given the small corpus of learning objects available for each language, we wouldn’t have had enough training data at our disposal.

The keyword extractor accepts linguistically annotated input and outputs a list of suggested keywords to be included in the LOM metadata (cf. figure 1).

More specifically, the input for the keyword extractor is constituted by learning objects of various formats, e.g. PDF and DOC which are converted into HTML. From this intermediary representation an XML format is generated which preserves basic layout features of the original texts. Linguistic information is added to this format. The process yields a linguistically annotated document in an XML format which is derived from the XCESAna standard for linguistically annotated corpora. The linguistic annotation comprises: a) the base form of each word; b) the part of speech of this base form; c) further morphosyntactic features of the word form which is used in the text.

This linguistic information, which is extracted from the corpus of learning objects, is added to the language model for the specific language which consists of three parts:

- **Lexical units**: they represent the combination of a lemma and a part of speech tag. They are the basic units on which statistics are calculated and they are returned as possible keywords. Only those lexical units that can occur as possible keywords are retained – mainly nouns, proper nouns and unknown words.
- **Word Form Types**: they represent the actual form of the lexical unit in the input file in combination with their morphological information.
- **Documents**: they represent the documents which constitute the corpus including their names and domains. The two domains of our documents are information technologies for the non-expert and eLearning.

Potentially interesting sequences of words are extracted using the suffix array data structure [15] but a condition is that they must appear at least twice in the document. Afterwards, filtering occurs on the basis of language specific information and sequences longer than a certain threshold are discarded. In general, sequences comprising up to 3 words are retained.

The list of candidate keywords is ranked by their saliency and to determine it an approach based on frequency has been adopted.

As already mentioned, keywords are those terms that best identify the text and represent what the text is about (i.e. the topics of a text). They tend to occur more often in that text than could be expected if all words were distributed randomly over a corpus.

A well-established way to measure the distribution of terms over a collection of documents is TFIDF, cf. equation 1.

\[
TFIDF \quad \text{where} \quad IDF = \log_2 \frac{N}{df} \quad (1)
\]

Church argued that Poisson distributions or mixtures of Poisson distributions of words in texts are quite useful statistics (cf. [4] and equation 2).

\[
\pi(k; \theta) = \frac{e^{-\theta} \theta^k}{k!} \quad (2)
\]

While the distribution of e.g. function words like of, the, it is close to the expected distribution under the Poisson distribution model, good keyword candidates deviate significantly from the expectation. The score of this deviation can be used as a statistics by which the lexical units are ranked ([3]). The deviation of the observed distribution of a word from the expected distribution under the Poisson model, i.e predicted IDF (cf. equation 3) is called Residual IDF (short: RIDF, cf. equation 4).

\[
-\log_2 (1 - e^{-\theta}) \quad \text{where} \quad \theta = \frac{cf}{N} \quad (3)
\]

\[
IDF - PredictedIDF \quad (4)
\]

A closer look at the formula reveals that RIDF does not take the term frequency in the analysed document...
into account. It measures the overall distribution of a term in a collection of documents, but not the frequency of a term in an individual document. Since we like to know more about the role which a term plays in a particular document, we extended this metric with a factor which measures the frequency of the term in that document and arrived at a statistics which we call Adjusted Residual IDF (short: ADRIDF, cf. equation 5).

\[ ADRIDF = RIDF \sqrt{tf} \]  

In our project, we have implemented and evaluated the appropriateness of these statistical measures in ranking the most relevant keywords from our multilingual learning objects. In section 3, the results are discussed in detail.

The keyword extractor is built to deal with a wide range of languages: Bulgarian, Czech, Dutch, English, German, Polish, Portuguese and Romanian. This is a relevant result since techniques to extract keywords and keyphrases have been usually tested on English and never on such a variety of languages.

On the one hand, the management of such a wide range of languages makes it necessary: a) to build a common annotation format for all annotated corpora and b) to keep the language specific components of the tool as lean as possible. On the other hand, the multilingual aspect of the development gives us the chance to broadly evaluate the performance of the tool and its underlying information extraction methods, as discussed in detail in the next section.

The evaluation strategy must be both formative – i.e. inform the development of the tool, in particular the language-specific settings – and summative – i.e. assess the performance of the final tool. It has to be both intrinsic – i.e. assess the performance of the tool in isolation – and extrinsic – i.e. assess the performance of the tool as part of a learning environment.

As discussed more at length in section 3, the novelty of our application makes it difficult to adapt current evaluation tests for our purposes. On the other hand, we do need to assess the performance of the tool and verify that it achieves acceptable results before integrating it in the Learning Management System. Therefore we have to accept the limitation of these tests and work towards the development of new ones more fit to the purpose. It should be noticed that a non-optimal performance of the tool in the intrinsic evaluation might still lead to an appropriate behavior of the keyword extractor in the extrinsic evaluation.

In the development of the keyword extractor, special attention has been devoted to multiword terms. A first analysis of the manually selected keywords revealed, that for some languages a substantial amount of them is multi word. E.g. for Polish we have 67 % keyphrases of two or more words. For other languages, e.g. German, multi word key phrases do not play a significant role, see table 1 for details.

We therefore put some effort to properly deal with these items and several tests have been carried out to detect the most appropriate length for multiword keywords and possible variation due to language. We followed the approach of Yamamoto and Church, cf. [15], to effectively identify and extract recurrent multi-word key phrases up to a predefined length. Additionally, we used linguistic information to further restrict this set of multi-word key phrases, e.g. to exclude phrases which end in a preposition. Statistically, multi-word phrases are treated as single words.

Providing users with multiword keywords raises the issue of which should be the best way to represent them. We have noticed that, at least for some languages such as Polish, a sequence of base forms looks quite unnatural. Therefore we have decided that the selected multi-word keywords are represented by their most frequent attested forms.

We refer to [9] for additional details on the use of the keyword extractor within the LT4eL project.

### 3 Evaluation of the keyword extractor

As already mentioned, it is not easy to establish which is the best way to evaluate the keyword extractor. In our approach, we have mainly used statistical measures which are usually employed for term extraction but our application is different from the construction of a domain ontology or a terminological lexicon.

In the case of these applications, precision is usually measured by dividing the extracted terms which are appropriate for a given domain by the number of accepted terms. On the other hand, this approach cannot be used to evaluate our keyword extractor given the application envisaged in our project. Recall that the identification of appropriate keyword that describe the document will be employed for the semi-automatic metadata annotation of the learning objects. Thus, appropriate keywords will be much more restricted in number than appropriate terms for a given domain. In addition, the choice of keywords for a given document is often determined by the context of its use and we thus expect there to be variation among annotators in determining which keywords are appropriate for a given document.

Ultimately, the best way to validate the keyword extractor might be in the context of the Learning Management System, that is by authors or content providers which will employ it to annotate learning objects with LOM metadata semi-automatically. On the other hand, the keyword extractor which will be integrated into the LMS should be optimized for this task and thus a methodology should be developed to verify its appropriateness and to eventually improve

<table>
<thead>
<tr>
<th>language</th>
<th>Keywords (%)</th>
<th>Keyphrases (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bulgarian</td>
<td>57</td>
<td>43</td>
</tr>
<tr>
<td>Czech</td>
<td>73</td>
<td>27</td>
</tr>
<tr>
<td>Dutch</td>
<td>75</td>
<td>25</td>
</tr>
<tr>
<td>English</td>
<td>38</td>
<td>62</td>
</tr>
<tr>
<td>German</td>
<td>90</td>
<td>10</td>
</tr>
<tr>
<td>Polish</td>
<td>33</td>
<td>67</td>
</tr>
<tr>
<td>Portuguese</td>
<td>86</td>
<td>14</td>
</tr>
<tr>
<td>Romanian</td>
<td>70</td>
<td>30</td>
</tr>
</tbody>
</table>

Table 1: Percentages of keywords and keyphrases per language
its performance.

There are certain parameters which have been taken into account in this process: a) the language(s) of the learning objects and the corresponding language models which influence the preselection of keyword candidates; b) the maximal length of keyphrases to be extracted; c) several distributional statistics. We also envisage to employ additional features to select and rank keywords, as discussed in section 5 and once these features have been implemented, their impact has also to be evaluated.

Therefore, the verification must be formative in a sense that it is repeated several times in the development cycle. It informs the optimization process for each language and verifies that certain changes or adjustments have a positive impact on the performance of the tool. The verification has also to be summative in the sense that at the end of the optimization process the overall performance for each language should be assessed.

In the rest of the section, we describe three tests which we have planned to evaluate the keyword extractor.

**Test 1** In order to have a rough idea of the performance of the tool, we have measured recall and precision of the keyword extractor for each language and each appropriate parameter setting. A gold standard has been established on the basis of manually annotated keywords (i.e. 1,000 keywords for each corpus of learning material). This part of the evaluation has been performed automatically.

**Test 2** In order to assess the difficulty of the task that the keyword extractor has to perform, that is how much variation there is among users in the assignment of keywords to a text, an evaluation of inter-annotator agreement (IAA) on the keyword selection task has been performed.

**Test 3** In order to assess the appropriateness of the keyword extractor in the context of the semi-automatic metadata annotation of the learning objects, we present test persons for each language with a document and a limited set set of keywords which have been extracted from this document. Each member of this set of keywords is assessed by the test person with respect to the adequacy to represent the text. This is an ongoing part of the evaluation for which we cannot report results yet.

### 3.1 Test 1: measuring performance of the keyword extractor

This evaluation of the keyword extractor is based on the keywords which have been selected and annotated manually. More specifically, for each language, at least 1,000 keywords have been manually selected and marked in the corpus of learning objects by one annotator. Table 2 gives additional information on how many documents were annotated per language, how many keywords were selected per language (= keyword types), average length of these documents and average number of keywords per document.

In this step of the evaluation, automatically extracted keywords have been matched with the manually selected ones. Thus, the manually selected keywords are used as gold standard. Recall and precision of the keyword extractor are measured against this standard in the following way:

- For each document $d_i$, let $WM = wm_1 \ldots wm_n$ be the set of manually selected keywords. Let $N$ be the number of these keywords. For each $i, j$, if $i \neq j$, then $wm_i \neq wm_j$ (i.e., there are no duplicates in the keywords lists).
- Let $WA = wa_1 \ldots wa_n$, be the keywords selected by the keyword extractor and $M$ the number of these keywords. We abridge these keyword lists such that $M = N$. For each $i, j$, if $i \neq j$, then $wa_i \neq wa_j$.
- Both $WM$ and $WA$ contain two subsets: $WMS$ and $WAS$, the subsets of single word keywords, and $WMW$ and $WAM$, the subsets of multi word keywords.
- For each element in $WMW$ and $WAM$, the length is calculated as the number of words. If $wm_k$ is a two word keyword, then $L_{wm_k} = 2$. For each single word keyword $wm_i$, $L_{wm_i} = 1$.

Recall and precision are calculated as follows:

- For each $i : 0 < i < M$, check whether $wa_i$ matches any $wm \in WM$. If this is the case and the match is exact, add a match value of one. All exact matches are summed up to a total value of $EMV$.
- If the match is partial, divide the length of the shorter keyword by the length of the longer keyword. If $wa_i$ partially matches $wm_k$ and

<table>
<thead>
<tr>
<th>language</th>
<th># annot. doc.</th>
<th># annot. KWs</th>
<th>tokens / doc.</th>
<th>KWs / doc.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bulgarian 55</td>
<td>3236</td>
<td>3980</td>
<td>77</td>
<td></td>
</tr>
<tr>
<td>Czech 365</td>
<td>1640</td>
<td>672</td>
<td>24</td>
<td></td>
</tr>
<tr>
<td>Dutch 72</td>
<td>1706</td>
<td>6912</td>
<td>26</td>
<td></td>
</tr>
<tr>
<td>English 36</td>
<td>1174</td>
<td>9707</td>
<td>39.5</td>
<td></td>
</tr>
<tr>
<td>German 34</td>
<td>1344</td>
<td>8201</td>
<td>41</td>
<td></td>
</tr>
<tr>
<td>Polish 25</td>
<td>1033</td>
<td>4432</td>
<td>34</td>
<td></td>
</tr>
<tr>
<td>Portuguese 29</td>
<td>997</td>
<td>8338</td>
<td>62</td>
<td></td>
</tr>
<tr>
<td>Romanian 41</td>
<td>2555</td>
<td>3375</td>
<td>3.5</td>
<td></td>
</tr>
</tbody>
</table>
Recall
Recall
Recall
Recall
Recall
Recall
Recall
Recall
0.15
0.15
F-Measure
0.16
0.23
0.26
0.11
Precision
0.18
0.22
Precision
0.18
0.42
0.33
0.22
Precision
0.10
0.21
0.24
0.33
0.13
0.11
0.12
F-Measure
0.26
0.34
0.26
0.39
0.32
F-Measure
0.19
0.18
F-Measure
0.15
F-Measure
0.25
0.21
0.15
0.27
0.42
0.19
0.17
0.18
0.29
0.19
0.18
0.14
0.23
0.36
0.15
0.48
0.17
Precision
0.29
0.17
0.14
Precision
0.25
Precision
0.19
0.21
0.19
0.15
0.18
0.15
0.36
0.31
0.24
0.38
0.12
0.47
0.16
F-Measure
0.18
0.25
Precision
F-Measure
0.28
F-Measure
0.30
0.25
0.22
0.26
0.12
This means that if multi-word keywords up to a length of 3 words were included. This is at least partially due to the fact that a higher proportion of multi word keywords increases the number of partial matches.

With respect to precision and recall, results varied significantly across languages. If we only consider TFIDF, the best result is 48 % for recall reached for English and the worst is 18 % for German, while with respect to precision, the best result is again obtained for English with 26 % while the worst is obtained for Romanian with 11 %. These values are influenced by two factors: a) the quality of the human judgment when selecting the keywords; b) the quality of the linguistic annotation of the corpora.

We also tested the impact of multiwords on results and we noticed that results improved for all languages if multi-word keywords up to a length of 3 words were included. This is at least partially due to the fact that a higher proportion of multi word keywords increases the number of partial matches.

As already mentioned, this test was performed only to get a rough impression of the performance of the keyword extractor as well as to determine which statistical measure performed best and to determine the maximum length for multiword keywords. More generally, its major purpose lies in informing the developers by presenting those keywords which did not match with the manually annotated ones and by presenting those manually selected keywords which have not been extracted by the tool. Note that not all keyword candidates which do not match manually selected keywords are necessarily bad keywords.

In fact, we believe that there might be some variation among users in identifying keywords and it is for this reason that we have performed an experiment to measure inter annotator agreement which is described in detail in the following section.

<table>
<thead>
<tr>
<th>Language</th>
<th>Method</th>
<th>Recall</th>
<th>Precision</th>
<th>F-Measure</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bulgarian</td>
<td>ADRIDF</td>
<td>0.38</td>
<td>0.18</td>
<td>0.25</td>
</tr>
<tr>
<td></td>
<td>RIDF</td>
<td>0.36</td>
<td>0.18</td>
<td>0.23</td>
</tr>
<tr>
<td></td>
<td>TFIDF</td>
<td>0.39</td>
<td>0.19</td>
<td>0.25</td>
</tr>
<tr>
<td>Czech</td>
<td>ADRIDF</td>
<td>0.22</td>
<td>0.17</td>
<td>0.18</td>
</tr>
<tr>
<td></td>
<td>RIDF</td>
<td>0.14</td>
<td>0.10</td>
<td>0.11</td>
</tr>
<tr>
<td></td>
<td>TFIDF</td>
<td>0.23</td>
<td>0.17</td>
<td>0.18</td>
</tr>
<tr>
<td>Dutch</td>
<td>ADRIDF</td>
<td>0.34</td>
<td>0.24</td>
<td>0.27</td>
</tr>
<tr>
<td></td>
<td>RIDF</td>
<td>0.25</td>
<td>0.19</td>
<td>0.21</td>
</tr>
<tr>
<td></td>
<td>TFIDF</td>
<td>0.36</td>
<td>0.25</td>
<td>0.29</td>
</tr>
<tr>
<td>English</td>
<td>ADRIDF</td>
<td>0.47</td>
<td>0.28</td>
<td>0.33</td>
</tr>
<tr>
<td></td>
<td>RIDF</td>
<td>0.33</td>
<td>0.18</td>
<td>0.22</td>
</tr>
<tr>
<td></td>
<td>TFIDF</td>
<td>0.48</td>
<td>0.26</td>
<td>0.32</td>
</tr>
<tr>
<td>German</td>
<td>ADRIDF</td>
<td>0.16</td>
<td>0.14</td>
<td>0.15</td>
</tr>
<tr>
<td></td>
<td>RIDF</td>
<td>0.15</td>
<td>0.12</td>
<td>0.13</td>
</tr>
<tr>
<td></td>
<td>TFIDF</td>
<td>0.18</td>
<td>0.15</td>
<td>0.16</td>
</tr>
<tr>
<td>Polish</td>
<td>ADRIDF</td>
<td>0.42</td>
<td>0.19</td>
<td>0.26</td>
</tr>
<tr>
<td></td>
<td>RIDF</td>
<td>0.29</td>
<td>0.15</td>
<td>0.19</td>
</tr>
<tr>
<td></td>
<td>TFIDF</td>
<td>0.42</td>
<td>0.19</td>
<td>0.25</td>
</tr>
<tr>
<td>Portuguese</td>
<td>ADRIDF</td>
<td>0.30</td>
<td>0.17</td>
<td>0.21</td>
</tr>
<tr>
<td></td>
<td>RIDF</td>
<td>0.21</td>
<td>0.12</td>
<td>0.15</td>
</tr>
<tr>
<td></td>
<td>TFIDF</td>
<td>0.31</td>
<td>0.18</td>
<td>0.22</td>
</tr>
<tr>
<td>Romanian</td>
<td>ADRIDF</td>
<td>0.26</td>
<td>0.12</td>
<td>0.15</td>
</tr>
<tr>
<td></td>
<td>RIDF</td>
<td>0.24</td>
<td>0.12</td>
<td>0.15</td>
</tr>
<tr>
<td></td>
<td>TFIDF</td>
<td>0.26</td>
<td>0.11</td>
<td>0.15</td>
</tr>
</tbody>
</table>

Table 3: Performance of the keyword extractor for the various languages

3.2 Test 2: Inter annotator agreement

In order to assess the intrinsic difficulties of the keyword selection task and to verify the performance of the keyword extractor compared to human annotators, we have investigated the inter annotator agreement on at least one document for each language. More specifically, we wanted to investigate where the performance of our keyword extractor stands relative to the performance of a group of human annotators. In the test described in the previous section, we have compared the output of the keyword extractor to the choices of
one single human annotator. We have thus relied completely on the performance of this individual annotator. The experiments which we describe in this section reveals how reliable the human judgement is and where the judgments of the keyword extractor stands relative to the human judgments.

A document of average size – around 10 pages – has been chosen for manual keyword selection. The content of the learning object was chosen so that it would be easy to understand for the test persons: it was a document dealing with Multimedia belonging to chapter 3, part 7 of the Calimera corpus. This material is available for all the languages under consideration.

A minimum of 12 test persons have been recruited for the experiment (with the exception of English and Czech, where we had only 9 and 6 judgments). In the instructions, which have been written in English and have been translated into the eight languages we have tested, the annotators were asked to select not more than 15 keywords and to mark for each keyword how sure they were that this is a good keyword. A scale was given from 1 (very sure) to 3 (not so sure).

For German and Romanian, two experiments were performed. For German, the same text was given to two groups, a group of students who were not familiar with the topic and a group of experienced scientists. We wanted to investigate whether experienced scientists achieve a higher inter-annotator agreement than students who are not familiar with the topic. The Romanian group ran the experiment with different texts to check whether characteristics of the text influence inter annotator agreement.

We used the approach of [2] to calculate the inter-annotator agreement for this task. This means that we have modeled it in a way that for every token in a text it is recorded whether an annotator decided that this word is a keyword or not. Let $A = a_1 \ldots a_A$, where $A$ is the number of annotators. Let $D$ be the Document to be annotated and $T_D$ be the number of tokens in the document. For any two annotators $a_i$ and $a_j$, where $a_i, a_j \in A$ and $i \neq j$, and a document $D$, we record the following values:

- $t_i \land t_j$, the number of words chosen by both annotators
- $t_i \land \neg t_j$, the number of words chosen by the first annotator, but not by the second one
- $\neg t_i \land t_j$, the number of words chosen by the second annotator but not by the first one
- $\neg t_i \land \neg t_j$, the number of words chosen by neither annotator

It follows that $(t_i \land t_j) + (t_i \land \neg t_j) + (\neg t_i \land t_j) + (\neg t_i \land \neg t_j) = T_D$.

Let $n_{11} = t_i \land t_j$, $n_{12} = t_i \land \neg t_j$, $n_{21} = \neg t_i \land t_j$, $n_{22} = \neg t_i \land \neg t_j$ the observed values. These values can be filled in a contingency table from which marginal sums can be computed.

From this contingency table, the following formula for inter-annotator agreement can be derived, following the approach of Bruce and Wiebe:

$$\kappa = \frac{\sum n_{ij} - \sum n_{ii} + n_{jj}}{1 - \sum n_{ii} + n_{jj}}$$

where $\kappa$ is 1 if the agreement is perfect, and zero if the agreement is that expected by chance. Negative values are possible by if the agreement is lower than that expected by chance.

We proceed in the following way. For each language, we calculated for each pair of annotators, the agreement value of these two annotators measured by the kappa statistics.

For each annotator $a_i$, we furthermore calculated the average agreement of this annotator with the other annotators as:

$$\text{avg kappa}_{a_i} = \frac{\sum j \kappa_{a_i, a_j}}{A - 1}$$

In a follow up experiment, we extracted keywords from the same text with our keyword extractor. We used the three different statistics – RIDF, ADRIDF and TFIDF – which we have described above. We selected the $n$ highest ranked keywords, where $n$ is closest to the average number of keywords selected by the human annotators, and measured the inter annotator agreement of these three machine agents with the human annotators.

Table 5: Results of inter annotator agreement experiments

<table>
<thead>
<tr>
<th>Language</th>
<th>Human</th>
<th>TFIDF</th>
<th>RIDF</th>
<th>ADRIDF</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bulgarian</td>
<td>0.10</td>
<td>0.25</td>
<td>0.28</td>
<td>0.37</td>
</tr>
<tr>
<td>Czech</td>
<td>0.23</td>
<td>0.33</td>
<td>0.28</td>
<td>0.39</td>
</tr>
<tr>
<td>Dutch</td>
<td>0.16</td>
<td>0.22</td>
<td>0.14</td>
<td>0.28</td>
</tr>
<tr>
<td>English</td>
<td>0.09</td>
<td>0.29</td>
<td>0.29</td>
<td>0.43</td>
</tr>
<tr>
<td>German</td>
<td>0.25</td>
<td>0.23</td>
<td>0.23</td>
<td>0.23</td>
</tr>
<tr>
<td>Polish</td>
<td>0.28</td>
<td>0.15</td>
<td>0.15</td>
<td>0.20</td>
</tr>
<tr>
<td>Portuguese</td>
<td>0.18</td>
<td>0.16</td>
<td>0.15</td>
<td>0.19</td>
</tr>
<tr>
<td>Romanian</td>
<td>0.20</td>
<td>0.24</td>
<td>0.22</td>
<td>0.26</td>
</tr>
</tbody>
</table>

The results are reported in table 5. From this experiment, we can conclude that inter annotator agreement is not very high therefore this is in general not a task on which humans easily agree. What is relevant for us though, is that the keywords extractor performs for most languages better than humans. Furthermore, on the basis of the results obtained for the additional experiment performed for Romanian and German, we can conclude that experts do not seem to behave more consistently than non-experts.

The experiments might also provide useful feedback which might help improve the performance of the keyword extractor. It might be relevant to look at

---

3 http://www.calimera.org/
the words on which the majority of human annotators agree and to check why they are not captured by the key word extractor and adapt the keyword extractor if this is possible.

Another interesting finding is that, while in test 1 TFIDF performed slightly better than ADRIDF, in test 2 ADRIDF produced lists which resulted in a better agreement with the human annotators. First, the results cannot be compared directly, because in test 1 we refer to only one human annotator and a collection of documents, while in test 2 we refer to a group of annotators and only one document. Since ADRIDF did not perform much worse than TFIDF in test 1, we are inclined to favour ADRIDF.

4 Integration into ILIAS

The keyword extractor is a functionality which has been integrated in a learning management system to support the semi-automatic metadata annotation of the learning objects. It should assist authors of learning material to find and assign appropriate keywords to their learning objects. In the context of the LT4eL project, the tool has been integrated in the ILIAS learning management system even though it should be possible to enhance other LMS with it.

The tools and data reside on a dedicated server and are approached from inside the Learning Management System via Web Services. Figure 2 shows the major components of the integration setup. The language technology server on the left provides the keyword extractor and other NLP components (cf. [9] for more details). The functionalities can be accessed directly on the webserver for test purposes or they can be used by the learning management system through the web service interface. Figure 3 shows the first integration of the keyword extractor into the ILIAS learning management system. The function is embedded into the existing LOM metadata handling of ILIAS to enable a semi-automatic generation of keywords. Users can:

a) run the keyword extractor and get a list showing a number of suggested keywords for a document; b) select keywords from this list and c) add their own keywords. The interactivity is an important feature of this tool. It will not be used to completely perform the task of keywording a documents, but to make improved suggestions which the user has to approve or reject.

5 Conclusions and future work

One of the functionalities developed within the LT4eL project is the possibility to annotate learning objects semi-automatically with keywords that describe them, to this end a keyword extractor has been created. The approach employed is based on a linguistic processing step which is followed by a filtering step and keyword ranking based on frequency criteria.

Two tests have been carried out to provide a rough evaluation of the performance of the tool and to measure inter annotator agreement in order to determine the complexity of the task and to evaluate the performance of the keyword extractor with respect to human annotators.

The results are promising also considering that the task has been carried out for 8 different languages. However, there are possible ways in which the results of the keyword extractor could be improved.

A further distributional characteristic of keywords is what has been called their burstiness (cf. [8]). Good keywords tend to appear in clusters within documents. It can be observed that, once a word appeared in a text, it tends to appear in the following section of the text in much shorter intervals as would be the case if all the occurrences of this word were distributed evenly throughout the text. When the burst ends, the occurrence intervals return to normal, until the keyword appears - and probably bursts - again. A method to capture this behaviour of key words is described in [12]. This distributional behaviour reflects the fact that keywords represent topics, and reference to a certain topic is a local phenomenon: many texts deal with many (sub)topics and in some texts a topic is resumed after a while. We are currently investigating the im-
impact of term burstiness on the extraction of keyword sets on a subset of the languages under consideration.

Keyword candidates tend to appear in certain salient regions of a text. These are the headings and the first paragraphs after the headings as well as an abstract or summary. Salient terms might also be highlighted or emphasised by the author, e.g. by using italics. Investigations of the manually annotated keywords have shown that a word with a salient position or marked by layout features is at average twice as probable to be marked as keyword as those words which do not bear these features. Therefore, we plan to use layout information as additional filter in proposing salient keywords (cf. [13]).

Another approach to improve the extraction of appropriate keywords is to look at words which are related syntactically and/or semantically. Sets of related lexical units represent the topic(s) of a text better than individual words can do. In general, there are two methods to build structures of related words from text. First, one can relate words which show a higher probability of co-occurrence that could be expected if words would be distributed randomly. Second, lexical-semantic networks can be used to find pairs of words or concepts which are linked by lexical-semantic relations. We envisage to extract lexical chains from texts (cf. [11], [1]), using wordnets for those language in the project for which they are available. Lexical chains are rich structures of semantically related words. Keywords are assumed to participate in long and/or dense chains.

References


Crosslingual Ontology-Based Document Retrieval

Eelco Mossel
University of Hamburg
Department of Informatics
Vogt-Kölln-Str. 30
D-22527 Hamburg, Germany
mossel@informatik.uni-hamburg.de

Abstract

An approach for crosslingual ontology-based document retrieval has been devised and is being implemented. It allows the user to enter a query in any language that is part of the system and retrieve documents in selected languages. A domain ontology and term-concept lexicons, containing synonymous terms where applicable, are used to overcome discrepancies between the search query and the words occurring in the documents, in a monolingual situation for the individual languages as well as in a crosslingual setting.

The ontology is used in two different ways. First, concepts relevant for a search query are found automatically and used to retrieve documents. Second, relevant parts of the ontology are displayed to the user, who can navigate further starting from the displayed part of the ontology, and explicitly select concepts to continue the search with.

Keywords

multilinguality, document retrieval, search, ontologies, eLearning

1. Introduction

The most well-known kind of document retrieval is based on full-text indexing, and the retrieval results heavily depend on the correct choice of words and word forms in the query. With the increasing availability of digital documents, also in languages other than English, it is important to have more targeted means to find the wanted document from an available collection. This applies particularly to the multilingual case, where the user would have to translate his search query into all the languages he wants to retrieve documents in, which requires an active command of the involved languages and in particular knowledge about the habitual terms in the respective domain.

In this paper, I describe a setup for using domain ontologies to facilitate crosslingual search in an available document collection. The approach was developed in the framework of the BIS-21++ project1 and implemented within the project LT4eL (Language Technology for eLearning)2. The design fits closely to the resources and needs of LT4eL, but is in principle applicable to other situations with similar preconditions.

The project LT4eL aims at improving the retrieval and usability of learning materials in multiple languages. Among other things, learning materials are semi-automatically annotated with keywords and concepts, and a domain ontology has been created.

The relevance of crosslingual retrieval in eLearning lies in the fact, that learning materials might be available in a different language than the student’s native language. Instead of having to come up with a good search query in foreign languages, the user will be presented immediately with relevant documents in languages he knows.

In the framework of LT4eL, the search functionality described in this paper is being integrated in the Learning Management System ILIAS3.

Section 2 describes the general idea for the crosslingual ontology-based search, the underlying assumptions and a user scenario. Based on all this, the search functionality was developed, as described in Section 3. Section 4 explains how the approach will be integrated and applied in the project LT4eL. Section 5 contains the conclusion and information about future work.

2. Basis for the document retrieval

The goal of the approach is a search functionality with the following three characteristics:

1. Improved access to documents. The search mechanism must exploit semantic characteristics of search queries and documents, and be able to find relevant documents that would not be found by a simple full-text search4.

2. Multilinguality. With one implementation, the approach should work for multiple languages.

---

1 Bulgarian IST Centre of Competence in 21 Century (BIS-21++,
http://bis-21pp.acad.bg/); one-month research stay of Eelco Mossel at the Bulgarian Academy of Sciences, Institute for Parallel Processing, Sofia.

2 www.lt4el.eu - The LT4eL project is supported by the European Community under the Information Society and Media Directorate, Learning and Cultural Heritage Unit.

3 www.ilias.de

4 Or rather, the relevant documents must be found in a more efficient way than having to try numerous full-text searches, hoping to capture the words that occur in a relevant document.
3. Crosslinguality. It should be possible to retrieve documents in languages other than the language that is used for representing the ontology or formulating/selecting terms.

2.1 Assumptions
The starting point for designing an ontology-based search functionality have been the following assumptions, which hold for LT4eL and are realised as described in [2].

- There is a multilingual document collection (this is not a requirement, but with documents in only one language, the full potential of the approach will not be exploited).
- There is a (language independent) ontology that includes a domain ontology on the domain of the documents. There may be more than one domain ontology. A domain ontology consists of a set of concepts, belonging to the same domain, and various kinds of relations between the concepts are possible. Between two separate domain ontologies, there are no relations, because they have been created independently. But as they can all be connected to the same upper ontology and become a part of one large ontology in this way, it is not necessary to iterate over a list of ontologies – the descriptions in this paper always just refer to “the ontology”.
- For each of the addressed languages, there is a lexicon with words or phrases that are mapped to concepts of the ontology.
- The concepts have a name and/or description in each of the addressed languages, which can be used for presenting them in the user interface.
- The documents are annotated with concepts. In the simplest case, for each document there is a set of concepts that are relevant for the contents of the document. In LT4eL, more detailed information is available: the annotation of a concept is attached to every place in the document where this concept is mentioned. It can thus be derived, how often each concept occurs. If this information and also the length of the document are available, they are used for ranking the found documents.
- For each document, its language is available (requirement).
- It is known, which languages the user can use to formulate his query (e.g. by means of a user profile). This is used to determine which lexicons to use; however, if it is not known, all available lexicons could be consulted by default.
- It is known, in which languages the user wants to retrieve documents (if this is not known, all relevant documents in any available language could be shown by default).

Having those assumptions in mind, I proceeded to define a search scenario (see Section 2.2) from the point of view of the user, showing what the user will (have to) do and what he will see. On the basis of this scenario, the search functionality (see Section 3) was specified.

The basic idea of the ontology-based search (or semantic search) is, that concepts from the ontology lead to the documents to retrieve. The search will probably work best, when the user selects exactly the concept(s) from the ontology that he wants as a topic for the retrieved documents. However, there are two reasons to start with a free-text query by the user.

First, the semantic search will “compete” with other search strategies such as full-text search (all words from the text are considered when looking for a match for the query) and keyword search (matching with words that are annotated as keywords in the documents – to avoid confusion, the words the user types in are called search words and not keywords). The user, who is used to Google, wants his results fast, with not too many intermediate steps where he has to choose things. Therefore, we would like to invoke semantic search as soon as the user has entered his search words, and give first results simultaneously with the other search methods.

Second, it is good to give the user a starting point for finding the desired concept in the ontology, so that he does not have to start at the root of the ontology. The search words are used to find the starting point in the ontology. In a second step, the search can be refined by selecting concepts from the ontology. Because of this approach, two different ways of semantic search occur in the search scenario.

2.2 Search scenario

1. **Submit query**
   - User enters free text and submits this as a query.

2. **See document list**
   - A list of retrieved documents is displayed with some meta information, for example:
     - title;
     - length;
     - original language;
     - keywords and concepts that are common to both the query and the document;
     - other keywords and concepts that are related to the document but not to the query.

3. **See concept browsing units**
   - Each concept that is assumed to be related to the search query, is presented to the user together with its neighbourhood from the ontology (related concepts and relations between the concepts). Call such a displayed part of the ontology a “browsing unit” (for an example see Figure 1).
   - If no concept related to the search query is found, the root of the ontology with its neighbourhood is chosen as the browsing unit. The user can browse the ontology: by clicking on a concept or a related button, the user will see related
concept. If there are many relations, a possibility is, to let the user select the kind of relation he would like to use to explore the ontology (e.g. “show only concepts that are related to ApplicationProgram by a part-of relation”).

![Application Program](image)

**Figure 1** – Example of a possible representation of a browsing unit, where only taxonomical relations are present.

4. **View documents**
   User looks into the documents from the list.

5. **Browse ontology**
   User browses the ontology: starting from the presented concepts, he can proceed to related concepts, and concepts that are related to those again, etc.

![Search interface](image)

**Figure 2** – User interface for search in ILIAS. In the upper part, search words can be entered and languages can be selected. In the lower left part, the resulting documents are listed. In the right part, concepts can be selected.

6. **Select concepts**
   User selects a set of concepts ontology fragments (sets of related concepts, possibly only indirectly related) from the presented browsing units.

7. **Select search option**
   User selects an option about how to use the ontology fragments for search. Options might include “disjunctive” (find documents, in which any of the selected concepts occur) and “conjunctive” (find documents, in which all of the selected concepts occur).

8. **See new document list**
   A new list of documents is displayed, based on ontological search. If also new search terms have been specified, the documents based on the selected concepts come first, followed by the ones that are found only by the new search terms and not by the selected concepts.

9. **See browsing units including shared concepts**
   As in step 3, but now, also shared concepts are presented: concepts, that are common to more than N of the found documents; this includes the concepts that were used as the search key but might include further concepts. The number of documents, specified by the threshold parameter N, can be relative (a percentage of the number of found documents) or absolute.

10. **Repeat steps from step 6 (Select concepts)**
    User selects another set of related concepts and submits it as the search key, etc.

The search scenario is now realised in LT4eL by integrating the functionality into the Learning Management System ILIAS. Figure 2 shows the user interface.

3. **Design of the search functionality**
   The search functionality as a whole has as input parameters:
   - Possible languages of search query (determines which lexicons to use for lookup)
   - Retrieval languages (find documents in those languages)
   - Search terms, entered by user
   - Concepts, selected by user
   - Method for combining the concepts (conjunctive, disjunctive)
   - Option indicating whether to also retrieve documents that do not contain the desired concept, but do contain a superconcept or a subconcept of it

   and as output:
   - the ID of the found documents
   - the initial contents for the browsing units (parts of the ontology, see further Section 3.5), including shared concepts

To support the search scenario outlined in Section 2.2, the search function has to incorporate the following steps.

1. Find terms in the term-concept-lexicons that match the search query.
2. Find corresponding concepts for derived terms.
3. Find relevant documents for concepts.
4. Create a ranking for a set of found documents.
5. Support displaying and selecting concepts from the ontology.
In the subsections 3.1 through 3.6, each of those points is explicated.

### 3.1 From query to term-concept mappings

A few issues have to be regarded when trying to transform a query to terms that could be found in a term-concept lexicon, namely:

- Uppercase/lowercase differences
- For languages other than English, words that officially contain letters with diacritics might be entered in the search query using simple characters without diacritics, depending on the keyboard layout and settings.
- Dashes, spaces and apostrophes might be added or omitted by the user.

A thorough way to find all terms that are relevant for an entered search query is the following.

1. Tokenise the query.
2. Create combinations for multi-word terms. Not only the individual words could refer to concepts, but also sequences of words. Depending on the language, they can be compound words, such as “school year”, or expressions and collocations that might denote a concept, such as “high-speed connection”. There are several ways to form compounds. Besides treating them as a whole with just a space in between, also concatenating them with a dash or with nothing in between is possible.

Currently, the lexicon lookup is just simple string matching, so all the possible multi-word terms have to be generated from the query and looked up one by one. The basis for the combination are all the subsequences of the query; for example, for query “a b c d”, the lists of terms to be combined would be `[a,b] [b,c] [c,d] [a,b,c] [b,c,d] [a,b,c,d].

3. Find a normalised form for each term, as not all inflected forms are included in the lexicons. The LT4eL lexicons mainly contain lemmas. The best recall will be achieved, if for every addressed language, a lemmatiser is used. Furthermore, depending on the matching mechanism, normalisation can comprise replacement of diacritics and uppercase letters by their simple, lowercase variants.

4. The set of strings for lookup consists of the original terms, the normalised terms and the created multi-word terms.

The current prototype supports the creation of multi-word terms, but no lemmatisation.

Instead of adding lemmatisation, the following alternative approaches can be considered:

- Expand the term lexicon, by generating word forms.
- Allow partial matches: take lexicon terms into account, that only match part of a token of the query, and the other way around: lexicon terms of which only a part matches a token from the query. Or even more flexible: take into account such partial matches, where a token of the query and a lexicon term have a part in common, but both have an unmatched part. Obviously, this solution has disadvantages. It can result in false matches, while at the same time it does not work for an inflected word form whose stem is different from the stem of the corresponding lemma. And decisions will have to be taken on how large the overlapping part must be. Perhaps this parameter should even be chosen language-independently.

The partial-matching solution would not only work for single words: it could also be used instead of generating possible multi-word terms from the query.

Related to the mechanism of automatic partial matching, an interactive way of term selection can be thought of: the user will see a dynamically changing list of all available terms that contain the entered characters as a substring. Every time a character is added or deleted, the list is updated to reflect the change. When this works fast, it is a convenient way of selecting terms. However, it is conflicting with the idea of just typing a free-text query, which can be used for non-ontological search at the same time.

### 3.2 From terms to concepts

When trying to find a concept for a certain term, in principle, the following situations can occur:

- The corresponding concept is missing from ontology.
- Unique result: the term is a lexicalisation of exactly one concept.
- The term denotes multiple concepts from one domain, for example:
  - Key (from keyboard).
  - Key (in database).
- The term denotes concepts from more domains, for example:
  - Window (graphical representation on monitor).
  - Window (part of a building).
- The term denotes different concepts for different languages:
  - “Kind” (English: sort/type)
  - “Kind” (German: child)

By presenting all the possibilities to the user and let him choose, the ambiguities do not have to be resolved automatically.

### 3.3 From concepts to documents

Given the annotation of concepts in the documents, it is a trivial task to find the documents dealing with a certain

---

5 This mechanism is used in electronic dictionaries, with the restriction, that substrings are matched only from the beginning of the word.
concept. Two non-trivial issues, however, are the following.

First, what further role can the ontology play in finding relevant documents? The concepts are not just labels for the documents; they have their place in the used ontology. Intuitively, a document dealing with a subtopic of the desired topic (thus containing a subconcept of the central concept) partially satisfies the information need of the user, and should therefore be included in the result. This brings us to the question, whether this also holds for the subconcept of this subconcept. And it might also apply for other kinds of relations, such as part-of. But every step down in the taxonomy changes the level of detail, and every step following a different relation might take us further away from the desired topic (e.g. a computer has a processor, a processor is made of silicon, silicon is produced in Australia, …)

In order to make use of the relations in the ontology and yet avoid very unpredictable results, we employ the following strategy. The user can select for each available kind of relation (currently: superconcept and subconcept) whether he wants to include the concepts reached by this kind of relation in the search. Two restrictions are used: only related concepts are used that are one step away in the ontology, and this is done only, when a search concept occurs in less than N documents (determined by a threshold parameter). In addition, however, the user can select an option to include the related concepts in any case, independent of the threshold.

The usefulness of this kind of query expansion also depends on the quality and granularity of the ontology. In the current LT4eL ontology on the domain of computer science for non-experts, for example, “Computervirus” is a subconcept of “Program”, but most people will probably not be interested in computer viruses when searching for “program”. On the other hand, adding results for “web portal” (a subconcept of “web site”) to the results for “web site” is very useful.

The second issue concerns how to deal with the selection of more than one concept from the ontology. Two obvious possibilities, which are also easy to understand for the user and do not involve explicit specification of boolean operators, are conjunctive search (find only documents in which all of the selected concepts occur) and disjunctive search (find documents in which any of the selected concepts occur). However, when using the ranking criterion that is described in Section 3.4, the top-ranked results of the disjunctive search will be exactly the same as the results for conjunctive search.

Baumann et al. [1] use an ontology for automatic query expansion. They state a relationship between the issues of related concepts and disjunctive/conjunctive search in the following way: "we are generating boolean queries on the fly based on the assumption that part-of edges can be interpreted as every sub-concept should be contained in one document, while edges of the type is-a allow for alternatives.” This is in line with our approach where a document is returned if a subconcept occurs instead of the original concept; with other relations, we cannot experiment yet in LT4eL (see also Section 4).

3.4 Ranking
As a measure to rank the retrieved documents, we use the following two ranking criteria:

1. The number of different concepts that are used for searching and occur in the document. The intuition is, that a document serves the search query better, if a larger part of the search query is matched.

   Example:
   - User enters terms create and folder.
   - create denotes concept CREATE
   - folder denotes concept DIRECTORY

   So the search is done with concepts CREATE and DIRECTORY.
   - Document 1 deals with concepts CREATE, DIRECTORY and LINUX.
   - Document 2 deals with concepts FILE, DIRECTORY, LINUX and SYMBOLICLINK.
   - Document 1 contains two concepts that were used for searching, so it is ranked higher than document 2, which only has DIRECTORY in common with the search query.

2. Normalised annotation frequency: the number of times that the looked up concepts are annotated in the documents, divided by the length of the document. This criterion is applied to documents that got the same ranking by the first criterion. The normalisation (dividing by the document length) is done because otherwise, long documents would be favoured, as they are more likely to mention the concept more often. However, if the occurrences are concentrated only in certain parts of the document, the document as a whole will have a low score because of the division by a relatively large length, even though it contains very relevant parts. This low score can be justified by the fact that indeed not the whole document is very relevant. Another option could be to take not the length of the whole document but the length of the section in which the concept occurs several times. However, this makes things more complicated than it seems at first sight: if the concept happens to occur also outside of the relevant part, the entire (much larger) distance between the first and this last occurrence would be taken as divisor, so the score will be penalised just because of the additional occurrence. If a subconcept or superconcept is counted instead of the original concept, its will get a lower weight (determined by
This holds for both of these ranking criteria.

This is a relatively easy ranking mechanism; experiments will show whether it is satisfying. Vallet et al. [4], who describe a similar approach for semantic search, although not multilingual, use a more advanced ranking algorithm, based on similarity vectors between queries and documents.

3.5 Browsing Units and Ontology Fragments

For the second phase of the search (searching documents by selecting concepts), ontology fragments (parts of the ontology) will be displayed to the user as browsing units (cf. point 3 in Section 2.2).

For every concept that is found through the search terms, the neighbourhood of the concept is looked up in the ontology, and the result is passed to the search function as an ontology fragment. In order to not display the same part of the ontology twice, overlapping fragments are optionally merged, and displayed within one browsing unit. For example, if one fragment contains “A has subclass B”, and another contains “A has subclass C”, then the merged fragment will be displayed as A having subclasses B and C.

From the architecture perspective, the browsing units are on the border between the search functionality and the graphical user interface. The search function determines which of the concepts are displayed initially, but the GUI determines, besides from the fact how they are displayed, what will be presented when navigating through the browsing units.

3.6 Shared concepts

As described in step 9 of the user scenario, a set of concepts that are common to a certain part of the found documents is calculated. Ideally, this includes concepts that were not found by means of the search terms, and it can guide the user to other documents he is interested in but could not figure out the direct way to them. For example, the user enters terms leading to the concept denoted by “Report”. Some documents on academic writing are found, and they share the concept “Publication”.

For the first experiments, we will use a threshold of 50%: the set of shared concepts includes each concepts, that occurs in at least 50% of the found documents (the 50% of the documents can be a different subset for each shared concept).

The idea of using the contents of retrieved documents to adapt/expand a search query and find additional relevant documents is applied by Song et al. [3] on the level of terms rather than concepts, and in an automatic way.

4. Architecture and use within LT4eL

In the project LT4eL, learning objects (LOs) have been collected in eight languages: Bulgarian, Czech, Dutch, English, German, Polish, Portuguese, Romanian. The size of the collection varies from around 30 to 80 documents per language, with a total of approximately 200,000 words per language.

One domain, that is covered by LOs in all of those languages can be called “computer or information science for non-experts”. Based on the contents of those LOs, a domain ontology for this domain was created. It contains 700 concepts and only taxonomic relations.

Furthermore, for the languages mentioned above plus Maltese, term lexicons are being generated, containing lexicalisations for all concepts of the ontology. The lexicons contain a few synonymous terms: terms in the same language, that are mapped to the same concept. Some of the terms consist of multiple words. The occurrences of terms that denote a concept from the ontology have been semi-automatically annotated in the documents.

The lower part of Figure 3 gives a schematic view of the relation between terms, ontology and LOs.

The search functionality, as described in Section 3, uses the following functions during runtime (not necessarily in this order – the following numbers 1 through 8 are referred to in Figure 3, while the order of steps is given in the beginning of Section 3):

1. For a specified term and a specified language, find all the concepts denoted by this term in this language.
2. Find the superconcepts for a specified concept.
3. Find the subconcepts for a specified concept.
4. For a specified concept, find all the concepts that are reachable by at most one step by following any relational edge in the ontology.
5. Find all the documents in a certain language that contain the specified concept(s).
6. For a specified document, return all the concepts that occur in it.
7. For a specified document and a specified concept, return the number of times the concept occurs in the document (annotation frequency).
8. Return the length of a document (used for normalisation).

These functions are provided by three software components, developed by IPP-BAS:

- Lexicon Tool (LEX).
- Ontology Management System (OMS). It allows for more complex reasoning than is currently used by the search functionality.

---

6 Institute for Parallel Processing at the Bulgarian Academy of Sciences, Sofia.
Ontology-based Search Engine (OSE). It is used for the functions 5 through 8, but also provides more advanced functionality, such as finding documents in which several concepts occur within the same paragraph.

Figure 3 gives an overview of the architecture. The search functionality is being integrated into the Learning Management System ILIAS, in which the collected LOs are stored.

5. Conclusion and further work

An approach for crosslingual ontology-based document retrieval has been devised and is being implemented. It allows the user to enter a query in any language that is part of the system and retrieve documents in selected languages. A domain ontology and term-concept lexicons, containing synonymous terms where applicable, are used to overcome discrepancies between the search query and the words occurring in the documents, in a monolingual situation for the individual languages as well as in a crosslingual setting.

The ontology is used in two different ways. First, concepts relevant for a search query are found automatically and used to retrieve documents. Second, relevant parts of the ontology are displayed to the user, who can navigate further starting from the displayed part of the ontology, and explicitly select concepts to continue the search with.

Several decisions had to be taken about the chain from search query through terms and concepts to documents, about ranking and query expansion. The chosen solutions and some alternatives were discussed in this paper.

The search functionality is being integrated and tested in the Learning Management System ILIAS in the framework of the project LT4eL, which provides a multilingual collection of learning materials that are annotated with concepts, a domain ontology that covers the domain of a substantial part of the learning materials, and ontology, and a term lexicon with mappings to the concepts of the ontology.

Experiments, to be carried out in the near future, have to show whether the provided options are useful, whether the relevant documents are easily found, and whether the ranking mechanism is satisfying.

For the monolingual case, we will evaluate in terms of recall/precision, whether documents relevant to a certain query are better found by the semantic search than by full-text search.

For the multilingual case, scenarios involving bilingual or multilingual test persons will be set up, to show the added value of crosslingual search in a learning management system. Learning material that is useful for a certain task will have to be found from the available collection; a collection that covers the topic better can be obtained by selecting more than one retrieval language.

References


On the evaluation of Polish definition extraction grammars

Adam Przepiórkowski, Łukasz Degórski, Beata Wójtowicz

Polish Academy of Sciences, Institute of Computer Science
ul. Ordona 21, 01-237 Warsaw, Poland
adamp@ipipan.waw.pl, {ldegorski,beataw}@bach.ipipan.waw.pl

Abstract
This paper presents the results of experiments in the automatic extraction of definitions (for semi-automatic glossary construction) from usually unstructured or only weakly structured e-learning texts in Polish. The extraction is performed by regular grammars over XML-encoded morphosyntactically-annotated documents. The results, although perhaps still not fully satisfactory, are carefully evaluated and compared to the inter-annotator agreement; they clearly improve on previous definition extraction attempts for Polish.

1. Introduction
The aim of this paper is to report on experiments in the automatic extraction of those fragments of Polish text which could be used as definitions of terms (whether they were written as definitions or not) and to discuss possible ways of evaluating the results of such experiments.

The context of the work reported here is a European (IST) Specific Targeted Research Project aiming at constructing various language technology tools for eLearning, for a number of languages, including Polish.\textsuperscript{1} The input to the definition extraction module is XML-encoded\textsuperscript{2} morphosyntactically-annotated text, so the definition extraction module must be XML-aware and the effect of the operation of the module should consist in adding XML elements marking the defined term (the \textit{definiendum}) and the defining text (the \textit{definiens}). Such automatically extracted term definitions are to be presented to the author or the maintainer of the Learning Object (LO; i.e., course materials) and, thus, significantly facilitate and accelerate the creation of a glossary for a given LO. From this specification of the task it follows that good recall is much more important than good precision, as it is easier to reject wrong glossary candidates than to browse the LO for term definitions which were not automatically spotted.

The structure of the paper is as follows: §2. mentions previous work on definition extraction, §3. presents a shallow grammar developed for definition extraction from Polish texts, §4. describes the evaluation experiments and their results, §5. compares these results to the state of the art, §6. discusses the inherent difficulty of the task, and §7. concludes the paper.

2. Related Work
Definition extraction is an important NLP task, most frequently a subtask of terminology extraction (Pearson, 1996), the automatic creation of glossaries (Klavans and Muresan, 2000; Westerhout and Monachesi, 2007), question answering (Miliaraki and Androustopoulos, 2004; Fahmi and Bouma, 2006), learning lexical semantic relations (Malaisé et al., 2004; Storrer and Wellinghoff, 2006) and automatic construction of ontologies (Walter and Pinkal, 2006). Tools for definition extraction are invariably language-specific and involve shallow or deep processing, with most work done for English (Pearson, 1996; Klavans and Muresan, 2000; Saggion, 2004) and other Germanic languages (Fahmi and Bouma, 2006; Storrer and Wellinghoff, 2006; Walter and Pinkal, 2006; Westerhout and Monachesi, 2007), as well as French (Malaisé et al., 2004). (Some comparison to this earlier work is made in §5.)

There is very little previous work on definition extraction in Slavic languages, with the exception of some work on Bulgarian reported in Tanev 2004 and Simov and Osenova 2005, and a recent article on Bulgarian, Czech and Polish (Przepiórkowski et al., 2007). In all cases shallow grammars were constructed for the identification of definitions in texts, but only Przepiórkowski et al. 2007 contains the evaluation of these grammars in terms of precision, recall and \textit{F\textsubscript{2}} measure.\textsuperscript{3} Our results improve on the results for the best grammar presented in Przepiórkowski et al. 2007, namely, the grammar for Czech, and far exceed their results for Polish.

3. Regular XML Grammars of Definitions
The complete corpus of instruction texts with manually annotated definitions was split into three parts: a training corpus, a held-out corpus and a testing corpus. The training corpus and the testing corpus consist of 12 different instruction texts each, so they are reasonably well balanced. The held-out corpus, on the other hand, is a specific homogeneous text, namely, the Calimera guidelines (http://www.calimera.org/). The quantitative characteristics of these corpora, in particular the number of manually annotated definitions, including the number of definitions split across two or (in one case) three sentences, is given in Table 1. The first version of the grammar, GR, was developed on the basis of the training corpus, in many (well over 100) iterations, where in each iteration the grammar was improved and the results were evaluated both quantitatively (automatically) and qualitatively (manually). This grammar was then applied to held-out data, and subsequently modified, in relatively few iterations (around 5), to increase precision and recall measured

\begin{table}
\begin{tabular}
\midx{3}{5}{|c|c|c|c|c|}
\hline
Corpus & Training & Held Out & Testing & Deviation \%
\hline
\text{Manually Annotated Definitions} & 256 & 14 & 16 & 10.1
\hline
\text{Manually Annotated Definitions} & 256 & 14 & 16 & 10.1
\hline
\end{tabular}
\caption{Quantitative Characteristics of the Corpora}
\end{table}

\textsuperscript{1}Details of the project are withheld for reasons of anonymity; they will be provided in the final version of the paper.
\textsuperscript{2}More precisely, the input adheres to the XML Corpus Encoding Standard (Ide et al., 2000).
\textsuperscript{3}\textit{F\textsubscript{2}} = (1 + \alpha) \cdot (\text{precision} \cdot \text{recall})/(\alpha \cdot \text{precision} + \text{recall}).
The input to the current task of definition extraction is XML-encoded morphosyntactically-annotated\textsuperscript{5} text adhering to the XML Corpus Annotation Standard (XCES; Ide et al. 2000). For example, the representation of a Polish sentence starting as \textit{Konstruktywizm kładzie nacisk na} (Eng. ‘Constructivism places the emphasis on’) may look as follows:\textsuperscript{6}

\begin{verbatim}
<tok base="konstruktywizm" msd="sg:nom:m3" ctag="subst" id="t3">Konstruktywizm</tok>
<tok base="kłaśc" msd="sg:ter:imperf" ctag="fin" id="t4">kładzie</tok>
<tok base="nacisk" msd="sg:acc:m3" ctag="prep" id="t5">nacisk</tok>
<tok base="na" msd="acc" ctag="interp" id="t6">na</tok>
...
<tok base="." ctag="interp" id="t17">
</tok>
\end{verbatim}

The grammar is a regular grammar implemented with the use of the \texttt{lxtransduce} tool (Tobin, 2005), a component of the LTXML2 toolset developed at the University of Edinburgh. An example of a simple rule for prepositional phrases (PPs) is given below:

\begin{verbatim}
<rule name="PP">
<seq>
<query match="tok[@ctag = 'prep']/">
<ref name="NP1"/>
</query>
<with-param name="case" value="''"/>
</seq>
</rule>
\end{verbatim}

This rule identifies a sequence whose first element is a token tagged as a preposition and whose subsequent elements are identified by a rule called \texttt{NP1}. This latter rule (not shown here for brevity) is a parameterised rule which finds a nominal phrase (NP) of a given case, but the way it is called above (value='''') ensures that it will find an NP of any case.

The grammar GR contains 44 rules, in a 14K file, most of them more complex than the PP rule above (the average size of a rule is 11.8 lines, compared with 8 lines above).

The grammar is split into 4 layers, with rules of each layer possibly calling only rules of the same and previous layers. The first layer contains low-level rules making reference to particular orthographic forms, such as rules correcting the results of the tagger (e.g., a rule gluing back e and \textit{learning} incorrectly split by the tagger), rules finding various textual realisations of the expression corresponding to the English \textit{‘that is’} or \textit{‘namely’} (i.e., \textit{to jest}, \textit{tj.}, etc.), but also rules identifying Polish copulae (być, to) and a rule identifying verbs characteristic of defining sentences (called \textit{definitor verbs} in Storrer and Wellinghoff 2006), e.g., \textit{oznaczać ‘signify’, określą ‘depict’, obejmować ‘comprise’}, etc.

The second layer contains linguistically justified rules identifying nouns, NPs, PPs, etc.

The third layer contains various important auxiliary rules, e.g., a rule finding a possible term, i.e., an NP, possibly followed by any number of genitive NPs, PPs or parenthetical expressions, perhaps intermingled.

Finally, the fourth layer contains 12 top-level rules, corresponding to various types of definitions, e.g., two rules for copular definitions (one for the nominative pattern and one for the instrumental pattern), rules for various kinds of parenthetical definitions (one based on parentheses, another for \textit{that is} expressions), a rule for structural definitions with the use of a colon, etc.

After evaluating the grammar on the held-out data, the grammar was extended to GR\textsuperscript{’}, containing 13 top level rules (with 48 rules in total, in a 16K file, 12.5 lines for a rule, on the average).

### 4. Experiments and Results

Apart from grammars described above, GR and GR\textsuperscript{’}, we constructed three baseline grammars: B1, which marks all sentences as definition sentences; B2, which marks as definitory all sentences containing a possible copula (\textit{jest, sq, to}), the abbreviation \textit{tj.} ‘\textit{i.e.},’ or the word \textit{czyli ‘that is’}, \textit{‘namely’}; and B3, a very permissive grammar marking as definitions all sentences containing any of the 27 very simple patterns (in most cases, single-token keywords) manually identified on the basis of manually annotated definitions (these patterns include all patterns in B2, as well as various definitor verbs, apparent acronym specifications, the equal sign ‘=’, etc.).

For all five grammars (B1, B2, B3, GR and GR\textsuperscript{’}), experiments were conducted on two corpora: the training corpus, consulted when developing the grammar, and a testing corpus, unseen by grammar developers.\textsuperscript{7}

In each experiment, precision and recall were calculated at two levels: at the token level and at the sentence level, as both ways of the evaluation of definition extraction may be found in the literature.\textsuperscript{8} At the token level, we do not provide results for the held-out corpus, as it is a homogeneous and rather unrepresentative corpus of definitions and, hence, the results are systematically worse than either for the training or for the testing corpus.

\textsuperscript{5}We do not provide results for the held-out corpus, as it is a homogeneous and rather unrepresentative corpus of definitions and, hence, the results are systematically worse than either for the training or for the testing corpus.

\textsuperscript{7}Carletta 1996, p. 252, advocating the use of the kappa statistic in Computational Linguistics, draws attention to the importance of selecting appropriate units; we believe that it is sentences that are such appropriate units for the task at hand.
The results of all experiments are given in the four Tables 2–5, separately for each corpus and for each evaluation level. Apart from precision (P) and recall (R), also the usual F-measure is given (F2), as well as F5 used in Przepiórkowski et al. 2007 and F5 (apparently) used in Saggion 2004.9 Note that for the task at hand, where recall is more important than precision, the latter two measures seem appropriate, although whether recall is twice as important as precision (F2) or five times as important (F5) is ultimately an empirical issue that should be settled by user case evaluation experiments.

5. Comparisons

While the recall of the definition extraction grammar described in §3. is much lower than that of the very permissive baseline grammars, all other measures, i.e., precision and various F-measures, show the clear improvement of the grammar over all three baselines, including the relatively sophisticated baseline grammar B3: according to the least favourable (for GR') F5 measure, GR' is a 25% token-level improvement over B3 when calculated on the testing corpus, and 10% sentence-level improvement. The gain is much clearer when, e.g., F2 is taken into account (62% and 35%, respectively).

The results presented above also constitute a clear improvement over earlier definition extraction attempts for West Slavic, reported in Przepiórkowski et al. 2007. The best results given in that paper concern Czech, for which

---

It should, however, be noted that Saggion 2004 uses F5 to evaluate definition answers to particular questions.

---

9It should, however, be noted that Saggion 2004 uses F5 to evaluate definition answers to particular questions.

---

10Przepiórkowski et al. 2007 do not present sentence-level evaluation results; instead they consider another way to calculate precision and recall, based on overlaps of manual and automatic definitions. For reasons of space we do not present here detailed results of this evaluation method, but measured this way our grammar achieves F2 = 33.70 (measured on the testing corpus), compared to their 28.4 for Polish and 33.9 for Czech.

---

11Note also that their input text contained manually added annotations of keywords, many of which were natural candidates for defined terms. No such clues are present in the current experiments.
uator (with the numbers raised to 71.8–75.2% for a subset of 17–18 best rules from the 33-rule grammar). While we can only envy such precision results, it is not clear how the overall definition extraction results, as measured by the F-measures, compare to our system.

Third, in some of the remaining work, the evaluation method is not completely clear or a little different than in the current paper, or the evaluation is limited to a few types of definitions only. For example, Saggion 2004, participating in the TREC QA 2003 competition, measures (apparently) F5 separately for answers to different definition questions, giving the “combined F-score of 0.236”, but it is not clear if this F-score is directly comparable to our F5 of 0.4355 (for GR’, testing corpus). Similarly, Storrer and Wellighoff 2006 give 34% precision and 70% recall for their German definition extraction system, although it is not fully clear whether these measures are token-based, sentence-based, or calculated in some other way, neither is it clear whether the evaluation was performed on previously unseen testing data. In any case, their system seems to be one of the best systems reported so far, probably exceeding our results.12 On the other hand, Westerhout and Monachesi 2007 provide results directly comparable to our token-level results, albeit only for two most frequent and typical types of definitions: copular definitions (precision = 26%, recall = 72%) and definitions involving other connector verbs (precision = 44%, recall = 56%). We are planning to perform similar evaluation of particular definition types in the future.

6. Inter-annotator Agreement

The task of finding definitory contexts in instruction texts is relatively ill-defined: such definitory contexts often provide definitions of terms in implicit or indirect way, without any structural definitional clues. It is sometimes controversial whether a given sentence in fact provides a definitory context, as in the following sentence actually marked as a definitory: Użytkownicy Internetu spotykają i korzystają z portalów, aby dotrzeć do poszukiwanych przez siebie informacji.13 ‘Internet users meet and use portals in order to find the information they seek’. One way to measure the inherent difficulty of the task is to calculate the inter-annotator agreement (IAA).

Przepiórkowski et al. 2007 provide the IAA in terms of the usual Cohen’s kappa statistic, κ, calculated at token-level, i.e., the classification task is understood here as classifying each token as either belonging to a definitory context or not. Calculated this way, they report the IAA for Polish equal to 0.31, while our result is 0.259, i.e., in either case very low for any classification task, and especially low for a binary classification task; this confirms that definition extraction is intrinsically very difficult.

There are at least two possible objections to this way of measuring IAA. First, token-based IAA seems to be flawed as it assumes a probabilistic model in which an annotator throws a weighted coin for each token, which results in many short (often one-token long) “definitions”, rather than a few long ones. In order to address this problem, we also calculated IAA at the sentence level, approximating definition extraction as a classification problem of marking a sentence as definitory or not. For that we used a subcorpus of the training corpus, i.e., over 83K-token parts of a popular book about computers, in which the original annotator marked 158 definitions (including 13 definitions split across 2 sentences). It turned out that the second annotator found as many as 595 definitory contexts (none split across sentences) in the same text! The comparison resulted in the very low sentence-level κ of 0.307, which confirms cross-linguistic token-level results of Przepiórkowski et al. 2007 and Westerhout and Monachesi 2007 ranging from 0.26 (Westerhout and Monachesi 2007, for one pair of annotators, for Dutch; cf. also our result for Polish given above), through 0.31 (Przepiórkowski et al. 2007, for Polish) and 0.42 (Westerhout and Monachesi 2007, for another pair of annotators, for Dutch), to 0.44 (Przepiórkowski et al. 2007, for Czech). Our contingency tables for both ways of calculating IAA are given in Tables 6–7.14

12If these are sentence-level results, as seems probable, the values of F1, F2 and F5 are, respectively, 45.77, 51.74 and 59.50, compared to our 35.68, 42.95 and 53.94 for GR and for the training corpus.

13This sentence actually contains a grammatical error, as the intransitive spotykac się is an inherently reflexive verb, while the sentence lacks the reflexive marker się.

14Note that some sentences contain 2 or more definitions, hence, the numbers of definitory sentences in Table 7 do not exactly correspond to the numbers of definitory contexts reported above.

15Depending on the annotator, non-definitory tokens constitute 91.2% or 96.0% of all tokens; for sentences the proportions are 84.6% or 95.3%.
and bias effects by replacing, in the contingency table, the values of the two (diagonal) agreement cells with their average, and the values of the two (off-diagonal) disagreement cells with their average, and then proceeding as in case of Cohen’s κ. The token-level and sentence-level values of PABAK for our experiments are, respectively, 0.821 and 0.742.

However, we side with those authors who note that some effects of bias and prevalence on the magnitude of kappa are actually meaningful and consider PABAK on its own as “uninformative” (Sim and Wright, 2005, p. 264). Instead, it is more instructive to compare κ with the maximum value κ could attain given the actual proportions of decisions by annotators (Dunn, 1989; Sim and Wright, 2005); in order to calculate such κmax, as much as possible weight should be moved from the disagreement cells in a contingency table to the agreement cells, but without changing the marginal totals. Such maximal κ values are 0.606 (token-level) and 0.425 (sentence-level), to be compared with 0.259 and 0.307, respectively. This comparison clearly demonstrates that the IAA agreement is much higher at the sentence level (0.307 out of 0.425) than at the token level (0.259 out of 0.606), which provides one more argument for evaluating definition extraction at sentence level, as common in Question Answering, rather than at token level, as in Westerhout and Monachesi 2007 or Przepiórkowski et al. 2007.

7. Conclusion

This paper presents the results of a series of experiments on definition extraction for Polish, an inherently very difficult task, which seem to be comparable to the state of the art. Unlike most other reports on such experiments, we carefully distinguish between training, held-out and testing data, with the testing corpus never, even indirectly, consulted by grammar developers. We also explicitly distinguish between token-level and sentence-level results, although the latter seem to be more relevant for the task at hand: a developer or maintainer of a Learning Object, when creating a glossary, should probably be presented with full sentences possibly containing definitorial contexts, rather than shorter snippets of texts. Measured at this level, our system achieves 75% recall and over 20% precision on previously seen text, or almost 60% recall and almost 19% precision on new texts, and it compares favourably to previous Slavic definition extraction experiments, far exceeding previous results for Polish.

References


Applying Ontology-Based Lexicons to the Semantic Annotation of Learning Objects

Kiril Simov, Petya Osenova
Linguistic Modelling Laboratory, IPOI, Bulgarian Academy of Sciences
Acad. G.Bonchev 25
Sofia 1113, Bulgaria
{kivs|petya}@bultreebank.org

Abstract
This paper discusses the role of the ontology in the definition of domain lexicons in several languages and its usage for the semantic annotation of Learning Objects (LOs). We assume that the ontology has the leading role and the lexicons are created on the basis of the meanings defined within the ontology. The semantic annotation requires the construction of special partial grammars connected to the terms in the lexicons. These special grammars are used for automatic annotation of domain texts. The ambiguous cases are resolved manually on the base of the context. The process of semantic annotation plays a twofold role: first, it produces semantically annotated texts (gold standard corpus), and second, it helps in checking the coverage of the lexicon as well as the precision of the ontology.

Keywords
Ontology, Ontology-based lexicon, Semantic annotation.

1. Introduction
LT4eL European project[1] aims at demonstrating the relevance of the language technology and ontology document annotation for improving the usability of learning management systems (LMS). This paper discusses the role of the ontology in the definition of domain lexicons in several languages and its usage for the semantic annotation of Learning Objects (LOs). The relation between the domain ontology and the domain texts (the learning objects) is mediated by two kinds of information (or layers of information as they are called in [2]) – domain lexicons and concept annotation grammars. In our model the lexicons are based on the ontology. We assume that the ontology is defined first in a formal way and then the lexicons are built on the basis of the concepts and relations defined in the ontology. The terms in the lexicons are mapped to grammar rules for partial analyses of texts. These rules constitute annotation grammars for recognizing the ontological concepts in the texts. The last component of a complete set of knowledge sources for semantic annotation – the disambiguation rules – is not implemented within the project. For the experiments within the project and for the creation of gold standard for concept annotation we have disambiguated the LOs manually.

The structure of the paper is as follows: in the next section we give a short overview of ontology creation process in the project; then we present in detail the lexicon model that we exploit within the project; section 4 discusses the annotation of learning objects with concepts; section 5 outlines the main problems of the current annotation process; the last section concludes the paper and points to the future work.

2. LT4eL Domain Ontology
The domain of LT4el Project is “Computer Science for Non-Computer Scientists”. It covers topics like operating systems; programs; document preparation – creation, formatting, saving, printing; Web, Internet, computer networks; HTML, websites, HTML documents; email, etc. The main application of the ontology has to do with the indexing of LOs within the domain.

The creation of the ontology was done on the basis of manually annotated keywords in the eight languages of the project (Bulgarian, Czech, Dutch, German, Maltese, Polish, Portuguese, Romanian). The annotated keywords were translated into English. Then by search on the Web we collected definitions for the keywords. The set of definitions of a keyword highlights the various meanings of the keyword and the relations between its meanings and other concepts. After the determination of the keywords meanings we created concepts corresponding to these meanings.

These concepts constitute the backbone of the domain ontology. The next step of the ontology development was to map the domain concepts to an upper ontology (in our case we used DOLCE – [3], [4]) in order to inherit some knowledge already encoded in the upper ontology (relations, for instance) and to ensure right concept classification with respect to concept metaproperties defined in the ontology creation methodology – OntoClean [5]. The mapping to the upper ontology was done via

---

[1] http://www.lt4el.eu/ – the LT4eL (Language Technology for eLearning) project is supported by the European Community under the Information Society and Media Directorate, Learning and Cultural Heritage Unit.
OntoWordNet [6] – a version of WordNet restructured in accordance to DOLCE.2

The ontology was extended with additional concepts taken from: the restriction on already existing concepts (for example, if a program has a creator, the concept for program creator is also added to the ontology); superconcepts of existing concepts (if the concept for text editor is in the ontology, then we add also the concept of editor (as a kind of program) to the ontology); missing subconcepts (if left margin and right margin are represented as concepts in the ontology, then we add also concepts for top margin and bottom margin); from the annotation of the learning objects (if a concept is represented in the text of a learning object and it is relevant for the search within the learning material, we add the concept to the ontology). After having applied these steps we built a domain ontology (the current version) with about 750 domain concepts, about 50 concepts from DOLCE and about 250 intermediate concepts from OntoWordNet. We also have about 200 new concept candidates to be added to the ontology. They are extracted from LOs in the different languages on the basis of their ontological annotation.

Although we have started with keywords annotated within the LOs, they are not enough to connect the concepts in the ontology with their explications within the texts. The incompleteness is due to the following facts: (1) not all keywords are shared by the annotations in all languages; (2) some concepts from the extension of the first set of concepts (created on the basis of the keywords) appear in the texts, but were not annotated as keywords. Thus, in order to connect the concepts within the ontology (later also the relations) to the text of the LOs we need also a lexicon for each language aligned to the ontology; a mechanism for recognition of the lexical items in the texts; and a mechanism for selection of the right concept for a phrase in the text when the phrase is ambiguous with respect to several concepts.

3. Ontology-Based Lexicon Model

In this section we present the model and the creation of the lexicons for each project language on the basis of the existing ontology. The lexicons represent the main interface between the user's query, the ontology and the ontological search engine. The annotation of the learning object is facilitated by an additional language tool – annotation grammars for concepts which will be discussed later.

There exist various attempts to approach this mapping task. Most of them start from lexicon compilation for different languages, and then try to establish the connection to the concept. Such initiatives were WordNet [7], EuroWordNet [8], SIMPLE [9]. In spite of the fact that we employ the experience from these projects (mapping to WordNet and Pustejovsky’s ideas in SIMPLE), we also suggest an alternative in connecting the ontology and the lexicons. Our model is very close to LingInfo model (see [11]) with respect to the mapping of the lexical items to concepts, but also with respect to the other language processing tools we connect to the ontology – the concept annotation grammars and concept disambiguation tools. We will discuss LingInfo below.

The terminological lexicons were constructed on the basis of the formal definitions of the concepts within the ontology. By this approach of construction of the terminological lexicon we escaped the hard task of mapping different lexicons in several languages as it was done in EuroWordNet Project [8]. The main problems with this approach are that (1) for some concepts there is no lexicalized term in a given language, and (2) some important term in a given language has no appropriate concept in the ontology which to represent its meaning. In order to solve the first problem we allow the lexicons to contain also non-lexicalized phrases which have the meaning of the concepts without lexicalization in a given language. Even more, we encourage the lexicon builders to add more terms and phrases to the lexicons for a given concept in order to represent as many ways of expressing the concept in the language as possible. These different phrases or terms for a given concept are used as a basis for construction of the regular grammar rules for annotation of the concept in the text. Having them, we could capture in the text different wordings of the same meaning. The following picture shows the mapping varieties:

---

2 We also a mapped the domain concepts to WordNet 2.0 and in this way we ensured a mapping to SUMO [10].
language as form(s) of a lexicalized term (or item), or as a free phrase. In general, a concept might have a few terms connected to it and a (potentially) unlimited number of free phrases expressing this concept in the language. Some of the free phrases receive their meaning compositionally regardless their usage in the text, other free phrases denote the corresponding concept only in a particular context. In our lexicons we decided to register as many free phrases as possible in order to have better recall on the semantic annotation task. In case of a concept that is not-lexicalized in a given language we require at least one free phrase to be provided for this concept.

In order to solve the second problem we modify the ontology in such a way that it contains all the important concepts for the domain. However, this solution requires a special treatment of the "head words" in the lexicons, because such phrases allow bigger freedom with respect to their occurrences in the text. Variability is a problem even with respect to the lexicalized cases and our idea is to represent the most frequent (based on the learning objects we already processed) variants for each concept. We are not able to solve this problem in general within the project, but we hope to demonstrate some approaches to it.

The specific solutions for the lexical terms without appropriate concept in the ontology are the following:

More detailed classes in the ontology. In cases where it was possible, we created more specific concepts in the ontology. For example, the concept of shortcut, as it was initially defined, was the general one, but the lexical items in English to some extent depend on the operating system, because each operating system (MS Windows, Linux, etc) as a rule introduces its own terminology. When the notion is borrowed in other languages, it could be borrowed with different granularity, thus, we introduce more specific concepts in the ontology in order to ensure correct mapping between languages.

More complex mapping between the ontology and some lexicons. Our initial idea was that each meaning of a lexical item in any language is mapped to exactly one concept in the ontology. If for some lexical item this one-to-one mapping is not appropriate or it requires very complicated changes in the ontology, we realize a mapping based on OWL expressions. This mechanism allows us to keep the ontology simpler and more understandable, and to handle cases that do not allow appropriate mappings. Currently, such cases are not detected in our domain.

We could summarize the connection between the ontology and the lexicons in the following way: the ontology represents the semantic knowledge in form of concepts and relations with appropriate axioms; and the lexicons represent the ways in which these concepts can be realized in texts in the corresponding languages. Of course, the ways in which a concept could be represented in the text are potentially infinite in number, thus, we could hope to represent in our lexicons only the most frequent and important terms and phrases. Here is an example of an entry from the Dutch lexicon:

```xml
<entry id="id60">
  <owl:Class rdf:about="lt4el:BarWithButtons">
    <rdfs:subClassOf>
      <owl:Class rdf:about="lt4el:Window"/>
    </rdfs:subClassOf>
    <owl:Class>
      <def>A horizontal or vertical bar as a part of a window, that contains buttons, icons.</def>
      <term lang="nl">werkbalk</term>
      <term lang="nl">menubalk</term>
      <term lang="nl">balk met knoppen</term>
    </owl:Class>
  </owl:Class>
</entry>
```

Each entry of the lexicons contains three types of information: (1) information about the concept from the ontology which represents the meaning for the terms in the entry; (2) explanation of the concept meaning in English; and (3) a set of terms in a given language that have the meaning expressed by the concept. The concept part of the entry provides minimum information for formal definition of the concept. The English explanation of the concept meaning facilitates the human understanding. The set of terms stands for different wordings of the concept in the corresponding language. One of the terms is the representative for the term set. Note that this is a somewhat arbitrary decision, which might depend on frequency of term usage or specialist's intuition. This representative term will be used where just one of terms from the set is necessary to be used, for example as an item of a menu. In the example above we present the set of Dutch terms for the concept lt4el:BarWithButtons. One of the term is non-lexicalized - attribute type with value nonlex. The first term is representative for the term set and it is marked-up with attribute shead with value 1.

Here we present a (part of) DTD for the lexicon:

```xml
<!ELEMENT LT4ELLex (entry+)>  
<!ELEMENT entry ((owl:Class|rdf:Description|rdf:Property), def, termg+)>  
<!ELEMENT def (#PCDATA)>  
<!ELEMENT termg (term+,def?)>  
<!ATTLIST termg
  lang (bg|cs|en|nl|pl|pt|ro) # REQUIRED
>  
<!ELEMENT term (#PCDATA)>  
<!ATTLIST term
  type (lex|nonlex) "lex"
  shead (1|0) "0"
  gram CDATA #IMPLIED
>
```
The lexicon consists of entries. Each entry consists of a concept, relation or instance (partial) definition, followed by a definition of the concept content in English and one or several term groups. Each term group represents all the available lexical terms or free phrases for the corresponding concept (relation or instance) in a given natural language (determined by the attribute lang). Optionally, the term group for a given language could contain a definition of the concept in that language. Each term represents a normalized form of the term. Additionally, we could state whether: the term is a lexicalization of the concept in the language or it is a free phrase (attribute type); the term is representative for the concept in the language (the attribute shead) or not; and which grammar rules recognize this term (related to the concept (relation or instance) of the entry) in text. The format of the currently implemented grammars is given below.

As it was mentioned above, we envisage two more language tools to help in the annotation of text with ontology information, namely – concept annotation grammars and (sense) disambiguation rules. The first kind of information could be seen as a special kind of partial parsing tool which for each term in the lexicon contains at least one grammar rule for recognition of the term. The second kind of information in our project is still in a very preliminary state of development and it is not discussed here.

For the implementation of the annotation grammar we rely on the grammar facilities of the CLaRK System\(^3\). The structure of each grammar rule in CLaRK is defined by the following DTD fragment:

```xml
<!ELEMENT line (LC?, RE, RC?, RM, Comment?) >
<!ELEMENT LC (#PCDATA)>
<!ELEMENT RC (#PCDATA)>
<!ELEMENT RE (#PCDATA)>
<!ELEMENT RM (#PCDATA)>
<!ELEMENT Comment (#PCDATA)>
```

Each rule is represented as a line element. The rule consists of regular expression (RE) and category (RM = return markup). The regular expression is evaluated over the content of a given XML element and could recognize tokens and/or annotated data. The return markup is represented as an XML fragment which is substituted for the recognized part of the content of the element. Additionally, the user could use regular expressions to restrict the context in which the regular expression is evaluated successfully. The LC element contains a regular expression for the left context and the RC for the right one. The element Comment is for human use. The application of the grammar is governed by XPath expressions which provide additional mechanism for accurate annotation of a given XML document. Thus, the CLaRK grammar is a good choice for implementation of the initial annotation grammar.

The creation of the actual annotation grammars started with the terms in the lexicons for the corresponding languages. Each term was lemmatized and the lemmatized form of the term was converted into regular expression of grammar rules. Each concept related to the term is stored in the return markup of the corresponding rule. Thus, if a term is ambiguous, then the corresponding rule in the grammar contains reference to all concepts related to the term.

The following picture depicts the relations between lexical items, grammar rules and the text:

![Diagram showing relations between lexical items, grammar rules, and domain texts](http://www.bultreebank.org/clark/index.html)

The relations between the different elements of the models are as follows. A lexical item could have more than one grammar rule associated to it depending on the word order and the grammatical realization of the lexical item. Two lexical items could share a grammar rule if they have the same wording, but they are connected to different concepts in the ontology. Each grammar rule could recognize zero or several text chunks.

In the next two sections we present the process of annotation of LOs with the grammars constructed in the way, explained above. Also, we discuss the problematic cases. Because of lack of any disambiguation rules at this stage of our work we did the disambiguation manually (see Figure 1 at the appendix).

4. Semantic Annotation of Learning Objects

From the perspective of the Learning Management System, the semantic (ontological) annotation concerns only the metadata section of the learning objects. In the metadata,
according the Learning Object Metadata [12] standard, some ontological information can be stored and used later on to index the learning objects for retrieval. The annotation needs not be anchored to the content of the learning object. The annotator of the learning object can include in the annotation all concepts and relations he/she considers to be important for the classification of the learning object. However, in order to accurately link a learning object and/or its parts to the proper places in the conceptual space of the ontology, the inline annotation of the content of learning objects becomes an obligatory intermediate step in the meta-annotation of the learning objects with ontological information. The inline annotation is done by regular grammar rules attached to each concept in the ontology reflecting the realizations of the concept in texts of the corresponding languages (as it was explained in the previous section). Additionally, rules for disambiguation between several concepts are applied when a text realization is ambiguous between several concepts. Recall that at the current stage of the project we do not have a great progress on disambiguation rules.

Within the project we performed both types of annotation, inline and through metadata. The metadata annotation is used during the retrieval of learning objects from the repository. The inline annotation will be used in the following ways: (1) as a step to metadata annotation of the learning objects; (2) as a mechanism to validate the coverage of the ontology; and (3) as an extension of the retrieval of learning objects where, except for the metadata, we could use also cooccurrences of concepts within the whole LO or its subparts (paragraphs or sentences).

Let us consider in more detail the strategy behind the semantic annotation. The annotation was done through a version of CLaRK System that includes the appropriate DTDs, layouts, grammar and constraints. The process included the following phases: (1) preparation for the semantic annotation and (2) actual annotation. The former refers to the compilation of appropriate regular grammars that explicate the connection between the domain terms in some natural language and the ontological concepts. It also considers the construction of a DTD, layouts and support semi-automatic tools for assigning and disambiguating concepts, namely the constraints. The annotation phase envisages the execution of the above-mentioned tools. The regular grammar finds the occurrences of terms in the text and it assigns all the possible concepts per term. As it was explained in the previous section, the regular grammars were constructed automatically on the basis of the lemmatization of the terms in the lexicons. Thus, in some cases the grammar can under- or over-generate.

The constraints, on the other hand, aim at making the annotation process more accurate. The constraints support the manual annotation. They work in the following way – if there is no ambiguity, the unique concept is assigned as an annotation. If the term is ambiguous, then the constraint proposes to the annotator the possible options and he or she has to select the right choice. The annotator has two possibilities: using Constraint 1 (Select Concept) or using Constraint 2 (Select LT4el Concept). The Constraint 1 stops at each recognized term despite being ambiguous or non-ambiguous one and suggests artificial ambiguity via the options ERASE and EXTENDED. The option ERASE is chosen when a concept was assigned to a common word, not term. The option EXTENDED is chosen when a concept is recognized partially. This option covers two basic cases: occurrence of general vs. specific notions (e.g. Internet vs. Wireless Internet), or notions that can be expressed by single word as well as multiwords (disc vs. hard disc; user vs. end-user). There is a third option, which is incorporated into both constraints, namely – adding a correction over a concept. This happens when the term is used in broader or a narrower sense, which lacks in the assigned concept (e.g. Insert concept in narrowing sense of the term Paste, and in the broader sense of the term Insert). The usage of the Constraint 1 is recommended at the beginning of the annotation process, when the annotation grammar is not considered to be very precise, and when its automatically compiled versions rely only on lemmas.

All these repairing techniques (although subjective and depending on the annotator) lead to the improvement of the regular grammar, which assigns the concepts.

The usage of Constraint 2 can be relied upon at later stage, when the grammar for a language has been improved at least to some extent (as a result from the previous constraint). This constraint does not introduce artificial ambiguity choices. It stops only at real ambiguities in the texts. For example, the term ‘word’ might be assigned two concepts depending on the context: either being common words (WordLang), or elements of computer memory (WordMemory).

We have annotated all the learning objects in all languages in this way. A reasonable part (at least one third) of all learning objects was annotated by using the first constraint. On the basis of the problems found during this annotation we have improved the annotation grammars for some of the languages (Bulgarian, Dutch, English, Portuguese) and have introduced some disambiguation rules for the same languages. In the next section, we present some of the more frequent problems with the current concept annotation.

5. Problematic Cases
In this section we discuss some typical problems of the semantic annotation process, and point out to the possible solutions.

The sources of the problems with assigning the appropriate concepts to the domain terms are of various kinds. We can sum them up in the following way: (1) some
concepts in the ontology might be quite specific or rather broad with respect to the term which they were assigned to; (2) some general terms in a connected text refer to specific entities (in anaphoric relations); (3) the boundaries of the terms might need extension; (4) the ambiguity within concepts might be fake; (5) the detected term has also a common-use meaning and hence – in the context this common meaning is triggered; (6) do verbs receive semantic annotation.

Let us consider each of the above-mentioned issues in more detail.

(1) **Some concepts in the ontology might be quite specific or rather broad with respect to the term.** For example, the concept Size is defined as ‘number of unique entries in a database’. However, it has broader sense. The same holds for the concept TableOfContents. Thus, an additional concept is suggested (Content), which covers the whole content of an information object, not just the reference to it.

(2) **Some general terms in a connected text refer to specific entities (in anaphoric relations).** For example, the term is Systems for Personalization, but further in the text they are called just Systems.

(3) **The boundaries of the terms might need extension.** For example, within the term Personalized system, only system is detected, and hence – the more general concept is assigned, namely – ComputerSystem. Extension of terms helps in adding more domain specific concepts to the ontology. The same holds for the term Agent technology, in which only technology was detected and tagged.

(4) **The ambiguity within concepts might be fake.** This problem contributes to the precision of the ontology, since the annotators detected ambiguous-like concepts which practically coincide, such as: Metadata and DescriptiveMetadata; Search and Searching. This issue arose due to several reasons: the naming of the concept is conventional, and thus language-independent. The real pointer is the definition. Consequently, sometimes several conventional namings compete, resulting in spuriousness.

(5) **The detected term has also a common-use meaning.** For example, Help as command and as request. Another example is the word Sector. In the sequence ‘educational sector’ it is not in its computer domain meaning. Also, point in ‘point of view’ is definitely not the format-oriented concept Bullet (at least in Bulgarian there is homonymy).

(6) **Do verbs receive semantic annotation?** This problem concerns preferably the commands and buttons. Button ‘Save’ and the command ‘(to) Save’. We decided not to treat the commands as verbs, and also not to assign real verbs semantic concepts. This was done for practical reasons.

To sum up, most of the problematic issues might be repaired via the annotator intervention. Later on, the ontology and lexicon builders could use these observations for extending the lexicons and making the ontology more precise. Needless to say, the specificities of the above cases vary with respect to the language. However, the problem abstraction is cross-lingual.

6. Conclusions

In this paper we have presented the current state of ontology, lexicons and ontology annotation within the LT4el project. We have built a lexicon model which relates ontology concepts, relations and instances to lexical items, grammar rules and disambiguation rules. In this model the central role is played by the ontology which determines the content of the other components of the model. Another advantage of our model is that it supports the work in multilingual environment. The mappings between lexicons and ontology are performed with the aim each term in each language to have a corresponding concept, and vice versa – each concept in the ontology to be exemplified by at least one term expression (be it lexicalized or a free phrase). Thus, the ontology itself as well as specific language lexicons are verified in a cross-lingual context. We have created lexicons and annotation grammars for all the languages in the project. The mapping between the language specific lexicons was facilitated by the ontology. Our model shares common features with other lexicon models: with WordNet-like ([7], [8]) lexicons we share the idea of grouping lexical items around a common meaning and in this respect the term groups in our model correspond to synsets in WordNet model. The difference in our case is that the meaning is defined independently in the ontology. With SIMPLE model ([9]) we share the idea to define the meaning of lexical items by means of the ontology, but we differ in the selection of the ontology which in our case represents the domain of interest, and in the case of SIMPLE reflects the lexicon model. With the LingInfo model ([11]) we share the idea that grammatical and context information also needs to be presented in a connection to the ontology, but we differ in the implementation of the model and the degree of realization of the concrete language resources and tools.

In other approaches to semantic annotation usually information extraction systems are adapted. This choice is appropriate as much as more instance information is expected to be annotated within the domain text. Thus, in such cases recognition of new instances and their classification with respect to the ontology information is of great importance. In our application in addition to the instance information we also have to annotate mentionings of ontological information where the properties of the concepts are stated in general and not as on a particular instance. Thus, the term lexicons are a good way to create
the necessary resources for implementing the model for many languages.

In future, more work has to be done on the extension of the annotation grammars, on the implementation of disambiguation rules, on the connection of the lexicon and the grammars to non-domain dependent concepts. Also, we need to add domain relations.

7. Acknowledgements
This work has been supported by LT4eL (Language Technology for eLearning) project of the European Community under the Information Society and Media Directorate, Learning and Cultural Heritage Unit.

We would like to thank our partners from the project for their useful feedback. Especially to Paola Monachesi, Cristina Vertan, Lothar Lemnitzer, Corina Forascu, Claudia Borg, Rosa Del Gaudio, Janine Trapman, Beata Wójtowicz, Eelko Mossel.

8. Appendix
The following screenshot depicts manual disambiguation within CLaRK System. The grammars were used to annotate the texts which refer to some concepts. In case of ambiguities the user is asked to select the relevant concept from a list of possible ones.

Figure 1: Here the concept *Title* is chosen as the correct one for the term *Titel* in Dutch.

9. References
Combining pattern-based and machine learning methods to detect definitions for eLearning purposes

Eline Westerhout and Paola Monachesi
Utrecht University
Trans 10
Utrecht, The Netherlands
Eline.Westerhout,Paola.Monachesi@let.uu.nl

Abstract

One of the aims of the Language Technology for eLearning project is to show that Natural Language Processing techniques can be employed to enhance the learning process. To this end, one of the functionalities that has been developed is a pattern-based glossary candidate detector which is capable of extracting definitions in eight languages. In order to improve the results obtained with the pattern-based approach, machine learning techniques are applied on the Dutch results to filter out incorrectly extracted definitions. In this paper, we discuss the machine learning techniques used and present the results of the quantitative evaluation. We also discuss the integration of the tool into the Learning Management System ILIAS.

1 Introduction

Glossaries can play an important role within eLearning, since they are lexical resources which can support the learner in decoding the learning object he is confronted with and in understanding the central concepts which are being conveyed in the learning material. Therefore, existing glossaries or wikipedia entries can be linked to learning objects, but an obvious shortcoming of this approach is that the learner would be confronted with many definitions for the term he is looking for and not only with the definition which is appropriate in the given context.

A better alternative is to build glossaries based on the definitions of the relevant terms which are attested in the learning objects. By doing so, the exact definition that the author of a certain document uses is captured; in many cases, this definition overrides a more general definition of the term. By providing the most appropriate definition to the learner for the concept he is not familiar with, we facilitate the learning process.

One of the aims of the European project Language Technology for eLearning (LT4eL)\(^1\) is to show that Language Technology can provide a solution to the task of creating appropriate glossaries by developing a glossary candidate detector. More generally, the goal of the project is to show that the integration of Language Technology and Semantic Web techniques will enhance Learning Management Systems (LMS) and thus the learning process.

Definition extraction is the topic of much current research and techniques have been developed to this end within the Natural Language Processing and the Information Extraction communities mainly based on grammars that detect the relevant patterns and machine learning methods; in the LT4eL project, we adapt these techniques for eLearning purposes.

The glossary candidate detector we have developed, extracts definitions in all the eight languages represented in our consortium, that is, Bulgarian, Czech, Dutch, English, German, Polish, Portuguese and Romanian [17], [16]. In this paper, however, we focus only on the definitory contexts attested in the Dutch learning objects and the approach we have used to identify them. First, a substantial amount of definitions is selected and annotated manually in the learning objects which are the asset of this project. From these examples, a grammar is developed in order to extract possible definitions (cf. [18] for a similar approach). After the extraction of the definition patterns, machine learning techniques are applied on the extracted definitions to improve precision (cf. also [7] for Dutch).

The rest of the paper is organized as follows. Section 2 introduces related work on the area of definition extraction. The details of our approach are presented in section 3, in particular we discuss the corpus we have assembled, the grammar we have developed to detect definitions from our corpus of learning objects and the machine learning techniques employed to narrow down the set of definitions. Section 4 evaluates the results obtained. In section 5, we discuss the embedding of the glossary candidate detector within the Learning Management System ILIAS\(^2\) and its function within an eLearning context while section 6 contains our conclusions and suggestions for future work.

2 Previous work

Research on the detection of definitions has been pursued in the context of automatic building of dictionaries from text, question-answering and recently also within ontology learning.

In the area of automatic glossary creation, the DEFINDER system [18] combines shallow natural lan-

\(^{1}\) http://www.lt4el.eu

\(^{2}\) http://www.ilias.de
Language processing with deep grammatical analysis to identify and extract definitions and the terms they define from on-line consumer health literature. The system is based on two modules, the former one uses cue-phrases and text markers in conjunction with a finite state grammar to extract definitions while the latter one uses a grammar analysis module based on a statistical parser in order to account for several linguistic phenomena used for definition writing. Their approach relies entirely on manually crafted patterns.

Research on definition extraction has been pursued very actively also in the area of Question-Answering. The answers to ‘What is’-questions are usually definitions of concepts. A common approach in this area is to search the corpus for sentences consisting of a subject, a copular verb and a predicative phrase. If the concept matches the subject, the predicative phrase is returned as answer, also in this case research relied initially almost totally on pattern identification and extraction and only later, machine learning techniques have been employed.

In [21], both the analysis of document structure as well as dependency parsing are explored. Definitions consisting of a subject, a copular verb and a predicative phrase are extracted from Dutch texts in order to provide answers to questions in the medical domain. The texts used are often encyclopedias and wikipedia entries which are well structured and thus layout information is a reliable feature to detect definitions in a text, this is however not the case for other types of texts. Therefore, for texts that are not well structured the parsing approach is more promising. However, medical questions often require answers which are longer than a single sentence while parsing techniques are typically applied to sentences. Thus, a better alternative might be to combine the two approaches and [7] is an attempt in that direction. They propose an approach to definition extraction which operates on fully parsed text and machine learning techniques (cf. also [2], [14] for the use of machine learning methods in definition extraction). Also in this case, a rather well structured corpus is employed such as the medical pages of the Dutch version of Wikipedia. Therefore, first candidate definitions which consist of a subject, a copular verb and a predicative phrase are extracted from a fully parsed text by using their syntactic properties. Second, machine learning methods are applied to distinguish definitions from non-definitions and to this end a combination of attributes have been exploited which refer to text properties, document properties, and syntactic properties of the sentences. They show that the application of machine learning methods improve considerably the accuracy of definition extraction based only on syntactic patterns.

Research on definition extraction has been carried out also in the area of ontology learning. For example, within the German HyTex project [20], 19 verbs that typically appear in definitions were distinguished and search patterns have been specified based on the valency frames of these definator verbs in order to extract definitions. Furthermore, semantic relations have been extracted from these definitions. Even though this information has been employed for the automatic generation of hypertext views that support both reading and browsing of technical documents, one could imagine employing the same technique to actually update and enlarge existing formalized ontologies.

Work in this direction is that of [23] that proposes a rule-based method for extracting and analyzing definitions from parsed text on the basis of a semi-automatically oriented parsing system. The results are then employed to improve the quality of text-based ontology learning. Also this approach relies on pattern extraction techniques to detect definitions and doesn’t employ machine learning. A difference with respect to previous systems is its use of semantic information in the identification of patterns.

3 The glossary candidate detector

The extraction of definitions for glossary creation for eLearning purposes constitutes a novel application of current techniques which presents some interesting challenges.

The most relevant one is constituted by the corpus of learning objects which includes a variety of text genres and also a variety of authors writing styles that pose a real challenge to computational techniques for automatic identification and extraction of definitions together with the headwords. Our texts are not as structured as those employed for the extraction of definitions in question-answering tasks which include encyclopedias and wikipedia entries, thus layout information plays in our context a marginal role.

Furthermore, some of our learning objects are relatively small in size, thus our approach has not only to favor precision as is often the case in the approaches discussed in the previous section but also recall, that is, we want to make sure that all possible definitions present in a text are proposed to the user for the creation of the relevant glossary. Therefore, the extraction of definitions cannot be limited to sentences consisting of a subject, a copular verb and a predicative phrase, as is often the case in question-answering tasks, but a much richer typology of patterns needs to be identified than in current research on definition extraction.

Despite the challenges that the eLearning application involves, we believe that the techniques for the extraction of definitions developed within the Natural Language Processing and the Information Extraction communities can be adapted and extended for our purposes. In particular, our approach is similar to that of [18] since we use a grammar to identify a wide variety of possible definition patterns. However, we follow [7] in applying machine learning techniques to improve the precision of the definition extracted and distinguish definitions from non-definitions.

3.1 The grammar component

As already mentioned, the first step in the detection process of definitions is the development of a grammar which is able to identify the relevant patterns.

In order to detect the most common patterns in our corpus and write appropriate rules for their extrac-
Table 1: Examples for each of the definition types

<table>
<thead>
<tr>
<th>Type</th>
<th>Example sentence</th>
</tr>
</thead>
<tbody>
<tr>
<td>is_def</td>
<td>‘Gnuplot is a program for drawing graphs’</td>
</tr>
<tr>
<td>verb_def</td>
<td>‘E-learning omvat hulpmiddelen en toepassingen die via het internet beschikbaar zijn en creatieve mogelijkheden bieden om de leerervaring te verbeteren.’</td>
</tr>
<tr>
<td>punct_def</td>
<td>‘Passes: plastic cards equipped with a magnetic strip, that can be swiped through a card reader, by means of which the identity of the user can be verified and the user gets access to certain facilities.’</td>
</tr>
</tbody>
</table>
| layout_def | ‘RABE
 Een samenwerkingsverband van een aantal Duitse bibliotheken, dat gezamenlijk een Internet inlichtingendienst bieden, gevestigd bij de gemeenschappelijke catalogus, HBZ, in Keulen.
 ‘RABE
 Cooperation of a number of German libraries, that together provide an Internet information service, residing at the common catalogue, HBZ, in Cologne’ |
| pron_def | ‘Dedicated readers. These are special devices, developed with the exclusive goal to make it possible to read e-books.’ |

Given the variety of definition patterns present in our learning objects, we believe that the rule-based approach is the most appropriate to use to detect them. Previous research has shown that grammars that match the syntactic structures of the definitory contexts are the most successful approaches when deep syntactic and semantic analysis of texts is not available [18], [13]. We have extracted definitions from a corpus of learning material which has different formats, such as HTML, PDF or DOC. All these formats are via several steps converted into XML conforming to the LT4ELAna DTD, which is an adapted version of the XCES DTD for linguistically annotated corpora [8]. For our purposes, the XCES DTD has been enriched with elements that are relevant for our project. Besides the content of the original files (that is, information about layout and the text itself), the DTD allows encoding information about part-of-speech, morphosyntactic features and lemmas. The Wotan tagger presented in [5] has been used for the annotation of the Dutch learning objects with part-of-speech information and morphosyntactic features whereas the CGN lemmatizer discussed in [3] was used for the lemmatization. It should be noticed that the rules of the grammar for the extraction of the definitory context patterns make use also of the the information encoded in the LT4ELAna format.

The XML transducer *lxtransduce* developed by [22] is used to match the grammar against files in the LT4ELAna format. *Lxtransduce* is an XML transducer, especially intended for use in NLP applications. It supplies a format for the development of grammars...
which are matched against either pure text or XML documents. The grammars must be XML documents
which conform to a DTD (lxtransduce.dtd, which is part of the software). In each grammar, there is one ‘main’ rule which calls other rules by referring to them. The XPath-based rules are matched against elements in the input document. When a match is found, a corresponding rewrite is done.

The grammar contains rules that match the grammatical patterns described above. It is split into 4 layers, with rules of each layer possibly calling only rules of the same and previous layers. In the first layer, the part-of-speech information is used to make rules for matching separate words (e.g. verbs, nouns, adverbs). The second layer consists of rules to match chunks (e.g. noun phrases, prepositional phrases). We did not use a chunker to parse our data, because we wanted to be able to define the possible patterns of the chunks ourselves. The third layer contains rules for matching and marking the defined terms and in the last layer the pieces are put together and the complete definitory contexts are matched. The rules were made as general as possible to prevent overfitting to our training corpus.

In total, the grammar consists of 67 rules (part 1: 24 rules; part 2: 5 rules; part 3: 20 rules and part 4: 18 rules) in a 35K file.

An alternative approach could have been to parse the corpus syntactically with Alpino, a robust wide-coverage parser for Dutch [4], as proposed in [7]. However, the experiments showed that we do not need the level of deep syntactic representation produced by Alpino and that a shallower representation, as that produced by our grammar suffices for our purposes. Furthermore, since parsers (and chunkers) are not available for all the languages for which we have developed the glossary candidate detector, a shallow approach was the most promising one.

3.2 The machine learning component

After the detection of a large number of definitory context candidates with the grammar (1098 for the various types of definitions), another step follows to filter incorrectly extracted sentences which cannot be considered definitions. For the filtering, the Naive Bayes machine learning algorithm has been used. The Naive Bayes classifier is a fast and easy applicable classifier based on the probabilistic model of text [15]. It has often been used in text classification tasks ([11], [12]). It is also one of the classifiers used in [7] for the classification of definitions. Because our data set is relatively small, we used 10-fold cross validation for better reliability of the classifier results.

We aim at finding the best attributes for classifying definition sentences. We experimented with combinations of the following attributes (cf. also [7]).

Text properties: bag-of-words, bigrams, and bigram preceding the definition. Punctuation is included as [9] observe that it can be used to recognize definitions (i.e. definitions tend to contain parentheses more often than non-definitions). We include all bigrams in a sentence as feature. The use of the bigram preceding the definition is similar to the use of n-grams by [1] who add n-grams (n being 1, 2 or 3) occurring frequently either directly before or after a target term.

Syntactic properties: type of determiner within the defined term (definite, indefinite, no determiner). [7] investigated the use of determiners in definition sentences. They found out that for their data the majority of subjects in definition sentences have no determiner (62 %), e.g. ‘Paracetamol is een pijnstillend en koortsverlagend middel’ (Paracetamol is an pain alleviating and a fever reducing medicine), while in non-definition sentences subject determiners tend to be definite (50 %), e.g. ‘De werkzame stof is acetylsalicyzuur’ (The operative substance is acetylsalicylic acid).

Proper nouns: presence of a proper noun in the defined term. [7] observed a significant difference in the distribution of this feature between definition and non-definition sentences. Definition sentences tend to have more proper nouns in their subjects (40.63 %) compared to non-definition sentences (11.58 %).

[7] also used the document property of the position of a sentence in the document. For their type of texts (i.e. Wikipedia) this is a relevant feature, however, for our texts which are of a totally different structure this is not relevant. Another feature they used, which is difficult to simulate in our experiment, is the position of the subject in the sentence because we do not have the syntactic structure of sentences but only the part-of-speech information.

We have experimented with ten combinations of these attributes. In the first setting, only a bag-of-words has been used by the classifier and in the second setting only bigrams have been used. The third setting combines unigrams and bigrams. All other settings (4 - 10) use bigrams and the bag-of-words together, in combination with one or more other attributes. Table 2 summarizes the 10 settings. Weka, a collection of machine learning algorithms for data mining tasks, was used to perform the experiments [25].

<table>
<thead>
<tr>
<th>setting</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>using bag-of-words</td>
</tr>
<tr>
<td>2</td>
<td>using bigrams</td>
</tr>
<tr>
<td>3</td>
<td>combining bag-of-words and bigrams</td>
</tr>
<tr>
<td>4</td>
<td>adding bigram preceding definition to setting 3</td>
</tr>
<tr>
<td>5</td>
<td>adding definiteness of article in marked term to setting 3</td>
</tr>
<tr>
<td>6</td>
<td>adding presence of proper noun to setting 3</td>
</tr>
<tr>
<td>7</td>
<td>adding bigram preceding definition &amp; definiteness of article in marked term to setting 3</td>
</tr>
<tr>
<td>8</td>
<td>adding bigram preceding definition &amp; presence of proper noun to setting 3</td>
</tr>
<tr>
<td>9</td>
<td>adding definiteness of article in marked term &amp; presence of proper noun to setting 3</td>
</tr>
<tr>
<td>10</td>
<td>using all attributes</td>
</tr>
</tbody>
</table>

Table 2: Configurations used for the Machine Learning experiment
4 Evaluation

4.1 First step: using the grammar

As already mentioned, the grammar was used to detect definitions on the basis of syntactic patterns and we have calculated precision (P), recall (R) and F-score (F) for each of the types identified by the grammar to evaluate its performance. The manual annotation of definitions was used as gold standard against which precision and recall were measured. We should be aware of the fact that it is possible that the human annotator missed correct definitions or selected definitions which were not correct according to other humans. The sentence was identified as the most appropriate unit to evaluate the performance and therefore we report the results obtained when using the sentence as a unit [19].

<table>
<thead>
<tr>
<th>type</th>
<th>P</th>
<th>R</th>
<th>F</th>
</tr>
</thead>
<tbody>
<tr>
<td>is_def</td>
<td>28.10</td>
<td>86.52</td>
<td>42.41</td>
</tr>
<tr>
<td>verb_def</td>
<td>44.64</td>
<td>75.76</td>
<td>56.18</td>
</tr>
<tr>
<td>punct_def</td>
<td>9.91</td>
<td>68.18</td>
<td>17.31</td>
</tr>
<tr>
<td>pron_def</td>
<td>9.18</td>
<td>41.30</td>
<td>15.02</td>
</tr>
</tbody>
</table>

Table 3: Performance of the grammar

We refer to [24] for more details on the performance of the grammar. [6] present the work done for Portuguese using the same methodology. Their grammar slightly outperforms ours in recall, whereas our precision is much higher.

4.2 Second step: filtering the results using machine learning methods

The precision of the results obtained with the grammar is low, which means that a user who wants to use the glossary candidate detector to create a dictionary is presented with many incorrect definitions. In order to increase precision, we trained a Naive Bayes classifier and applied it on the results obtained with the grammar.

The 10 attribute settings were tested for the two most frequent definition types: the to be-patterns and the punctuation patterns extracted by the grammar. There were 274 to be-patterns extracted, of which 77 were real definitions. This means that we have a precision of 28.1 %. For the punctuation patterns, there were even more incorrect sentences contained. This set includes 454 sentences, of which 45 are correct definitions (precision of 9.9 %).

In classification experiments, often only the accuracy is reported. This is the proportion of correctly classified instances. However, for our purposes the recall and precision of the definitions are more important than the precision and recall of the non-definitory contexts. It is possible, that the accuracy is high, whilst the recall of the definitions is very low; this occurs when the classifier categorizes a large number of non-definitions correctly. Such a large difference between accuracy and recall is clearly present in the results for the punctuation patterns. Therefore, table 4 and table 5 report also the precision, recall and F-score for the definitory contexts.

For the to be-patterns, the accuracy is highest when all attributes are used. The precision, recall and F-score give also best results with this configuration. However, the differences between the settings are small for settings 3 to 10. Only for the first two settings, the scores are remarkably lower.

The accuracy and precision are highest for the punctuation patterns when configuration 7 is used for training the classifier. In this setting, the bigram directly appearing before the defining text and the definiteness of the article are taken into consideration. Recall and F-score are best for setting 9, the setting in which the definiteness of the article and the presence of a proper noun in the marked term are used as attributes.

Although the accuracy scores of the to be-patterns and the punctuation patterns are comparable (both around 90), precision, recall and F-score for the classification of definitory contexts are remarkably lower for the punctuation patterns. This has to do with the fact that there are far more non-definitions for the punctuation patterns whereas we are interested in the classification of the definitions.
The recall values reported in the previous section differ from the recall calculated after applying the grammar and manually annotated set of definitions, thus the final recall necessary to calculate the scores in relation to the previous section are calculated in relation to the values reported in the previous section.

### 4.3 Discussion

It should be noticed that the recall values reported in the previous section are calculated in relation to the number of correct definitions extracted by the grammar. In order to identify the actual recall values, it is necessary to calculate the scores in relation to the manually annotated set of definitions, thus the final recall values calculated after applying the grammar and the machine learning classifier differ from the recall values reported in the previous section.

The final recall is calculated with the formula:

\[
\text{recall} = \frac{\text{final \# correct definitions found}}{\text{\# manually annotated definitions}} \times 100
\]

The precision obtained after the machine learning filtering already represents the final precision values, because it shows the proportion of correctly classified definitions in relation to the total number of sentences classified as definition.

Therefore, the final precision values are the same as the values reported in table 4 and 5. In table 6 and table 7 we report all final results, that is precision, recall and F-score.

When we compare these results to the results obtained by the grammar, we should keep in mind that there is a restriction inherent to our approach: recall cannot improve with respect to the results obtained by the grammar, because we use these results as input. The correct definitions that were not detected by the grammar are definitively lost. As a consequence, it is inevitable that the recall decreases. However, the better the classifier performs, the smaller the loss will be.

For the to be-patterns, using the Naive Bayes classifier leads to an improvement of precision of 51.9% for the best setting (setting 10). Recall drops for this same setting with 19.1%, which means that 17 correct definitions of the 77 extracted by the grammar are lost during the classification step.

For the punctuation patterns, the precision increases with maximal 41.2%. The recall decreases with 31.8%, which means that 21 definitions are lost during the classification step. However, the F-score increases with 26.8%.

There is a trade-off between precision and recall. Before using the classifier, we had better recall whereas after using the classifier the precision was much better. For the 21 files we used, 1098 definitions were extracted by the grammar of which 269 were correct (19.0%). This means that on average 52 definitions are proposed for a file of which only 10 are correct. For the user who has uploaded a learning object and wants to generate a glossary related to it, this means that he has to check the proposed sentences very carefully and that 80% of them have to be thrown away, if we rely only on pattern-based methods to identify correct definitions.

At the moment, we have only employed machine learning methods to filter out results for the to be-patterns and the punctuation patterns. For these categories the grammar extracted 728 sentences of which only 122 were correct (16.8%). After using the Naive Bayes classifier, the number of definitions presented to the user has decreased to 127 of which 86 are correct (67.7%). This means that the user uploading a file is presented with on average 6 possible definitions per file which have to be checked for these two categories. Out of these 6 definitions, 4 are real ones. However, the counter effect of using machine learning after applying the grammar to detect definition patterns is that on average 2 correct definitions per file are lost for these categories. Given that our goal is the automatic development of glossaries for eLearning purposes, it remains to be evaluated whether a pure pattern-based approach for definition extraction might be more appropriate than one in which it is combined with machine learning techniques, as discussed in more detail in the section below.

It is difficult to compare our results with those achieved in the area of definition extraction for automatic building of dictionaries, question-answering and within ontology learning given the different setup, languages involved, applications and aims. Perhaps, the only work we could compare our results with is that of [7] given the similarity of tasks, methodology and languages involved, applications and aims. Perhaps, the only work we could compare our results with is that of [7] given the similarity of tasks, methodology and language. Their results with respect to accuracy are slightly better than ours since their best accuracy is 90.26% for the Naive Bayes classifier with respect to the to be-pattern while in our case the best result is 88.32%. However, it should be noticed that they have employed a much bigger and more structured corpus than ours.

On the other hand, [7] could not measure the effect of using machine learning on recall, because they did not annotate the definitions in their corpus manually and could therefore not compare the results obtained.

<table>
<thead>
<tr>
<th>setting</th>
<th>P</th>
<th>R</th>
<th>F</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>69.44</td>
<td>56.18</td>
<td>62.11</td>
</tr>
<tr>
<td>2</td>
<td>66.25</td>
<td>59.55</td>
<td>62.72</td>
</tr>
<tr>
<td>3</td>
<td>76.62</td>
<td>66.29</td>
<td>71.08</td>
</tr>
<tr>
<td>4</td>
<td>76.62</td>
<td>66.29</td>
<td>71.08</td>
</tr>
<tr>
<td>5</td>
<td>77.63</td>
<td>66.29</td>
<td>71.52</td>
</tr>
<tr>
<td>6</td>
<td>76.62</td>
<td>66.29</td>
<td>71.08</td>
</tr>
<tr>
<td>7</td>
<td>78.67</td>
<td>66.29</td>
<td>71.95</td>
</tr>
<tr>
<td>8</td>
<td>76.32</td>
<td>65.16</td>
<td>70.30</td>
</tr>
<tr>
<td>9</td>
<td>78.94</td>
<td>67.42</td>
<td>72.73</td>
</tr>
<tr>
<td>10</td>
<td>80.00</td>
<td>67.42</td>
<td>73.17</td>
</tr>
</tbody>
</table>

Table 6: Final results for the to be-patterns.

<table>
<thead>
<tr>
<th>setting</th>
<th>P</th>
<th>R</th>
<th>F</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>43.24</td>
<td>24.24</td>
<td>31.07</td>
</tr>
<tr>
<td>2</td>
<td>31.71</td>
<td>19.70</td>
<td>24.30</td>
</tr>
<tr>
<td>3</td>
<td>45.10</td>
<td>34.84</td>
<td>39.32</td>
</tr>
<tr>
<td>4</td>
<td>46.81</td>
<td>33.33</td>
<td>38.94</td>
</tr>
<tr>
<td>5</td>
<td>45.28</td>
<td>36.36</td>
<td>40.34</td>
</tr>
<tr>
<td>6</td>
<td>50.00</td>
<td>36.36</td>
<td>42.11</td>
</tr>
<tr>
<td>7</td>
<td>51.06</td>
<td>36.36</td>
<td>42.48</td>
</tr>
<tr>
<td>8</td>
<td>50.00</td>
<td>36.36</td>
<td>42.11</td>
</tr>
<tr>
<td>9</td>
<td>50.00</td>
<td>39.39</td>
<td>44.07</td>
</tr>
<tr>
<td>10</td>
<td>50.00</td>
<td>36.36</td>
<td>42.11</td>
</tr>
</tbody>
</table>

Table 7: Final results for the punctuation patterns.
to the set of manually annotated definitions. Thus, we cannot evaluate how we compare to them in this respect.

5 Embedding into ILIAS

The glossary candidate detector we have presented, is one of the functionalities which have been integrated in the ILIAS Learning Management System. One of the aims of the LT4eL project is to show that the automatic development of glossaries, on the basis of definitions attested in the learning objects, should help the student in its learning process.

Even though the glossary candidate detector has been integrated into ILIAS, it should be possible to enhance other LMS with it since this functionality has been offered as web service.

Figure 1 shows the first integration of the glossary candidate detector into the ILIAS system. Users can: a) select the option to generate a glossary for the learning object he has uploaded, this implies that the glossary candidate detector will become active and a list with terms and associated definitions will be produced; b) select the term and associated definitions from this list (shown in figure 1); c) generate a glossary on the basis of this list. The possibility of adding additional definitions should also be envisaged. It should be noticed that glossary generation is an interactive task, since the user can decide which definitions are appropriate and which should be removed.

In the previous section, we have discussed a quantitative evaluation of the performance of the glossary candidate detector which is crucial to verify that the tool produces state of the art results. To this end, various techniques have been explored. However, we believe that the best way to evaluate the glossary candidate detector is in the context of its use within ILIAS. Therefore, a scenario based evaluation, which will take user satisfaction into account, might be the best way to decide whether we should privilege recall or precision in the case of our application and thus a pure pattern-based method or a pattern-based method in combination with machine learning filtering.

6 Conclusions

One of the functionalities developed within the LT4eL project is the possibility to derive glossaries automatically on the basis of the definitory contexts identified within the learning objects.

A pattern-based approach is employed to identify the definitory contexts. The current grammar is able to identify most types of definitory contexts and we obtain an acceptable recall while precision should be improved. To this end, machine learning techniques have been employed which have shown that precision can be improved considerably, with the consequence that recall decreases.

Improvements can be envisaged to find a better balance between precision and recall. To this end, we plan to evaluate other classifiers and to include additional features, including semantic ones. Furthermore, we plan to extend the use of machine learning techniques to all types of definitions and not only to the most frequent ones.

Finally, a scenario based evaluation of the glossary candidate detector is envisaged. While a quantitative evaluation might be useful to establish whether the tool produces state of the art results, we wonder whether a qualitative evaluation might not be a better way to evaluate our results. Given the eLearning context in which we operate, it might be thus more relevant to evaluate the degree of satisfaction of the users. These are both the content providers who will exploit this functionality in order to develop glossaries semi-automatically and they can thus select among the proposed definitions those that they consider the most appropriate as well as the learners who thanks to this functionality will have glossaries at their disposal that should facilitate their learning process.

References


