Instituto de Engenharia de Sistemas e Computadores
Investigação e Desenvolvimento em Lisboa
Laboratório de Sistemas de Língua Falada

Creating and Maintaining Multi-purpose Lexical Knowledge

Natural Language Engineering on a Computational Grid POSC/PLP/60663/2004

— INESC-ID Lisboa Tech. Rep. 26/2006 —

Ricardo Ribeiro, David Martins de Matos, Bruno Oliveira, Carlos Pona, Luísa Coheur
L^2F – Spoken Language Systems Laboratory
INESC ID Lisboa, Rua Alves Redol 9, 1000-029 Lisboa, Portugal
{rdmr,david,bfso,cmpo,lcoheur}@l2f.inesc-id.pt

This report addresses the problem of maintaining linguistic data collections adequate to the needs of different applications. We posit that when developing NLP applications, one has to manage not only the software development process, but also the linguistic data: handling them separately will reduce the complexity of the process as a whole, thereby increasing the overall quality. Data consistency is also improved since there is only one collection to manage. We present two illustrative experiments that benefitted from the separation of data from processing, and from having a strong linguistic data management system: extended coverage and enhanced capabilities from integrating different data collections with similar levels of description; versatile configuration capabilities; multilinguality through the use of the repository in combination with language independent processing techniques; rapid development; and increased modularity without sacrificing compatibility.

This revision: January 30, 2008

Creating and Maintaining Multi-purpose Lexical Knowledge NLE-GRID – POSC/PLP/60663/2004

Ricardo Ribeiro, David Martins de Matos, Bruno Oliveira,
Carlos Pona, Luísa Coheur

L^{2}F – Spoken Language Systems Laboratory
INESC ID Lisboa, Rua Alves Redol 9, 1000-029 Lisboa, Portugal
{rdmr,david,bfso,cmpo,lcoheur}@l2f.inesc-id.pt
http://www.l2f.inesc-id.pt/

Abstract. This report addresses the problem of maintaining linguistic data collections adequate to the needs of different applications. We posit that when developing NLP applications, one has to manage not only the software development process, but also the linguistic data: handling them separately will reduce the complexity of the process as a whole, thereby increasing the overall quality. Data consistency is also improved since there is only one collection to manage. We present two illustrative experiments that benefitted from the separation of data from processing, and from having a strong linguistic data management system: extended coverage and enhanced capabilities from integrating different data collections with similar levels of description; versatile configuration capabilities; multilinguality through the use of the repository in combination with language independent processing techniques; rapid development; and increased modularity without sacrificing compatibility.

1 Introduction

The use of natural language processing (NLP) tools in daily life is becoming ubiquitous if sometimes discreet: spell checkers, summarization tools, and dialogue systems, just to name a few, are used often in our everyday routine. These tools make different demands on the underlying NLP modules supporting them. Yet, existing modules – developed in different contexts – use different data in different formats, making knowledge interchange (interoperability) difficult. This is the problem addressed by this work: the maintenance of linguistic data collections adequate to the needs of different applications. When developing NLP-based applications, one has to manage not only the software development process, but also the linguistic data. Handling both separately will certainly reduce the complexity of the whole process. Moreover, from the data point of view, several other advantages emerge from this separation, e.g. data reuse; easy adaptation of a tool to different possibilities of use (for example, providing smaller dictionaries for handheld devices); improvement of the performance of a tool through integration of different data collections in a repository; increase of the quality and consistency of the data by reducing to one the managed data collections.

It is well known (and generally accepted) that linguistic descriptions should not be hard-coded in the implementation of the applications that will use them. Nevertheless, we argue that it is not enough to keep linguistic descriptions apart from implementation: linguistic descriptions should remain at a declarative level, completely apart from processing information.

As a first example, consider a formalism for writing dependency rules between chunks. The PP-attachment problem represents so much ambiguity, that, when writing those dependencies one may be tempted to limit them to a certain distance (supposing that the formalism allows it). Unfortunately, in natural languages there is no empirical evidence of those limits. As so, although we will simplify the application, we are making linguistic descriptions non-declarative, as we merged them with processing information. Notice that there is nothing wrong about limiting those distances: it is simply the case that information should not be mixed with the linguistic descriptions.

Consider now a simple natural language processing system, where declarative rules have to be applied in the order they appear. The person responsible for writing those rules has to be aware of that (processing) fact which can influence his decisions. Moreover, if those rules are going to be used in other application, that fact has also to be taken into consideration. In this situation, even although the majority of the rules with probably have no relation between them, it is not obvious if one can partially use a subset of them, as it is not known whether there is a real relation between them or not (the fact that one rule is written before or after another rule will create a possible non-existing relation). Once again, the problem arises: processing information is merged with declarative information.

Our lexical repository [5,11] was developed to facilitate management and maintenance of linguistic data, allowing the representation of rich multi-level language-related information. The repository was devised to be independent from any applications. This means that adequate data must be selected from it, in accordance with concrete situations.

We carried out two experiments that use the data from the language resources repository to solve known natural language problems, both pointing out the advantages of separating knowledge from processing: the first, concerning morphological generation, demonstrates close interaction with the repository; the second, concerning morphological analysis, takes advantage of the previously defined interaction to show the use of repository data in an application not directly connected to the repository.

This document is organized as follows: Section 2 presents the language resources repository; and Sections 3 and 4 present the experiments. The document concludes with remarks about the experimental results. approach.

2 The Lexical Repository

As stated above, close development of linguistic data and natural language processing tools entails an undesirable dependency between them. Mutual dependency between data and tools makes reuse difficult due to various incompatibilities: at the description level (e.g., tag incompatibility in data used by morphosyntactic taggers); at the level of what entities are described, as some descriptions may describe objects missing from other descriptions; at the format/representation level (XML vs. tabular data); and at the expressiveness level ("United States of America" as a single entity vs. composition of separate entities).

Our lexical repository is a way to solve these incompatibilities and increment/improve data reuse: it is capable of storing data belonging to different paradigms and originally encoded in different formats, covering several description levels (e.g., morphology, syntax, semantics). Figure 2 shows several examples of data integrated or to be integrated in the repository. Moreover, as a dynamic repository, it provides for smooth extension of its description capabilities and coverage, minimizing the impact on the content.

The repository is based on a canonical model for storing/manipulating data, and a dynamic maintenance model for keeping the data model synchronized with new data developments. A canonical model has distinct advantages: it is easier to maintain and document a single format than multiple different ones; the effort dedicated to maintenance tasks is concentrated, possibly further improving them; it allows for deeper understanding of data, which in turn facilitates reuse (the reverse would require a user to understand multiple models). This model is based on existing large coverage models, i.e., we seek a broad linguistic description that crosses information from various levels, including but not limited to morphology, syntax, and semantics. Examples of existing models are the one resulting from GENELEX project or the current LMF (Lexical Markup Framework).

is described in UML/XMI [4]: this model was used to automatically generate creation, load, and access code.

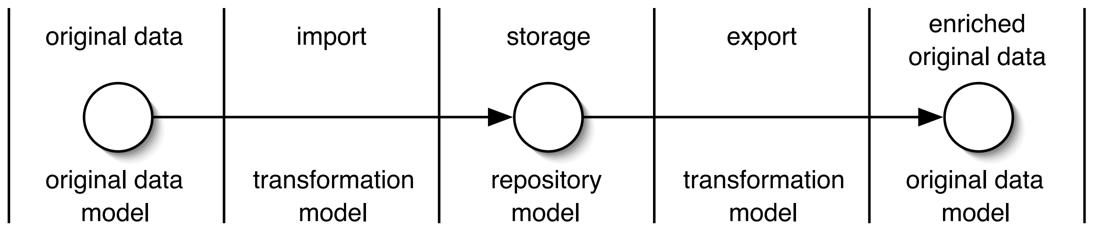

Fig. 1. Models

In addition to its basic storage capabilities, the repository acts as a bridge between data models: besides the definition of a large coverage canonical model, it accounts for transformations between this model and external models. These transformation models are important, since applications may precede the repository and require proprietary formats. By using the repository, existing applications may also benefit from the recommended separation between data and processing. Figure 1 shows the relations between the internal and external data models.

The repository is a more general solution to the problem of data reuse, integration, management, and maintenance, contributing to the enrichment of the data used by new and existing applications (enriching the applications themselves). Separating data and processing allowed the use of the repository for data management (and improvement) and, on the other hand, stimulated the separation itself.

3 The First Experiment: morphological generation

Morphological generation is the process whereby word forms are created based on a root form and a set of morphological features. This type of tool is useful in language generation applications but may have other uses, suach as creation of word lists.

The first tool to take advantage of the lexical repository is Monge. Figure 3 shows an example output.

3.1 Monge and the Lexical Repository

Monge is a special tool in that intrinsically depends on the repository's internal data model. It is, however, completely independent from its contents. These aspects impact the tool in different ways: the former implies that whenever the model is changed so must the tool; the latter states that whatever the repository's content, the tool remains useful. It is this last observation that empowers the approach: changes in repository structure are expected to be few and far apart, while changes in data are expected to be frequent.

Note that while Monge depends on the model, it does not depend on the concrete implementation. In particular, it can work without change with different database engines. Monge's output is independent of any particular support and is well suited for abstracting repository dependencies.

3.2 Morphological Generation

Monge takes as inputs a root form and an optional set of features. These features cover aspects such as categorization and subcategorization, as well as any other set of features deemed of interest to be described. Currently, Monge generates only inflected graphical forms.

The root form is used to determine the regular inflection bases for the word in question. These bases, along with the set of features given as argument will produce the resulting inflected form. Production of the inflected form is carried out using inflection paradigms, selected from the repository's collection according to the given root.

If specific features are selected in the input, then only the inflected forms corresponding to those features will be created. Otherwise, Monge generates all forms for all unrestricted features, including morphosyntactic categories.

Afterwards, this information is converted into an XML tree. For validation and structuring purposes, an XML Schema (XSD) is associated with the output tree. Monge is currently implemented in Perl and uses the general DBI interface for database access. XML output is perform using Xerces-p.

4 The Second Experiment: XISPA

XISPA is a morphological analyzer based on finite state automata. As most other analyzers, given an input word, XISPA will verify whether it exists and, if that is the case, returns the corresponding set of attribute/value pairs describing its morphology. As an example, consider word *algo*: XISPA's processing results are as shown in table 1.

To perform the analysis, XISPA uses the Monge-generated information, in the form of finite state automata. The underlying idea is to facilitate the construction of this kind of tool from large-coverage dictionaries.

PAROLE [10]

```
<mus id="r592" naming="algo" gramcat="adverb"
     autonomy="yes" synulist="usyn23987 usyn23988">
  <gmu range="0" reference="yes" inp="mfgr1">
    <spelling>algo</spelling>
  </gmu>
</mus>
<mus id="pi1" naming="algo" autonomy="yes"
     gramcat="pronoun" gramsubcat="indefinite"
     synulist="usyn23320">
  <gmu range="0" reference="yes" inp="mfgempty">
    <spelling>algo</spelling>
  </gmu>
</mus>
<ginp id="mfgr1" example="abaixo">
  <combmfcif combmf="combtm0">
    <cif stemind="0">
      <removal/><addedbefore/><addedafter/>
    </cif>
  </combmfcif>
</ginp>
<ginp id="mfgempty" comment="empty Mfg">
  <combmfcif combmf="combtmempty">
    <cif stemind="0">
      <removal/><addedbefore/><addedafter/>
    </cif>
  </combmfcif>
</ginp>
<combmf id="combtmempty"/>
<combmf id="combtm0" degree="positive"/>
```

SMorph [3]

```
algo            /pr_i/s/GEN:*/pri .
```

LUSOlex [13]

```
Adv191 <algo> ADVÉRBIO - FlAdv2 <algo>
Pi1 <algo> PRONOME INDEFINIDO - <algo>
FlAdv2  <abaixo>
        __P_____ 0       <><>
$
```

EPLexIC [6]

```
algo/R=p/"al~gu/algo
algo/Pi=nn/"al~gu/algo
```

Fig. 2. Different lexicon-dependent descriptions for word *algo* (*something*)

Table 1. Processing results for *algo*

lemma	**cat**	**subcat**	**degree**
algo	ADVERB	-	POSITIVE
algo	DETERMINER	INDEFINITE	n/a
algo	PRONOUN	INDEFINITE	n/a

```
$ ./monge.pl -x ser number singular gender masculine
```

```
<?xml version="1.0" encoding="UTF-8" standalone="no" ?>
 <monge:lemma
    xmlns:monge="http://www.l2f.inesc-id.pt/~bfso/monge.xsd"
    value="ser">
   <morph cat="NOUN" subcat="COMMON">
     <form value="ser">
       <feature given="yes" name="number" value="SINGULAR"/>
       <feature given="yes" name="gender" value="MASCULINE"/>
     </form>
   </morph>
   <morph cat="VERB" subcat="MAIN">
     <form value="sido">
       <feature given="yes" name="number" value="SINGULAR"/>
       <feature given="no" name="mood" value="PARTICIPLE"/>
       <feature given="yes" name="gender" value="MASCULINE"/>
     </form>
   </morph>
 </monge:lemma>
```

Fig. 3. Example execution of the Monge application for root *ser* (to be) without categorization restrictions. Note that the forms for all categories and subcategories are generated

Nowadays, the morphological analysis process is usually carried out using finite automata or transducers. In this area, the work of Koskenniemi [8,7] is a fundamental reference. Finite automata are especially interesting because their analysis time is $O(n)$, where n is the word's length, and it is independent from the total number of words in a dictionary. Attractive these features may look, they come at a price, although one not currently too high, and continuously becoming less of a limitation in modern devices: memory usage. As for the type of information used in current morphological analysis systems, two general choices are available: either a paradigm- or a rule-based approach. The actual choice may be defined by the language in question, i.e., whether it is more amenable to be processed more successfully using one the approaches. For European languages, and for the Portuguese language in particular, the paradigmatic approach is a common one. In fact, several large-scale projects, both past [1,10,12] and present [2], still use this approach to describe data.

Although some of the current systems, built according to traditional precepts of what should constitute a morphological analyzer, are more complex, they suffer from problems related to maintenance, both of their code base and of the data they use. While we also have to maintain lexical data, the collection is used for a wide variety of tasks and maintenance is amortized by the overall benefits brought to various fields. Compare this scenario with maintaining various data sets, one for each tool used in a given context. In addition to the above, we must also mention the simplicity associated with our approach: it is easily adapted to other contexts or to different data sets.

4.1 Constructing the Analyzer

Figure 4 depicts XISPA's construction process: XML data produced by Monge from the lexical database are transformed so that they specify a finite automaton containing all known forms for all the lemmas of a given language's lexicon. Note that this automaton does not do anything more complex than assigning an integer number to each token it is presented with: the transformation of the automaton's output into a set of morphological features characterizing each token is done separately. Furthermore, the automaton does not encode any ambiguity which may be associated with any of the word forms it recognizes, i.e., each form corresponds to only one output. This option simplifies the use of the finite state machine. Ambiguity is encoded in each of the selected feature sets: each may be subdivided according to each different classification possibility. This is equivalent to encoding ambiguity in the automaton proper, but simplifies it greatly, both in construction and use.

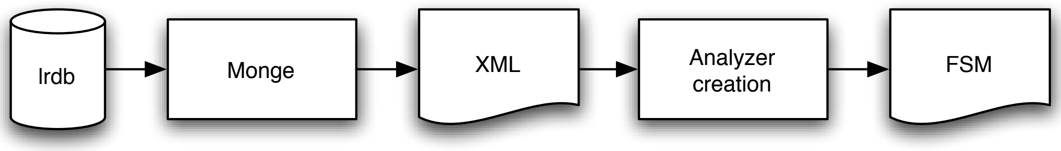

Fig. 4. XISPA's construction process (finite state automaton)

It should be noted that the construction process does not allow for some aspects traditionally handled along with morphological analysis, namely, input tokenization and capture of regularities in the input (e.g. processing of numeral strings). Nevertheless, these aspects do not have a negative impact on the overall functionality: these tasks may be handled efficiently by dedicated tools. Tokenization may be handled by a pre-processor and abstract rules may be handled by pre- and post-processing steps (see figure 5). In our group, in fact, the latter already exists to perform a series of rewriting tasks: these can now be expanded to cover the aspects mentioned above.

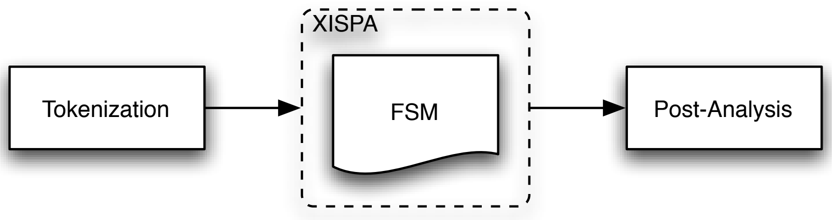

Fig. 5. Using XISPA in a morphological analysis chain: the tokenizer segments the input data; the post-processing module allows to analysis to be complemented and/or supplemented

Construction of a morphological analyzer using this method was exceedingly simple (our first prototype took less than a day to build). So simple, in fact, that we built several prototypes, in order to study the advantages of each approach: the prototypes ranged from naïf, using the compiler construction tool `flex`,[1] to the full fledged AT&T *fsmtools* [9].

5 Concluding Remarks

Although difficult to evaluate quantitatively, it is possible to make a critical analysis of the work presented here. The success of this approach can be assessed by verifying whether the resulting tools produce comparable (or better) results than existing ones and whether the previously mentioned advantages (data reuse, flexibility, reduced complexity of the development process) were effectively achieved.

The repository itself envisions a more general solution to the problem of data maintenance: it facilitates data reuse and integration, and accounts for the evolution of both data and structure.

[1] `http://www.gnu.org/software/flex/`

In what concerns our experiments, both morphological generation and analysis benefitted from the separation of the data from the processing, and from having a strong linguistic data management system: (i) extended coverage and enhanced capabilities (generation and tagging of phonetic information) from the integration of different data collections with similar levels of description; (ii) versatile configuration capabilities; (iii) multilinguallity through the use of the repository (designed to be language independent) in combination with language independent processing techniques; (iv) rapid development (especially in the morphological analysis experiment, even though the tool does not perform tokenization); and, (v) increased modularity (focus on the data, not on the processing).

References

1. Marie-Hélène Antoni-Lay, Gil Francopoulo, and Laurence Zaysser. A Generic Model for Reusable Lexicons: the Genelex Project. *Literary and Linguistic Computing*, 9(1):47–54, 1994. Oxford University Press.
2. Sue Atkins, Nuria Bel, Francesca Bertagna, Pierrette Bouillon, Nicoleta Calzolari, Christiane Fe llbaum, Ralph Grishman, Alessandro Lenci, Catherine MacLeod, Martha Palmer, Gregor Thurmair, Marta Villeg as, and Antonio Zampolli. From Resources to Applications. Designing the Multilingual ISLE Lexical Entry. In *Proceedings of the Third International Conference on Language Resources and Evaluation (LREC 2002)*, Las Palmas de Gran Canária, 2002.
3. S. Aït-Mokhtar. *L'analyse présyntaxique en une seule étape*. Thèse de doctorat, Université Blaise Pascal, GRIL, Clermont-Ferrand, 1998.
4. Grady Booch, James Rumbaugh, and Ivar Jacobson. *The Unified Modeling Language User Guide*. Addison-Wesley Longman, Inc., 1999. ISBN 0-201-57168-4.
5. David Martins de Matos, Ricardo Ribeiro, and Nuno J. Mamede. Rethinking Reusable Resources. In *Proceedings of the Fourth Language Resources and Evaluation Conference – LREC 2004*, pages 357–360, Lisboa, Portugal, 2004. ELRA – European Language Resources Association.
6. Luís Caldas de Oliveira. EPLexIC – European Portuguese Pronunciation Lexicon of INESC-CLUL. documentation, n.d.
7. Kimmo Koskenniemi. Two-level model for morphological analysis. In Alan Bundy, editor, *Proceedings of the Eighth International Joint Conference on Artificial Intelligence*, pages 683–685, EUA, 1983.
8. Kimmo Koskenniemi. Two-level morphology: A general computational model for word-form recognition and production. *Publications*, 11, 1983.
9. Mehryar Mohri. Finite-State Transducers in Language and Speech Processing. *Computational Linguistics*, 23(2), 1997. http://www.research.att.com/sw/tools/fsm/.
10. PAROLE. Preparatory Action for Linguistic Resources Organisation for Language Engineering, 1998.
11. Ricardo Ribeiro, David Martins de Matos, and Nuno J. Mamede. How to Integrate Data from Different Sources. In *A Registry of Linguistic Data Categories within an Integrated Language Resources Repository Area (LREC 2004)*, Lisboa, Portugal, 2004. ELRA – European Language Resources Association.
12. SIMPLE. *Semantic Information for Multifunctional Plurilingual Lexica*, 2000.
13. Luzia Wittmann, Ricardo Ribeiro, Tânia Pêgo, and Fernando Batista. Some Language Resources and Tools for Computational Processing of Portuguese at INESC. In *Proceedings of the Second International Conference on Language Resources and Evaluation (LREC 2000)*, pages 347–350, Athens, Greece, May/June 2000. ELRA – European Language Resources Association.