



Project no. 027391

Project acronym: LT4eL
Project title: Language Technology for eLearning

Instrument Specific Targeted Research Project

Thematic Priority Information Society Technology

D2.2b Validated keyword extraction tool – first cycle

Due date of deliverable: 30-11-2007
Actual submission date: 21-12-2007

Start date of project: 1-12-2005

Duration: 30 Months

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

Revision [1]

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

Keyword Extractor Deliverable

Contents

- 1 Title
- 2 Summary
- 3 Improving the keyword extractor
 - 3.1 Improving the runtime performance
 - 3.2 Adjusting the linguistic model
 - 3.3 Additional features of keyword candidates
 - 3.4 Additional distributional statistics
 - 3.5 Combining several values in a combined ranking statistics
- 4 Actions taken in response to user experience
- 5 Documentation
- 6 Scientific papers on the keyword extractor

1 Title

D2.2b Validated keyword extraction tool – first cycle

2 Summary

The first year of the project was devoted to the conceptual planning, development and implementation of the keyword extractor. This was done partially in parallel to the acquisition and linguistic annotation of the corpora. We mainly

- implemented some distributional statistics which were reported in the literature to detect good keyword candidates (i.e. TF*IDF, RDIF and an term frequency adjusted version of IDF, for details and literature, cf. Lothar Lemnitzer, Cristina Vertan, Alex Killing, Kiril Simov, Diane Evans, Dan Cristea, Paola Monachesi: Improving the search for learning objects with keywords and ontologies, the paper is in the appendix to this deliverable)
- implemented a routine which extracts keyphrases of any length
- introduced a linguistic filter for each language which removes words of certain parts of speech

On the review meeting in January 2007 we presented a prototype of this tool as a standalone tool which demonstrated the basic functionality of the extraction of keywords employing one of the methods. The tool was accessible via a simple interface which was useful to demonstrate this basic functionality.

The main activities in the reporting period wrt to the keyword extractor have been the evaluation and the integration of the language technology tools and their improvement in response to evaluation and validation (cf. Technical annex, section 7.2 and milestones for WP2, p. 27; for the evaluation cf. D2.4).

In the reporting period, we made progress in the development of the tool and in particular in its integration in a learning management system and its embedding into the learning process, which is represented by the use of the tool in some usage scenarios. While these usage scenarios are described in detail in the WP5 deliverable, we will outline in the

following the progress we made in developing and integrating the tool. In particular we will describe:

- improvements in the run-time performance of the tool;
- improvements in the linguistic models for each language which are used to filter the words which are inappropriate as keywords
- improvements in the processing and presentation of keyphrases
- use of non-distributional information into the weighting and ranking of keywords. We explore the potential of other distributional measures to improve the ranking of the keywords. We are also investigating the potential of lexical chaining as a means to assess to potential of each keyword candidate to reflect one of the major topics of the text.
- the last issue includes methods of integrating different measures into a combined weight. We outline alternatives in achieving this combined weight and experiments we have done so far to explore these alternatives
- integration of this tool in a learning management system and its embedding into learning processes which reflect the real needs of the users. The development of the tool has been guided by two use cases. One addresses the needs of authors of learning objects, while the other is addresses the needs of learners. We document these use cases in deliverable D2.4. Details of the integration are presented in deliverable D4.1b.

Based on the iterative evaluation of the tool and its validation in use scenarios, we are constantly improving the keyword extractor in response to evaluation and validation results.

There had been some interaction of the keyword development with the corpus development and annotation - some of corpora, i.e. the English, Bulgarian, German, Polish and Portuguese ones, have been extended or re-annotated in response to the evaluation of the keyword extractor output. There had been some interaction between the development process and the integration process performed by WP 4. The integration lead to the revision and simplification of the code. There has been and still is a close interaction between the development and the evaluation and validation of the tools. Evaluation and validation inform the tool development and improvement process.

For the evaluation part, cf. deliverable D2.4. For the validation process, cf. deliverable D5.1b.

3 Improving the keyword extractor

3.1 Improving the runtime performance

The figures for the space and time requirements of the tool which we present below show that the tool can be used in real time. The response time is adequate for a situation where a user requests keywords for a document in order to work on them.

In the following we will present some benchmarks for the performance of the keyword extractor.

We tested it with all available English documents. Assuming the language model (information extracted from all analysed documents) is loaded into memory, the time needed to extract keywords from one document ranges from negligible (a few milliseconds) up to 0.6 seconds, depending on the document size. That has been measured on a 1.4GHz Sempron 2500+ machine with 512MB RAM, with language model for English containing 111 files and over 1300000 tokens.

With language model for Polish (46 files, 463000 tokens) the results were comparable

(mostly below 40ms, the biggest file 0.5 seconds).

The biggest test files were huge (up to 130000 tokens) and it is very unlikely that the system will have to deal with documents of this size in real applications. For documents of rational size the times are below 100 milliseconds.

3.2 Adjusting the linguistic model

Depending on the morphosyntactic information, the tokens encountered in the learning objects are classified into types. Currently there are four types:

- **PLU** - punctuation element are ignored completely
- **FLU** - functional lexical units are not allowed as single-word lexical units, but can appear inside multi-word lexical units; "inside" means "not at the initial or final position". The English word *of* or the Polish word *do* are examples of this type.
- **CMLU** - lexical units which are meaningful in multi word lexical units, even at the beginning or end, but cannot form as a single-word lexical unit. For instance, the word *printed* in *printed copy*, or *visited* in *visited links*.
- **MLU** - meaningful both as single words even alone and as any part of a multi-word lexical unit. The English word *system* is an example for this type.

The classes and their applicability to single/multi word lexical units is depicted in the following table:

	Single-word	Multi-word first/last	Multi-word inside
PLU	-	-	-
FLU	-	-	+
CMLU	-	+	+
MLU	+	+	+

The consequences is that lexical units of the types PLU, FLU and CMLU are not accepted as single word keyword candidates. FLU and CMLU are accepted as parts of multi word keyword candidates. Words of the MLU types are always accepted.

For each language, a classification algorithm was defined. This was a responsibility of the language partners. Meeting their needs, the CMLU class has been added and classification function has been extended to contain (and thus allow to be used in language-specific algorithms) additional token features: base form (*base*) and morphological description (*msd*). This mechanism allows for a more fine-grained classification based on the available information, down to the inclusion / exclusion of individual words.

Provided with new possibilities, the Polish, Portuguese, Dutch, Czech and English language partners adjusted their linguistic models in response to the outcome of the evaluation of the keyword extractor for their languages.

Another feature which needed adjustment was the presentation of multi word keywords. Each keyword is represented by its base form (e.g. **system** if the word appearing in the text is *systems*). Accordingly, a multi word keyword has been represented as the sequence of base forms of its elements ("wirtualne sieć połączenie nauczyciel i uczyć się"). This turned out to be unnatural and therefore hardly acceptable as a keyword for our test users. We therefore decided to represent a multi word keyword by the form that appears most often in the text (for the above example this is "wirtualne sieci połączeń nauczycieli i uczących się", English: *virtual networks of teacher-student connections*).

3.3 Additional features of keyword candidates

Layout feature

The layout of tokens which is encoded in the original documents has been preserved in the BaseXML files and can therefore be used by the extraction tools. The layout information which is preserved is either structural (e.g. the "div"-tag) or typographical (e.g. the "em"-tag). In the LT4ELAna format this information is stored, as an array of values, in the "rend"-attribute of the "tok" element. Here is an example form an English learning object:

```
<tok id="t419" class="word" sp="y" rend="b" ctag="DT" base="a" msd="DT,SG,wh">A</tok>
<tok id="t420" class="word" sp="y" rend="b" ctag="NN" base="caveat" msd="N,SG,proper,vrbl">caveat</tok>
<tok id="t421" class="word" sp="y" rend="b" ctag="IN" base="before" msd="CJ">before</tok>
<tok id="t422" class="word" sp="y" rend="b" ctag="PRP" base="you" msd="PR,2,SG,ACC,rflx,wh">you</tok>
<tok id="t423" class="word" sp="y" rend="b" ctag="VBP" base="begin" msd="V,BASE">begin</tok>
```

For measuring the keywordiness, we consider this layout features relevant which highlight a particular verb. These are the typographical features *italics*, **boldface** and underlining.

For three languages, i.e. English, German and Portuguese, we explored whether highlighted words had a higher probability of being chosen as a keyword. For this purpose, we defined those token as highlighted which had been rendered in italics, boldface or underlined. In the following we present a) the share of highlighted tokens relative to all tokens; b) the share of highlighted tokens marked as keywords by a human annotator relative to the number of all tokens marked as keywords by the human annotator. The following table shows the results:

Language	highlighted tokens (%)	highlighted keyword tokens (%)
English	9.98	18.4
German	8.3	11.7
Portuguese	15	34.5

As can be seen from this table, there is a significantly higher probability of a highlighted word to be selected as keyword, higher than could be expected if highlighted and non-highlighted words would have been selected by chance. This tendency is clearer for English and Portuguese, where this probability is (more than) twice as high.

A qualitative evaluation however shows that there is also a considerable amount of *noise* due to the fact that there are many reasons to highlight a word, e.g. as a subheading or as an important position in a table or as a part of a quotation in another language. So there is not direct correlation between highlighting of a token and its keywordiness. Still, the figures encourage us to take the layout feature into account as an additional keywordiness indicator.

Lexical chains

The use of lexical chains is a way of introducing semantic knowledge - represented by a lexical semantic resource - in the weighting of keyword candidates. This approach has not been declared in the annex as one of the development tasks. The exact contribution of the method cannot be measured beforehand, so this approach is to be considered as an add-on to the core development which fits well with the lines of research followed by the involved partners.

A lexical chain is a sequence of semantically related words spanning an entire text or a part of it. Lexical chains are independent of the grammatical structure of a text. In effect, they are lists of words that capture the cohesive structure of a text. Lexical chains represent the overall topic of a text, if they span the whole text, or otherwise a subtopic of

the text. A word which is a member of a lexical chains can be assumed to be a good representative of the topic of the text which the lexical chain spans, and therefore good candidate for a keyword.

Building lexical chains is one way of complementing the distributional measures for detecting and ranking keyword candidates with a semantically grounded measure. We considered it therefore worthwhile to explore this approach further.

Constructing lexical chains presupposes the existence of a lexical semantic resource like a wordnet or an ontology, from which the lexical semantic relation(s) of a pair of words can be derived, if there is any. The lexical-semantic relation between each pair of members of a lexical chain is an important indicator for the weight of a lexical chain. There are two resources which we will use for our lexical chainer: a) wordnets of several languages; we started with using the German wordnet because it has a well-documented and easy-to-use API; b) the LT4EL domain ontology; an API to access this ontology has just been finished.

To integrate the lexical chainer into the keyword extracting process we have to:

- build the lexical chains for a text from which keywords should be extracted. This part has been implemented by now, using the German wordnet as the lexical resource
- give each individual word a score which is derived from the weight of the lexical chain(s) in which it takes part
- combine the "lexical chain" scores of each word with the distributional score and the layout score calculated for the same word

We are currently experimenting with the different lexical-semantic resources. While a general language lexical-semantic resource like GermaNet does not very well cover the domain specific and therefore important vocabulary of our learning objects, the domain ontology might be too small to generate any useful lexical chain at all, but we have to explore this. In general, a domain-specific ontology is to prefer over a general-language resource.

3.4 Additional distributional statistics

We announced in deliverable D2.2a that we want to follow term burstiness, as introduced by Sarkar and deRoeck (cf. Sarkar & deRoeck 2005), as an alternative way of deriving keywordiness from the distributions of the word. While the other distributional statistics we use model the **inter-document** distribution of words, term burstiness models the **intra-document** distribution. Therefore it seemed to be a promising approach.

We conducted an experiment, using available software (i.e. WinBUGS and JAGS) to calculate the values for 114 documents English documents containing 1356354 tokens. Our investigations revealed that the calculation of term burstiness takes too much time, i.e. 200 terms per second on a single CPU machine. In addition, the available software for the calculation of the statistics turned out to be not reliable enough to be used without human intervention. For two percent of the terms in our test collection the software failed.

It is against the spirit of our approach, which calls for stability and a real time response to a user's request, to add term burstiness to our keywordiness calculation. We therefore abandoned this approach.

The experiment yielded the following results, in comparison to the other distributional measures we have used so far:

Method	TFIDF	RIDF	ADRIDF	TB
Performance	.2473	.1020	.2597	.2633

The results, i.e. a slight improvement of the results, are in line with what Sarkar and deRoeck

report in their paper, i.e. that they can reproduce the results yielded by other distributional approaches. From this single experiment we can draw the preliminary conclusion that there is an upper limit of performance for distributional methods, unless one combines these distributional approaches with semantic knowledge.

3.5 Combining several values in a combined ranking statistics

We want to draw on any information and measure to find and rank the best keyword candidates. We are therefore going to combine heterogeneous information sources. The first two information sources which we will combine are the evidence from distribution and the evidence coming from the use of layout features. Other sources of evidence like lexical chains may follow later on.

We are aiming at combining the best distributional score and the layout score in a way that the combination yields an optimal result. As a benchmark for an optimal result we will use the manually annotated data as training and evaluation data. We train the linear combination algorithm on the training data and afterwards evaluate the combination on the evaluation data set.

We have investigated the use expectation maximization as the statistical method to combine the different scores of keywordiness we have. This method is typically for the combination of n-gram models of different complexity.

An expectation-maximization (EM) algorithm is used in statistics for finding maximum likelihood estimates of parameters in probabilistic models, where the model depends on unobserved latent variables. EM alternates between performing an expectation (E) step, which computes an expectation of the likelihood by including the latent variables as if they were observed, and a maximization (M) step, which computes the maximum likelihood estimates of the parameters by maximizing the expected likelihood found on the E step. The parameters found on the M step are then used to begin another E step, and the process is repeated. Since the range of values differs for the two scores, we first have to normalize them. For every keyword candidate, each measure then suggests a probability of this word of really being a keyword (i.e. a value between 0 and 1).

We aimed at finding the linear combination of two values λ_1 and λ_2 in a form that the sum of these values is 1. Later on, when adding further scores, we use EM to re-estimate the values for λ_i with $i > 2$.

The EM algorithm iterates in two steps

- step 1: go through the training data and performs a partial count computation
- step 2: estimate new values of λ_i which optimizes the probability of correct keywords

The algorithm iterates until the gain over the last step is below a certain threshold. The result of this algorithm are the locally optimal weights for λ_i . We use these values to combine the weights of a keyword candidate into a single value. The combined value then can be used for ranking the keywords.

It turned out that maximizing probability of correct keywords do not necessarily cause better performance.

We investigated several simpler alternatives:

- the *combination of judges*-method: we iteratively add top-ranking candidates from every method and select only those candidates that at least two method agree on
- the *cumulative-evidence*-method: for every candidate, we sum up the ranks assigned by all methods. Then, all candidates are re-ranked according to sum assigned.

We were able to improve F-2 measure by nearly two points which is still less than expected. We will proceed with testing various alternative methods of linear combining the scores for other languages in order to achieve the optimal result.

4 Actions taken in response to user experience

The following improvements of the keyword extractor have been initiated in response to feedback from users:

- Keywords are presented in a way that users see the 10 highest ranked keywords first and can access up to 50 keywords for their selection
- Multi-word keyphrases are presented by the most frequently attested forms instead of the base forms which look sometimes very artificial to users
- Linguistic models have been refined to restrict the number of keyword candidates further

The focus of the third phase of the project will be on improvements which are triggered by the feedback of users from the validation scenarios of WP5.

5 Documentation

Documentation of the keyword extractor goes alongside the development and integration of the tools. Currently, a description of all classes and methods are available as JAVADOC documents. As such, the documentation is easy to maintain. It is targeted to software developers who want to use the tools and integrate them into their applications. The language of documentation is English. The documentation in its current state can be downloaded from the portal server in lasi.

6 Scientific papers on the keyword extractor

The following papers have been written, presented on conferences and published. The full papers are in the appendix. In the following, the bibliographical data are listed, together with a short summary.

- Lothar Lemnitzer, Paola Monachesi: Keyword extraction for metadata annotation of Learning Objects. RANLP 2007 workshop: Natural Language Processing and Knowledge Representation for eLearning Environments. Borovets, 26. September 2007.
 - In this paper, we focus on the evaluation of the keyword extractor results.
- Lothar Lemnitzer, Cristina Vertan, Alex Killing, Kiril Simov, Diane Evans, Dan Cristea, Paola Monachesi: Improving the search for learning objects with keywords and ontologies. Appeared in: Duval, Erik; Klamma, Ralf; Wolpers, Martin (Eds.) Creating New Learning Experiences on a Global Scale. Second European Conference on Technology Enhanced Learning, Lecture Notes in Computer Science , Vol. 4753, pp. 202-216.
 - In this paper we describe, among others, the architecture of the keyword extractor and the distributional measures used.