Monica Gavrila, Walther von Hahn, Cristina Vertan

ManageLex
A Tool for the Management of Complex Lexical Structures

Abstract
This paper describes MANAGELEX, a lexicon management tool, developed at Hamburg University, Natural Language Systems Division. After a general introduction on lexicons, the authors present the architecture and functionality of MANAGELEX. Sections 3 and 4 give information on two of the MANAGELEX modules concerning the choice and the structural organization of the linguistic features in a lexicon.

1 Introduction

In both monolingual and multilingual environments, language resources play a crucial role in preparing, processing and managing the lexical information and knowledge needed by computers. A large variety of computational lexicons was created, leading to a huge amount of different lexical structures and formats. This variety was triggered by differences between languages, differences in purpose and content, and differences in linguistic theory. In the past, numerous small and medium size lexicons were built in projects and became non-reusable later on because of their specific linguistic model or non-standard format.

In order to reduce the work that is done repeatedly in creating lexicons, standard formats and models were created including

- **standard lexicon formats** like EAGLES (*http://www.ilc.pi.cnr/*), MILE (for details see CALZOLARI ET AL. 2003), SALT, etc.
- **standard lexicon models** like GeneLex, Multilex, Parole/Simple (PAROLE/SIMPLE REPORT 1, PAROLE/SIMPLE REPORT 2) etc.

However, for many applications these standards are too complicated (because they try to model everything), and still contain gaps in modeling features of less spoken languages. Sometimes, for projects or evaluations (a series of) smaller lexicons with specific or even changing specifications are needed.

Another problem is the complicated manipulation of the existing lexicons as stand-alone components; either some of them have been produced with acquisition / save tools that may not be maintained any longer or do not have flexible export facilities, or they may contain procedural elements dependent on the host system.

Another problem is the operation of merging lexicons. Especially for less spoken languages, merging several small lexicons developed in different projects is an important step towards the achievement of a computational lexicographic resource for that language. The merging of lexicons is complicated by several factors:

- differences in format and encoding which frequently do not match,
- differences in linguistic categories,
- inconsistencies of values or different granularities.

2 MANAGELEX

General lexical management tools, which help the user to manipulate and validate lexicons, represent an alternative to standardization. Such a tool is MANAGELEX, currently under development at Hamburg University. This tool is not intended for replacing the present standards, but

for managing the already existing lexicons (standard or non-standard).

MANAGELEX is "a generic lexicon management tool" (Vertan/von Hahn 2002) that permits the user to create, read, convert, and combine lexicons. MANAGELEX is also intended to enable the merging of lexicons that do not share common import and export formats. It also enhances the reusability of lexicons created in earlier projects by providing a tool that makes it possible to convert lexicographical data. Its design is format-, language- and platform-independent. Following functionalities are to be supported in MANAGELEX (the GUI is shown in Figure 1):

– reading and saving different encoding formats;
– accessing, creating and transforming different lexicon structures;
– merging of two lexicons either by merging their structure or merging lexical entries.

The main goals of MANAGELEX consist in improving the reusability of lexicons and and facilitating lexicon handling without dictates of a standard format or model.

2.1 Architecture

The MANAGELEX architecture follows the ANSI specification and contains three levels: real word v, model level and meta-model level.

Fig. 1: MANAGELEX GUI Snapshot

Real, distinct objects represent the real world level, i.e. files that consist of the lexicon structure (Structure files: ***StructA***), files that contain the encoded lexicon (Lexicon files: ***DocA***), and lexicon content files (***EntryA***).

The model level consists of four tools:

1. **EditTool** reads, adds or updates entries in a lexicon.
2. **StructTool/LexTool** defines or updates the linguistic specification of a lexicon. This tool is described in detail in Section 4.
3. **EncodTool** decodes lexicon files and encodes lexicon entries into files. The encoding/decoding operation is done according to the specification in EncodMode, or, where it is missing, according to the user specification.
4. **MapTool** merges two lexicons with possibly different linguistic specifications.

The **meta-model level** is composed of three models:

1. **LexMode** is a rich model of possible lexical information. It is described in detail in Section 3.
2. **EncodMode** specifies the data structure of a specific entry and of a specific lexicon. The model is to be built after analyzing several existing models (e.g. OLIF, SALT, etc.).
3. **MapMode** specifies how two lexicons can be mapped. It has to take into consideration mutual gaps, complex categories, etc.

For the moment only StructTool and LexMode are fully implemented.

2.2 Functionality

Following operations can be performed within MANAGELEX: building, reading, and updating a lexicon, and merging two lexi-

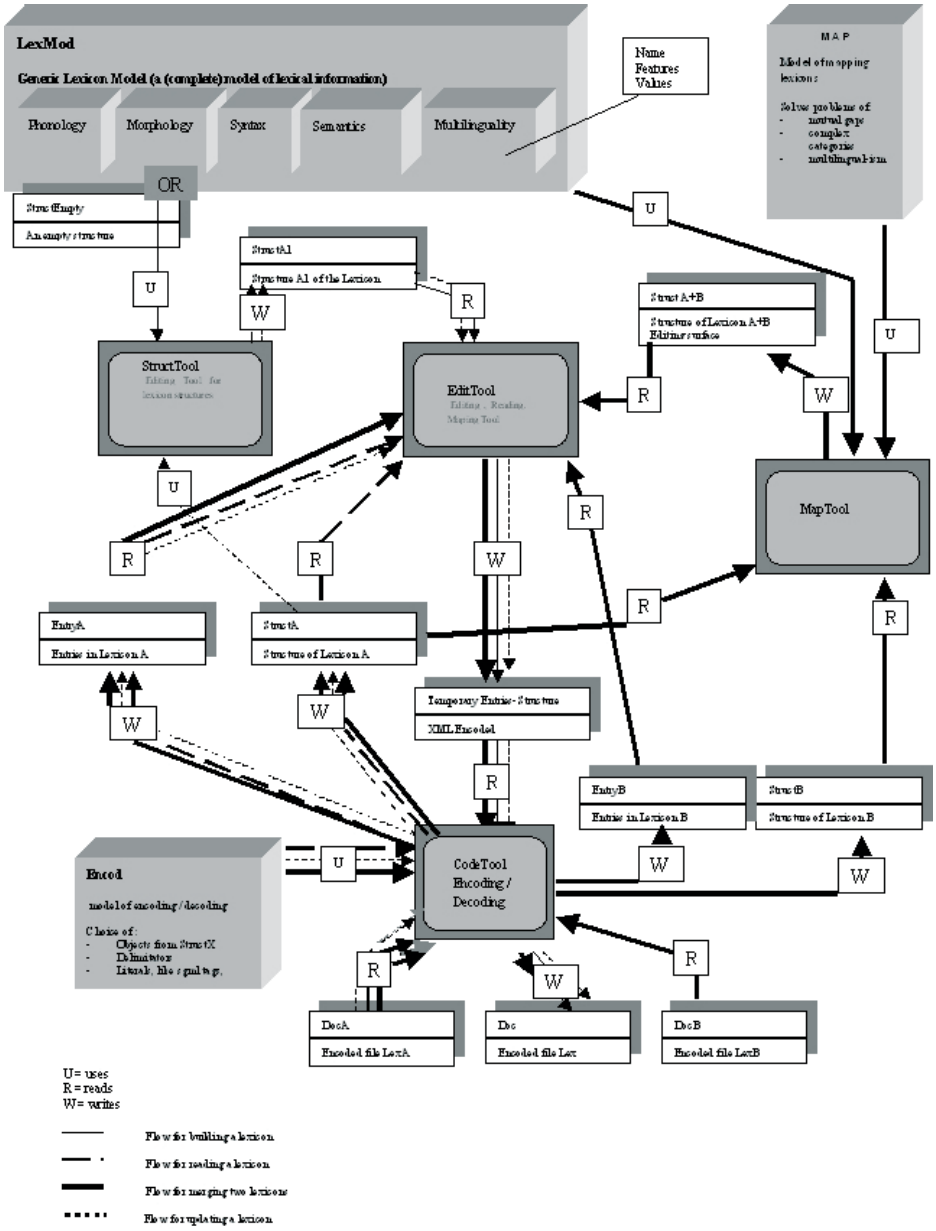

Fig. 2: MANAGELEX architecture and functionality (GAVRILA 2004)

cons. Figure 2 shows the architecture and functionality of MANAGELEX.

The most complex operation is the merge of two lexicons. It involves the use of LexMode, MapMode, EncodMode, and of three tools: the encoding/decoding tool (EncodTool), the mapping tool (MapTool), and the editing/reading tool (EditTool). Let us assume we want to merge Lexicons A and B. First, the encoding/decoding tool provides the two structure files StructA and StructB and two files containing the entries in the lexicon (LexA and LexB).

This operation is performed using EncodMode. The mapping tool uses the two structure files StructA and Struct B, LexMode and MapMode. As output is produced a new structure file which contains all linguistic elements of the two lexicons and in which all possible feature and value overlaps are resolved. The user will solve the overlapping problems if they cannot be resolved automatically. The mapping of the entries from the entry files and the new structure is done by the editing / reading tool. The sequence of these operations is illustrated in Figure 3.

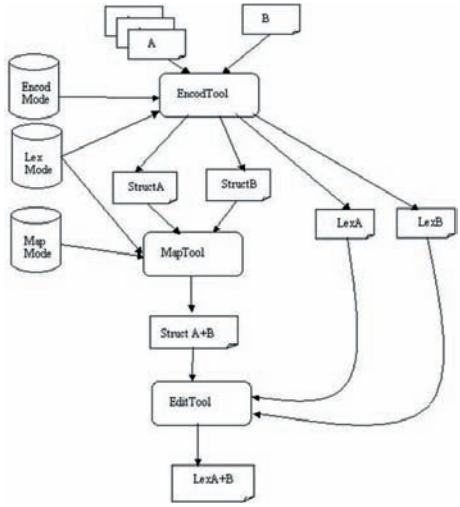

Fig. 3: Flow of operations for the merge of two lexicons in MANAGELEX

3 LexMode– the Linguistic Resource in MANAGELEX

MANAGELEX is structured around three metamodels describing the linguistic information (LexMode), the encoding format (EncodMode) and the mapping between two lexicons (MapMode). In this section we will introduce LexMode – a generic ***lexicon model***, which aims to contain as much lexical information as possible. In this model, linguistic features and their possible values are specified. The model construction is based on the study of more than 12 machine-readable lexicons (e.g. CELEX, MULTEXT, GermaNet, Verbmobil) and of several standard lexicon models (e.g. PAROLE/SIMPLE and MILE). Most of the lexicon formats were analyzed in (Gius 2003). More details on lexicons can be found in (Handke 1995).

In the design of LexMode, we paid special attention to the separation between linguistic data and language data (e.g. examples can be added in the entries, but not in the lexicon structure).

LexMode can be updvated with new linguistic features specific to other languages or linguistic focus. Trying to be a lexicon model, it contains as much information as possible which also implies that no optional grammatical features are included. Because all features have cardinality constraints set on value one, it means that all information should be specified in a lexicon entry. In case this is not needed by the user, the cardinality constraints can be modified, or a new lexicon structure can be created. In order to preserve the generality LexMode also contains no relations between features (as in other lexicon models – e.g. PAROLE/SIMPLE – (Parole/Simple Report 1, Parole/Simple Report 2, Ruimy et al. 1998), but, if required, these can be specified later using StructTool.

The LexMode structure contains following levels of information:

– lexicon information,
– entry information,
– morphological information,
– phonological information,
– syntactical information,
– semantic information,
– multilingual information.

The structure of LexMode is presented in Figure 4.

3.1 Formal Specification of LexMode

In this section we will motivate our choice for the formal specification language used for LexMode, and present the way of describing LexMode in this language (OWL).

Three possibilities were considered for the language specification of LexMode: XML, RDF/RDFS, and OWL.

Due to the availability of manipulating tools, and transparency of the language, **XML** could have been a straightforward solution. A first drawback of this approach was the redundancies, which it can introduce. An example of such redundancy is shown below.

```
<category>
  <cname>Part of Speech</cname>
  <category>
    <cname>Noun</cname>
    <attribute>
      <aname>Gender</aname>
      <value>masculine</value>
      <value>feminine</value>
      <value>neuter</value>
    </attribute>
  </category>
  <category>
    <cname>Adjective</cname>
    <attribute>
      <aname>Gender</aname>
      <value>masculine</value>
    </attribute>
  </category>
</category>
```

We observe that the features and their values for nouns and adjectives, although quite similar, have to be repeated. This problem can be solved through the introduction of parameter variables in the DTD. However the new versions of meta-

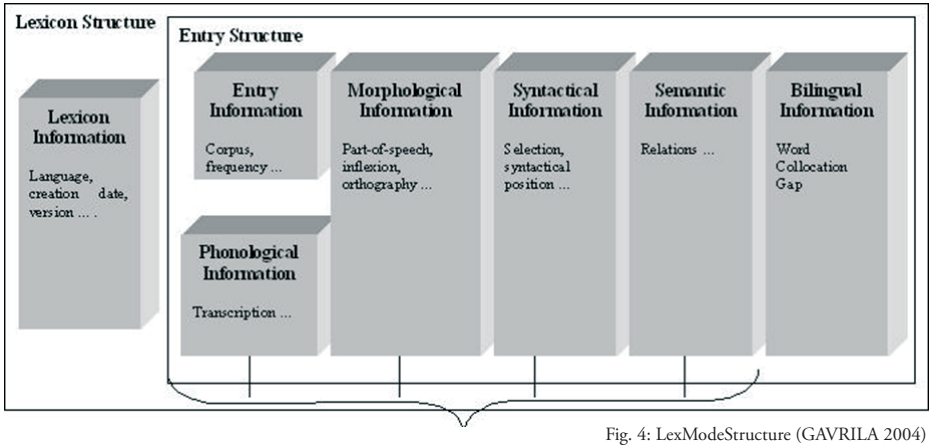

Fig. 4: LexModeStructure (GAVRILA 2004)

languages for XML (XML-Schema) do not preserve any longer this possibility, and its simulation through other mechanisms offered by the language is still cumbersome.

The second possibility – **RDF/ RDFS** (Resource Description Framework Schema) allows data description by means of triples (Subject, Predicate, Object), and for an organization of information in classes and properties. However, it offers only a limited set of relations between classes and/or properties (e.g. class, subclassof, subpropertyof, but no synonymy) and it does not allow restrictions (e.g. cardinality restrictions).

The **Web Ontology Language OWL** (*http://www.w3.org/TR/owl-features*) which is currently based on RDF / XML was powerful enough for the description of LexMod. 15 out of the 48 existing tags were used: **owl:cardinality**, **owl:Class**, **owl:DatatypeProperty**, **owl: maxCardinality**, **owl: minCardinality**, **owl:ObjectProperty**, **owl:onProperty**, **owl:Restriction**, **owl: unionOf**, **rdf: List**, **rdf: type**, **rdfs:comment**, **rdfs:domain**, **rdfs:Literal**, and **rdfs:range**.

Another reason for choosing OWL was to facilitate the integration into the Semantic Web framework (Gavrila/Vertan 2005). Figure 5 gives a short example from the LexModeOWL file.

The example presents what parts of speech we chose in the morphological description.

In LexMode the distinction in describing grammatical features as class or property was done according to the following criteria:

- If a grammatical feature is described using other features, than it is a class.
- If there are relations between classes/features, then an object property is used.

```
<owl:ObjectProperty
     rdf:ID="hasPartOfSpeech">
<rdfs:domain
 rdf:resource="#MorphologicalInfo"/>
<rdfs:range>
<owl:Class>
<owl:unionOf
 rdf:parseType="Collection">
<owl:Class rdf:about="#Verb"/>
<owl:Class rdf:about="#Noun"/>
<owl:Class rdf:about="#Numeral"/>
<owl:Class rdf:about="#Adjective"/>
<owl:Class rdf:about="#Pronoun"/>
<owl:Class rdf:about="#Determiner"/>
<owl:Class rdf:about="#Article"/>
<owl:Class rdf:about="#Conjunction"/>
<owl:Class rdf:about="#Preposition"/>
<owl:Class rdf:about="#VerbParticle"/>
<owl:Class rdf:about="#Particle"/>
<owl:Class rdf:about="#Adverb"/>
</owl:unionOf>
</owl:Class>
</rdfs:range>
</owl:ObjectProperty>
```

Fig. 5: Example from the LexModeOWL file

We also mark different literals and numbers when working with data type properties, as this is very useful for the merging operation.

The LexMode OWL encoding has 34 elements, 88 data type properties, and 23 object properties. In the table below we give some of the LexMode classes and properties.

The above organization of LexMode is flexible enough and fits into the MANAGE-LEX schema. This means that, apart from the role that LexMode is playing for the StructTool (starting point in creating a new structure), it also helps the MapTool in merging two lexicons structures.

The operations that can be done on linguistic categories are: adding, deleting, renaming, merging, and splitting. For example if one lexicon structure contains the verb category with the

Class	Property
LexiconStructure	hasLexiconInfo, hasEntryStructure
LexiconInfo	lexiconName, language, version, creationDate, modificationDate, copyright
EntryStructure	hasEntryInfo, hasMonolingualStructure, hasBilingualStructure
EntryInfo	corpus, frequency, workingState, termStatus, generationType, registeredEntry, refID, source
MonolingualStructure	hasMorphologicalInfo, hasPhonologicalInfo, hasSyntacticalInfo, hasSemanticInfo
BilingualStructure	toLanguage, toLexicon, hasCorrespondences
MorphologicalInfo	HasPOS, etc.
PhonologicalInfo	phoneticTranscription, terminalDevoicing, accents, audioFile
SyntacticalInfo	hasSelection, syntacticPosition, special
SemanticInfo	hasRelations, ontologyTypes, semanticFeatures, prototype, thematicRoles,

Table 1: Some classes and properties in LexMod

property transitivity and in the new structure there should be two different categories: transitive verb and intransitive verb, the category from the first structure is split into two different categories and the transitivity property is deleted.

4 Describing the Linguistic Structure

The structure tool (StructTool) allows the user to define the lexicon structure according to the particular application requirements. It allows the user to add, delete, merge, split, rename or select elements/grammatical features and create the needed lexicon structure (see Figure 6).

Following operations are implemented within StructTool:

– reading LexMod,
– selecting categories and their values and/or ranges,
– defining new categories,
– updating values of existing categories,
– defining the structure of a lexicon,
– calling EditTool.

StructTool reads LexMode (or other OWL encoded lexicon structure files), generates a GUI that supports selections and editing (Add, Delete, Merge, Split, Rename operations on grammatical features) and saves a new StructX lexicon structure file.

As an example of an operation that can be performed with StructTool we present how a property is updated by renaming. If in a certain moment the user wants to rename an existing property, this can be easily done from the graphical interface. The process of renaming itself is a little bit more complicated, because the new name has to replace the old name in the whole lexicon structure – everywhere there is a reference to it -, so that lexicon structure consistency is kept.

5 Conclusions and Further Work

In this paper we presented MANAGELEX, a generic lexicon management tool that can be regar-

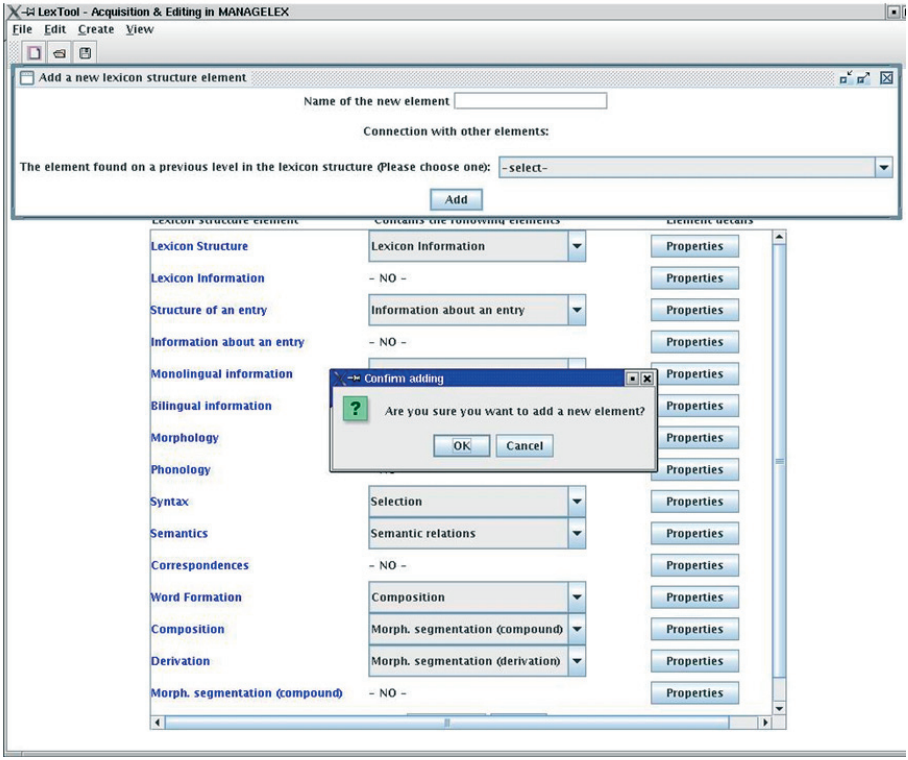

Fig. 6: StructTool GUI

ded as a possible alternative to lexicon standardization. The tool is flexible and easy to use also for non-specialists. For the moment, only European languages are modeled.

We are currently working at the implementation of the other tools and models of the MANAGELEX system as well as preparing ready-to-start configurations for widely used standards like PAROLE/SIMPLE and MILE. We are also planning an updating process for LexMod, by including important changes (adding operation) in structure files into LexMode. Further on, we would like to extend our linguistic model to other types of languages. Information regarding the last version of the system can be found at *http://nats-www.informatik.uni-hamburg.de/view/MainManageLex*.

References

Calzolari, N., Bertagna, F., Lenci , A., Monachini, M. (2003). "Standards and Best Practice for Multilingual Computational Lexicons &MILE (the Multilingual ISLE Lexical entry)". Deliverable D2.2-D3.2 ISLE Computational Lexicon Working Group, *http://www.ilc.cnr.it/EAGLES96/isle/clwg_doc/ISLE_D2.1-D3.1.zip*.

Gavrila, M. (2004). "LexMod and LexTool – Lexical Model, Acquisition and Editing in MANAGELEX", Diploma Thesis, Hamburg University, Computer Science Faculty, R36887.

Gavrila, M., Vertan C. (2005)."MANAGELEX and the Semantic Web". OntoLex Workshop Proceedings, IJCNLP-05, Jeju, South Korea, October 2005.

Gius, E. (2003). "Vergleich maschinenlesbarer deutscher Lexika nach linguistischem Inhalt, Wertebereichen und Kodierung", Diploma thesis, University of Hamburg, manuscript.

Parole/Simple Report 1. "Report on the Syntactic Layer", *http://www.ub.es/gilcub/SIMPLE/reports/parole/parole_syn/parosyn.html* .

Parole/Simple Report 2. "Report on the Morphological Layer", *http://www.ub.es/gilcub/SIMPLE/reports/parole/parole_morph/paromor.html* .

Handke, J. (1995). "The Structure of the Lexicon – Human versus Machine". Berlin-NewYork: Mouton de Gruyter, .

Ruimy, N., Corrazzari, O., Gola, E., Spanu, A., Calzolari, N., Zampolli, A. (1998). "The European LE-PAROLE Project and the Italian Lexical Instantiation". In: Proceedings of the ALLC/ACH, 1998, Lajos Kossuth University, Debrecen, Hungary, July 1998, pp. 149-153:

Vertan, C., von Hahn, W. (2002). "Towards a Generic Architecture for Lexicon Management". In: Proc. LREC-2002. Third International Conference on Language Resources and Evaluation. Las Palmas de Gran Canaría, May 2002.

von Hahn, W. (2005). "Merging Computer-Readable Heterogeneous Terminological Material". In: Proceedings of the XVth European Symposium of Languages for Special Purposes (LSP'05), Bergamo.