Natural Language Engineering 3(4): 317–345. Printed in the United Kingdom

ARIES: A lexical platform for engineering Spanish processing tools†

JOSÉ M. GOÑI, JOSÉ C. GONZÁLEZ
E.T.S.I. Telecomunicación, Universidad Politécnica de Madrid,
28040 Madrid, Spain
e-mail: `jmg@mat.upm.es`, `jcg@gsi.dit.upm.es`

ANTONIO MORENO
Dept. de Lingüística, Universidad Autónoma de Madrid,
Cantoblanco, Madrid, Spain
e-mail: `sandoval@lola.lllf.uam.es`

(*Received 20 November 1996. Revised 26 February 1997*)

Abstract

We present a lexical platform that has been developed for the Spanish language. It achieves portability between different computer systems and efficiency, in terms of speed and lexical coverage. A model for the full treatment of Spanish inflectional morphology for verbs, nouns and adjectives is presented. This model permits word formation based solely on morpheme concatenation, driven by a feature-based unification grammar. The run-time lexicon is a collection of allomorphs for both stems and endings. Although not tested, it should be suitable also for other Romance and highly inflected languages. A formalism is also described for encoding a lemma-based lexical source, well suited for expressing linguistic generalizations: inheritance classes, lemma encoding, morpho-graphemic allomorphy rules and limited type-checking. From this source base, we can automatically generate an allomorph indexed dictionary adequate for efficient retrieval and processing. A set of software tools has been implemented around this formalism: lexical base augmenting aids, lexical compilers to build run-time dictionaries and access libraries for them, feature manipulation libraries, unification and pseudo-unification modules, morphological processors, a parsing system, etc. Software interfaces among the different modules and tools are cleanly defined to ease software integration and tool combination in a flexible way. Directions for accessing our e-mail and web demonstration prototypes are also provided. Some figures are given, showing the lexical coverage of our platform compared to some popular spelling checkers.

1 Introduction

The lexical framework presented in this article was developed because of the lack of computerised lexical resources for the Spanish language at the time of its conception.

---

† This work has been supported in part by the Spanish *Plan Nacional de I+D*, through the project TIC91-0217C02-01, *ARIES: An* **AR***chitecture for Natural Language* **I***nterfac***ES** *with User Modelling.*

The situation of Spanish language technology is far behind most of its European counterparts. Even now, only a few dictionaries are publicly available in electronic format[1], and none of them are in a machine tractable form suitable for building the lexical component for NLP applications. This situation is especially worrying if we take into account the spread of Spanish around the world and, therefore, the potential market for Spanish processing tools.

The work described here constitutes the lexical platform of the ARIES project. This platform includes:

Morphological model: this model, presented in section 3, accounts for the inflectional morphology of Spanish. It is based on feature unification and its rule component, while very small, is bidirectional (valid for analysis and generation). The most salient feature of this model is its linguistic rigour, that guarantees neither over-acceptance nor over-generation.

Lexical formalism: this is used to encode the entries in a Source Lexical Base (SLB). These entries include lemmas (main form of words), classes (feature bundles to be incorporated in lemmas through multiple default inheritance), morphemes, allomorphy rules (to generate different stems for groups of forms corresponding to the same lemma) and lexicalized entries (for closed categories and very irregular or auxiliary words). The formalism, described in section 4, also permits the declaration of feature values and structures (useful for checking the consistency of an SLB). Finally, the lexical formalism includes mechanisms for specifying declaratively how to derive an object dictionary for a particular purpose from the SLB by means of rules for manipulating the feature bundles attached to entries, filtering, adding or modifying branches to the tree-shaped feature bundles.

Tools: a set of tools have been developed to automate the production and use of dictionaries. These include (section 5) aids for the lexicographer (verb classifier), for the production of object dictionaries (dictionary compiler), for efficient access from any application program (dictionary interface), for morphological analysis (using a *chart parser* with unification). Other tools more or less linked to the lexical platform, have been developed at our site, such as tokenizers, an stochastic tagger (in cooperation with Universidad Autónoma de Madrid) and a system for learning verbal subcategorization frames.

Lexical resources: Regarding lexical resources, a large lexicon (nearly 38,000 entries) has been derived from several sources: corpora and other collections of Spanish texts, machine readable dictionaries, etc. (section 4.7). Finally, the whole platform has been evaluated for spell checking purposes in comparison with commercial and public domain tools (section 6). Additionally, a demonstrator has been implemented for the public evaluation of the platform. The demonstrator permits the spell checking of texts and the morphological analysis of individual words. It is accessible either through e-mail or WWW browsers.

---

[1] For instance, the first electronic version of the *Diccionario de la Real Academia Española* (RAE 1992), the reference dictionary for the Spanish language, has been released in September, 1995.

2 Design criteria and system architecture

Spanish, as well as some other Romance languages such Catalan, Portuguese, French or Italian, is a highly inflected language, so, for any serious natural language processing application built for it, an account of morphology is needed, or at least of inflectional morphology[2]. This is true not only for reducing the size of the lexicon to a manageable level, but also for capturing the linguistic fact that different entries (word forms) are strongly related. The morphological model presented is based on two basic principles:

Empirical rigour: all and only correct forms are analysed and generated, whether regular or not; gaps in verb paradigms are observed; suppletive forms are considered valid, and so on. It is important to stress that it neither overgenerates nor overaccepts.

Simplicity and generalization: a really straightforward rule component is employed by the morphological model that captures the generalization of the combination of a stem and an ending to form an inflected word.

We have designed a language for lexical knowledge representation whose design has been strongly influenced by such a model. The main features of this language are:

Expressiveness: all the information needed in the lexical database can be expressed in a structured way. Linguistic generalizations are captured by grouping related entries in lemmas, or by using the mechanism of information inheritance. The information related to a lemma is structured as an attached tree-shaped feature bundle.

Versatility: different applications may have different lexical interfaces, depending on programming languages. In our approach translation to other representation formats and languages is easily done in a non-ambiguous way.

Economy of expression: the syntactic overload needed to structure the information has been reduced to a minimum, without endangering either the expressive ability or the non-ambiguity of the syntax of the formalism. However, readability is not degraded, since source lexical databases are intended to be edited by hand using any text editor.

Non redundancy: redundant information is kept to a minimum by exploiting default inheritance and the notational abbreviations included, such as value disjunction.

Efficiency: the platform has been developed from a strong practical orientation. Implementation efficiency issues, both in terms of access and storage, have been considered of paramount importance.

This work presents a general purpose lexical platform for the Spanish language. It has been designed and implemented as the core of a set of natural language applications currently under development at *Universidad Politécnica de Madrid*.

---

[2] The treatment of compositional or derivational morphology is not so critical.

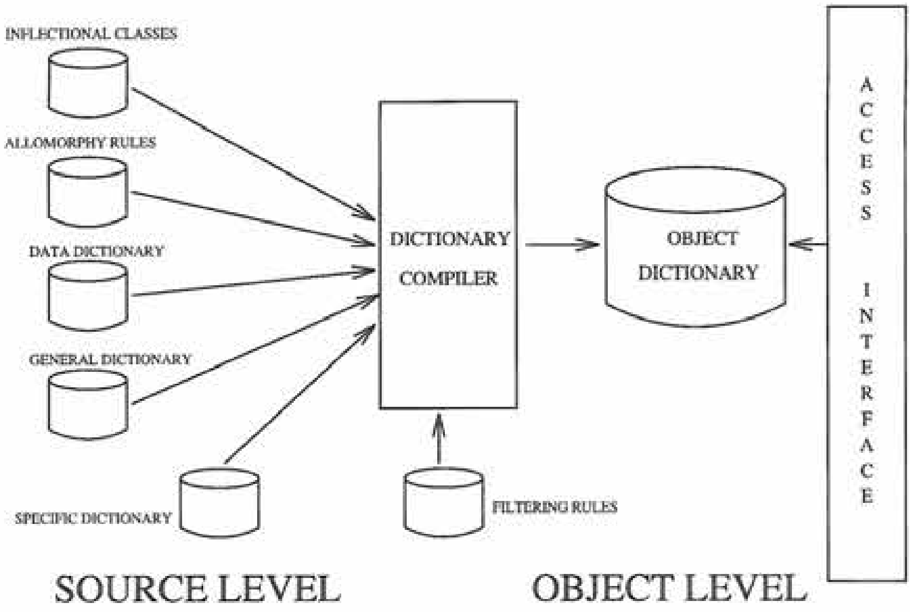

Fig. 1. Lexical architecture.

The overall lexical architecture of our platform is shown in figure 1. It considers two different levels of lexical representation:

Source Lexical Base: this level captures linguistic generalizations by merging related allomorph entries, by considering classes of lemmas or by specifying rules to compute different allomorphic variants. Inflectional morphemes, which constitute a closed group, are also included, as well as the set of lexicalized word forms. Lexicographers modify the database at this level.

Object Dictionary: this level is related to the computer processing of the lexical knowledge included in the Lexical Base. To facilitate such process, the lexical entries are expanded to different allomorphs that constitute the key entries in this level. It is automatically derived from the Source Lexical Base.

3 Morphological treatment

It is well known that morphological processes are divided into two types: processes related to the phonological and/or graphic form (morpho-graphemics), and processes related to the combination of morphemes (morpho-syntax). Each model treats these facts from its particular perspective. Two-level morphology uses phonological rules and continuation classes (in the lexical component). Mixed systems such as Bear (1986) have different sets of rules. These so-called non-concatenative processes are the most difficult single problem that morphological processors must deal with. Experience has shown that this is not easy for any approach. Two-level morphology (Koskenniemi 1983) uses rules that match lexical representations (lemmas) with surface representations (actual spelling forms). This approach has been claimed to be more elegant, but it is obvious that often the two-level model contains many rules needed for very few cases, since every language has irregularities that can only be treated as suppletive forms, e.g. *soy* (I am), *era* or *fui* (I was). In those cases, suppletion of morphemes or words is a must.

The pure two-level/finite-state automata model is not the most suitable for the treatment of some morpho-syntactic processes, and besides, in such cases one is required to depart from this approach, for example by adding an extension in which two-level rules are retained under the control of a feature based grammar, using simple features (Ritchie *et al.* 1987), or structured ones (Trost 1990).

Good finite-state processors for Spanish exist, for example: Tzoukermann and Liberman (1990), Badía *et al.* (1996), Carulla and Oosterhoff (1996), or even an on-line system for several languages at the URL http://www.xerox.fr/mltt/Tools/-morph.html, which is a pretty efficient implementation of the two-level model (Karttunen *et al.* 1992; Karttunen 1994).

The successful treatment of morphological phenomena in some languages by means of finite state automata appears to have led to the idea that this model is the most efficient and linguistically motivated way to deal with morphology computationally. However, we have opted to depart from the two-level approach, since we believe that the simplicity of our approach successfully competes against two-level/finite-state implementations for Spanish morphology. Our strongest argument is that allomorphy in Spanish is not phonologically conditioned (for which two-level models are the most suitable). Instead, sincretism and grammatically conditioned allomorphy (i.e. non-automatic allomorph selection triggered by specific morphemes) are the main features of fusional languages (cf. section 3.1). In addition, the defective paradigms and the alternation of correct forms are easy and handled naturally in a paradigmatic model, and they are much harder (although possible) to get in a two-level model. Our claim is that, both theoretically and computationally, paradigmatic morphology is the best suited for Spanish, and possibly for other languages with the same typological characteristics[3].

We claim that the following requirements for a morphological model are achieved by our approach:

Efficiency: if allomorphs are precomputed or stored in the dictionary, the morphological processors only need to carry out appropriate morpheme concatenation (Hausser 1991). In any case, unless a complete inventory of morpheme allomorphs is available, some device is needed for handling morpho-graphemic processes. It seems that Chomsky's hierarchy type 3 regular grammars are good enough for such purposes. Robust and efficient tools for regular expression processing also exist, so they can be used to implement morpho-graphemic descriptions, provided that a rule formalism is developed for such a purpose. Such descriptions will permit the *off-line* computation of the allomorphs by appropriate tools.

---

[3] Recently, Pirrelli and Battista (1996) have defended the paradigm-based approach for Italian as a "succesful blending of descriptive economy and declarativity", comparing favourably with a syntagmatic two-level morphology for selecting stem alternants in Italian.

Adequacy: suppletion is needed anyway – unless unique morpho-graphemic rules for special cases are written – and it is a much simpler approach than rules. We think it is more elegant to use a single, simpler device[4].

The morpho-syntactic component can be included as part of a syntax grammar. The origin of this work was the idea proposed by Lau and Perschke (1987) for the EUROTRA project, that morphological and surface syntactic processing can be combined into one context-free grammar augmented with unification. Our model is implemented now by means of a parsing platform, which is a chart-based context-free one augmented with feature unification, and morphological rules written in a PATR-II like format (Shieber 1986).

3.1 Major issues in Spanish computational morphology

Spanish morphology is not a trivial subject. As an inflectional language, it shows a great variety of morphological processes, particularly non-concatenative ones. We will try to summarize the most significant problems which any morphological processor of Spanish has to deal with:

1. A highly complex verb paradigm. For simple tenses, we consider 55 inflected forms (see table 1), excluding the archaic future subjunctive, but including the duplicate imperfect past subjunctive (six forms) and two *courtesy* imperative forms. If we add the 50 possible forms for compound tenses, then 105 inflected forms are possible for each verb[5].
2. The frequent irregularity of both verb stems and endings. Very common verbs, such as *tener* (to have), *poner* (to put), *poder* (to be able to), *hacer* (to do), etc., have up to seven different stems: *hac-er, hag-o, hic-e, ha-ré, hiz-o, haz, hech-o*. This example shows internal vowel modification triggered by different morphemes having the same external form: *hag-o*; *hiz-o, hech-o* (The first */-o/* is first person singular present indicative morpheme; the second */-o/* is third singular preterit indicative morpheme; and the third */-o/* is past participle morpheme – an irregular one, incidentally). As well as these non-concatenative processes, there are other, very common, kinds of internal variation, as illustrated by the following example:

   /e/ → */ie/*: *quer-er* (to want) → *quier-o* (I want)

   2300 out of 7600 verbs in our dictionary are classified as irregular, and 5300 as regular, i.e. only one stem for all the forms as in *am-ar*, *am-o*, etc. (*to love*).
3. Gaps in some verb paradigms. In the so-called *defective verbs* some forms are missing or simply not used. For instance, meteorological verbs such as *llover, nevar* (to rain, to snow), etc. are conjugated only in third person singular. Other ones are more peculiar, like *abolir* (to abolish) that lacks first,

---

[4] We refer to the device used at run-time by the morphological processor. The previous item showed that rule application has been moved to an *off-line* process.

[5] Our model accounts only for simple forms, since compound forms may have a syntactic treatment.

second and third singular and third plural present indicative forms, all present subjunctive forms, and the second singular imperative form. In other verbs, the compound tenses are excluded from the paradigm, like in *soler* (to do usually).

4. Duplicate past participles: a number of verbs have two alternative forms, both correct, like *impreso*, *imprimido* (printed). In such cases, the analysis has to treat both.
5. Suppletion: There are some highly irregular verbs that can be handled only by including many of their forms directly in the lexicon (like *ir* (to go), *ser* (to be), etc.).
6. Nominal inflection can be of two major types: with grammatical gender (i.e. concatenating the gender morpheme to the stem) and with inherent gender (i.e. without gender morphemes). Most pronouns and quantifiers belong to the first class, but nouns and adjectives can be in any of the two classes, with a different distribution: 4%[6] of the nouns have grammatical gender and 92% have inherent gender, while 70% of the adjectives are in the first group.
7. There is a small group (3%) of invariant nouns with the same form for singular and plural, e.g. *crisis*. On the other hand, 30% of the adjectives present the same form for masculine and feminine, e.g. *azul* (blue). There are also *singularia tantum*, where only the singular form is used, like *estrés* (stress); and *pluralia tantum*, where only the plural form is allowed, e.g. *matemáticas* (mathematics). Some nouns and adjectives present alternative correct forms for plural, e.g. for *bambú* (bamboo), *bambú-s* and *bambú-es*.
8. In contrast with the verb morphology, nominal processes do not produce internal change in the stem caused by the addition of a gender or plural suffix, although there can be alternative allomorphs produced by spelling changes: *luz*, *luc-es* (light, lights).

All these phenomena suggest that there is no such universal model (e.g. two-level, unification, or others) for (surface) morphology. Instead, we have approaches more suited to some processes than others. The computational morphologist must decide which is more appropriate for a particular language. We support the idea that unification and feature-based morphology is more adequate for languages, such as Spanish and other Romance languages, that have alternative stems triggered by specific suffixes, missing forms in the paradigm, and duplicate correct forms.

The formalized description of the morphological phenomena of Spanish was presented in Moreno (1991), where some interesting and well founded linguistic generalizations are made: Paradigms[7] for verbs are described to capture regularities in the inflectional behaviour of the Spanish verbs, and the same is done with nouns. All the lemmas belonging to a particular paradigmatic model not only share most of contextual and grammatical features but also have the same allomorph number

---

[6] The percentages shown in this section are relative to the lexicon we have built.

[7] These are not the traditional ones (Alarcos 1994), since they capture the problems arising in written language, such as diacritical marks, different surface letters for the same phoneme, etc.

```
imprim                        impres
concat  =  vl                 concat  =  vl
conj    =  3                  conj    =  3
stt     =  100                stt     =  99
sut     =  reg                sut     =  part1
cat     =  v                  cat     =  v
lex     =  imprimir           lex     =  imprimir
```

Fig. 2. Some entries for *imprimir* lemma.

and distribution. For instance, our model 11b has three allomorph stems, whose distribution is shown in the paradigmatic model in figure 11.

3.2 The model

As previously stated, our model relies on a context-free grammar augmented with feature unification that is particularly well suited for morpho-syntax. For morphographemics, our model depends on the storage – or *off-line* precomputation – of all the possible allomorphs both for stems and endings. This feature permits both analysis and synthesis to be limited to morpheme concatenation, as the general and unique mechanism. This dramatically simplifies the rule component and permits efficient implementation as a plus, since morphemes are no longer computed at run time.

We present some examples of dictionary entries: two allomorph stems for *imprimir* (figure 2), and two verbal ending entries (allomorphs) for the past participle morphemes (figure 7).

There `vm` and `vl` stand for the values of the 'morphological category' `concat`, that we are using to drive rule processing. All the object dictionary entries must have this morphological category. The inventory[8] of such categories follows:

w: For full inflected word forms.
wl: For words (nouns and adjectives) that can accept a number morpheme.
vl: For verb lexemes (stems).
nl: For nominal – nouns and adjectives – lexemes.
vm: For verb morphemes.
ng: For nominal gender morphemes.
nn: For nominal number morphemes.

We have introduced some contextual atomic features that impose restrictions on the concatenation of morphemes through standard unification rules. Such features are never percolated up to the parent node of a rule. Disjunction of atomic features is permitted to improve storage efficiency, and is used only for contextual

---

[8] Limited here to categories needed for verbal and nominal inflection. Our real system deals with clitic pronouns attached to verbs and some derivational processes also, but we omit them, for the sake of clarity.

Table 1. *Conjugation table:* stem_type *(***stt***) values*

	sing_1	**sing_2**	**sing_3**	**plu_1**	**plu_2**	**plu_3**	—
pres_ind	11	12	13	14	15	16	
impf_ind	21	22	23	24	25	26	
indf_ind	31	32	33	34	35	36	
fut_ind	41	42	43	44	45	46	
pres_subj	51	52	53	54	55	56	
impf_subj[a]	61	62	63	64	65	66	
cond	71	72	73	74	75	76	
imper		82	83		85	86	
inf							00
ger							90
part							99

[a] Two forms each.

Table 2. suffix_type *(***sut***) values*

`reg`	`pres`	`pret1`	`pret2`
`fut_cond`	`imp_subj`	`imper`	`infin`
`ger`	`part1`	`part2`	

features: *stem_type (*`stt`*), suffix_type (*`sut`*), conjugation (*`conj`*), gender_type (*`get`*)* and *number_type (*`nut`*)*.

In the conjugation table (Table 1), the *stem_type (*`stt`*)* values of the grammatical features person-number (heading row) and tense-mood (heading column) are displayed in boldface. For example, **sing_1** means first person, singular number; while **pres_ind** means present tense, indicative mood.

Each of the 49 entries[9] is represented by a numeric code[10], and the additional value `100` is used as a shorthand for the disjunction of all of them (used for regular verbs; see the entry example in figure 2). The contextual feature *stem_type* (**stt**) is used to identify the verb stem and ending corresponding to each form, and the contextual feature *suffix_type* (**sut**) distinguishes among several allomorphs of the inflectional morpheme by means of a set of values (Table 2).

Since this value set is much smaller than the *stem_type set*, we have chosen an alphabetic code. With the combination of both features, and the addition of a third feature `conj` (conjugation), we can state unequivocally which is the correct sequence of stem and ending for each case (see examples above, where *imprim* only matches

---

[9] The codes 83 and 86 stand for the *courtesy* imperative: *imprima (usted), impriman (ustedes)*. These word forms are the same as the 53 and 56 ones.

[10] Actually, this number encodes a particular combination of person, number, tense and mood features.

```
bambú                              president
cat       =   n                    cat       =   n
concat    =   wl                   concat    =   nl
agr gen   =   masc                 get       =   mas2 fem
agr num   =   sing                 nut       =   plu1
nut       =   plu1 plu2            lex       =   presidente
lex       =   bambú
```

Fig. 3. Entries for *bambú* and *presidente* lemmas.

```
doctor                             doctor
cat       =   n                    cat       =   n
concat    =   wl                   concat    =   nl
agr gen   =   masc                 get       =   fem
agr num   =   sing                 nut       =   no
nut       =   plu2                 lex       =   doctor
lex       =   doctor
```

Fig. 4. Entries for *doctor* lemma.

ido for all features, and *impres* matches *o*, thus preventing ill-formed concatenations such as *imprim-o*[11] or *impres-ido*).

In the same fashion, we have two special contextual features for the nominal inflection, `nut` (*number_type*) and `get` (*gender_type*), to identify the various allomorphs for the plural and gender morphemes, and associate them with the proper nominal stems. The examples in figures 3, 4, 8 and 9 show those contextual features both in nominal stem and morpheme entries.

These entries allow the analysis or generation of the word forms *presidente, presidenta, presidentes* and *presidentas* for the lemma *presidente*; *doctor, doctora, doctores* and *doctoras* for *doctor*; and *bambú, bambús* and *bambúes* for *bambú*.

The grammatical features *category* (`cat`), *lemma* (`lex`), *tense* (`vinfo tense`), *mood* (`vinfo mood`), *person* (`agr pers`), *number* (`agr num`) and *gender* (`agr gen`) are the only features that are delivered to the **w** node, and from this level can be used by a syntactic parsing grammar.

A unification-based system usually relies very much on the lexical side. A large, robust dictionary, properly coded, is needed. Additionally, our model depends on the accessibility of all possible allomorphs, so their storage is also necessary. Fortunately, there is no need to type all of them by hand, since this would be an impractical, time consuming and error-prone task. Morpho-graphemics for Spanish is quite regular and we have formalized and implemented the automatic computation of the allomorphs of any verb. This is done by means of regular-expression based

---

[11] Note that this is correct for indicative first singular, but not for past participle.

```
% 1. Regular verbs                       % 2. Non-regular verbs
w -> vl vm                              w -> vl vm
<$1 conj>    =   <$2 conj>              <$1 conj>    =   <$2 conj>
<$1 stt>     =   100                    <$1 stt>     =   <$2 stt>
<$1 sut>     =   <$2 sut>               <$1 sut>     =   <$2 sut>
<$0 cat>     =   <$1 cat>               <$0 cat>     =   <$1 cat>
<$0 agr>     =   <$2 agr>               <$0 agr>     =   <$2 agr>
<$0 vinfo>   =   <$2 vinfo>             <$0 vinfo>   =   <$2 vinfo>
<$0 lex>     =   <$1 lex>               <$0 lex>     =   <$1 lex>
```

Fig. 5. Rules for verbal morphology.

rules devised to compute these allomorphs automatically from the infinitive form, capturing morpho-graphemic generalizations in the paradigmatic models. The rule to compute the second allomorph (`alo 2 stem`) for the verbs in the model 11b in figure 11 is invoked with the expression `$rv11-2`. We will show how it works below, in the lexical formalism section.

3.3 *The grammar*

The rule component of the model is quite small, because most of the information is in the lexicon. In particular, inflected verb forms are analysed or generated by two rules. Actually, only one rule is needed, but as we used the value `100` for the `stt` feature for regular verbs instead of a disjunction of all the possible `stt` values, we split the rule in two. Verbal rules are shown in figure 5.

In the rules, `$0` refers to the root of the feature bundle of the left-hand side, and `$1`, `$2` to the roots of the feature bundles of the right-hand side, from left to right. As a notational shorthand, the context free part of the rules (`w -> vl vm`) is written with the required values of the `concat` feature of each of the feature bundles.

Nominal inflection is slightly more complicated, because of the combination of two inflectional morphemes (gender and number) in some cases. Our model needs the four rules shown in figure 6 to handle this. The first one is for singular words, when the stem has to be concatenated to a gender suffix (*niñ-o, niñ-a*); the second is for plural words, where an additional number suffix is added (*niño-s*); the third builds plurals from an allomorph stem and a plural morpheme (*león/leon-es*); and the fourth rule validates as words the singular forms (**wl**) obtained from the first rule without further concatenation.

4 Lexical formalism

The main characteristics of the Source Lexical Base representation language are described in this section. Further description can be found in Goñi and González (1995*a*). Examples will be given as needed, in order to illustrate the use of the language and its semantics, and special encoding conventions, since a formal description

```
% 1. Gender morphemes to stems
wl -> nl ng
<$1 get>       =  <$2 get>
<$0 nut>       =  plu1
<$0 agr>       =  <$2 agr>
<$0 lex>       =  <$1 lex>
<$0 ninfo>     =  <$1 ninfo>
<$0 cat>       =  <$1 cat>

% 2. Number morphemes to words
w -> wl nn
<$1 nut>       =  <$2 nut>
<$0 agr gen>   =  <$1 agr gen>
<$0 agr num>   =  <$2 agr num>
<$0 lex>       =  <$1 lex>
<$0 ninfo>     =  <$1 ninfo>
<$0 cat>       =  <$1 cat>

% 3. Number morphemes to stems
w -> nl nn
<$1 nut>       =  <$2 nut>
<$1 nut>       =  plu2
<$0 agr gen>   =  <$1 agr gen>
<$0 agr num>   =  <$2 agr num>
<$0 lex>       =  <$1 lex>
<$0 ninfo>     =  <$1 ninfo>
<$0 cat>       =  <$1 cat>

% 4. Singular words
w -> wl
<$0 agr>       =  <$1 agr>
<$0 lex>       =  <$1 lex>
<$0 ninfo>     =  <$1 ninfo>
<$0 cat>       =  <$1 cat>
```

Fig. 6. Rules for nominal morphology.

will not be presented. A detailed description of the formalism and its syntax is given in Goñi and González (1995*b*).

Each entry in the Source Lexical Base is composed of an *entry name* (EN) – a label – and an *entry structure* (ES) – a feature structure – restricted to be a tree. The ES has a number of labelled features, that can have an atomic value – a label assigned to that feature – or a structured one – another feature structure. Moreover, a string of characters can be encoded as an atomic value for a feature. Value assignment to a feature is achieved by an equation in the form:

$$p = v_1\ v_2\ \ldots\ v_n$$

where p is a sequence of one or more labels separated by blank spaces that constitute a path for accessing the feature from the root of the tree. The v_i are the atomic values that this particular feature can have. Only one value is permitted if it is of character string type. Paths in the left hand side of the equations are the mechanism provided to define a tree-shaped feature structure. The multiple-valued features are provided as a notational shorthand for disjunction.

The Source Lexical Base is split into sections, each one headed by a special keyword. An include facility is also provided in order to promote physical division of the Lexical Base into different computer files. The sections that may appear in the Source Lexical Base are reviewed below.

4.1 Morphemes and words

The morphemes section is designed for the inclusion of inflectional morphemes, which usually convey grammatical information such mood, aspect, person and tense

```
ido                                 o
concat         =  vm                concat         =  vm
conj           =  2 3               conj           =  2 3
stt            =  99                stt            =  99
sut            =  reg               sut            =  part1
vinfo tense    =  part              vinfo tense    =  part
vinfo mood     =  nofin             vinfo mood     =  nofin
```

Fig. 7. Some entries for past participle morphemes.

```
o                        e                        a
agr gen   =  masc        agr gen   =  masc        agr gen   =  fem
agr num   =  sing        agr num   =  sing        agr num   =  sing
get       =  mas1        get       =  mas2        get       =  fem
concat    =  ng          concat    =  ng          concat    =  ng
```

Fig. 8. Entries for gender morphemes.

(verbs), or gender and number (nouns or adjectives). The entries in this section will pass almost unchanged to the Object Dictionary upon compilation. Examples are shown in figures 7–9.

The words section is intended for lexicalized words that are included *as is* in the Source Lexical Base (see the example in figure 10). The most frequent *clients* of this section are very irregular words, usually with an auxiliary function. The entries in this section will also pass almost unchanged to the Object Dictionary. The section is provided to physically separate morphemes from words, although the behaviour of the entries in both sections will be almost the same when compiling the Object Dictionary.

4.2 Classes

Information inheritance has been widely used in artificial intelligence, as a knowledge representation mechanism, as well as a limited reasoning device. The use of such a technique in Computational Linguistics is widespread as Daelemans *et al.* (1992)

```
s                              es
agr num   =  plu               agr num   =  plu
nut       =  plu1              nut       =  plu2
concat    =  nn                concat    =  nn
```

Fig. 9. Entries for number morphemes

```
sé                                      soy
cat           =  v                      cat           =  v
lex           =  saber                  lex           =  ser
concat        =  w                      concat        =  w
agr pers      =  1                      agr pers      =  1
agr num       =  sing                   agr num       =  sing
vinfo tense   =  pres                   vinfo tense   =  pres
vinfo mood    =  ind                    vinfo mood    =  ind
```

Fig. 10. Entries for lexicalized words.

```
MV11b (MV)
alo 1 stt    =  0 12 13 14 15 16 21 22 23 24 25 26 31 32 33 34 35 36 \
                61 62 63 64 65 66 85 90 99
alo 1 sut    =  reg
alo 2 stem   =  $rv11-2
alo 2 stt    =  11 51 52 53 54 55 56 83 86
alo 2 sut    =  reg
alo 3 stem   =  $rv11-3
alo 3 stt    =  41 42 43 44 45 46 71 72 73 74 75 76
alo 3 sut    =  fut_cond
```

Fig. 11. Verbal inflectional model 11b.

show. We have opted for a simple version of default inheritance. Our language defines classes as bundles of feature-value pairs that can be inherited by any particular entry defined to be a member of a class. The entries belonging to a particular class inherit all the feature-value pairs present in their parent class. Inheritance is overridden for those feature-value pairs explicitly stated in the entry. This mechanism (default inheritance) provides a convenient and natural way to express regularities and exceptions. Classes can be members of other classes if desired and multiple inheritance is also allowed, so it is possible to build complex inheritance hierarchies that group and optimize the information organization. As multiple inheritance can cause conflicts, inheritance is expressed by means of an ordered list, and a priority schema – the *class precedence list* – is established to avoid conflicts. Although Russell *et al.* (1992) use a sophisticated strategy to build this list, we chose a simple depth-first and left to right searching algorithm to avoid conflicts, modified appropriately to remove duplicates.

A class definition is a label (EN) and a feature structure (ES) attached to it. If the class defined is a member of a set of other classes, these are listed in parenthesis after the EN. This also holds for entries in other sections (words, morphemes and lemmas).

Allomorphy rules are usually stated in a class definition. Rule invocation, however, is made when a particular child entry from that class is processed because of the

```
salir (MV11b C3)

usurpador (DOCTOR N)

mezzosoprano (LUNA N)

desosar (MV1 C1)
alo 2 stem = deshues
```

Fig. 12. Lemma entries.

inheritance mechanism. The EN of such entry acts as the argument for the rule. Figure 11 shows an inflectional class that inherits information from the model `MV`. All the verb lemmas that belong to this inflectional paradigm inherit such information by default.

4.3 Lemmas

A lemma is a grouping of related entries that share common information. Each lemma will be expanded to different entries when the Object Dictionary is compiled. For our purposes a lemma groups the allomorphs needed to build all the inflected forms, not all the possible surface realizations – for verbs, where 55 word forms are possible, a maximum of eight allomorphs are encoded.

For regular and subregular inflection, entries can be very short if the inheritance mechanism is used. For irregular lemmas, the entry is usually longer, because most of the information must be provided inside. In figure 12 we show some examples of lemma entries that make use of default inheritance. The last entry explicitly overrides the value inherited for the `alo 2 stem` feature.

4.4 Allomorphy rules

Allomorphy rules are stated in a separate section, and are designed to build particular allomorphs for a given lemma entry. Rules are usually invoked in class definitions, although they could be in a particular entry in the lemmas section instead. The example above that illustrated the classes section (figure 11) had three rule invocations[12]. Rule invocation is always stated in the value for a particular feature, and the value returned by the rule is assigned to such a feature. Rule invocation is done by rule name, preceding it with the special character $. A special invocation is provided with the token $$ for the identity rule[13]. As previously stated, any allomorphy rule invocation takes as its argument the relevant EN for the entry

---

[12] Two of them are explicit, and the third, to compute `alo 1 stem`, is inherited from the `MV` class.

[13] That is, returns the EN of the entry.

```
rv9a-2
 {X = .*}
 {C = [bcdfghjklmnpqrstvwxyz]+}
$Xe$Cir   ->   $Xie$C
$Xo$Cir   ->   $Xue$C
```

Fig. 13. Allomorphy rule.

under consideration, and if invocation takes place in a class definition the argument is the EN of the entry that inherits that feature.

Rule application is a pattern-matching process. The argument of the rule is matched sequentially against the *left hand side* of each production in the rule. When a match succeeds, the relevant right hand side is returned. If there is no successful match, the rule fails and neither value is returned nor assigned to the feature that invoked the rule. Patterns on the left hand side of the rule are a sequence of characters and variables – that represent a regular-expression pattern declared in a rule header. When the argument is successfully matched against the left hand side of the current production, variable instantiation takes place. If the right hand side of the production contains that variable, its instantiated value is used to compute the returned value.

A rule contains a name, followed by local variable declarations and one or more productions, whose left and right sides are separated by the special token ->. Variable declarations are assignments – enclosed in brackets – of regular expressions to the variable identifier (some single alphabetic character). Regular expressions are the standard ones used in some UNIX[14] operating system commands (Kernighan and Pike 1984), so we will not discuss them here. Variable invocation in the productions are preceded by the special character $.

The example in figure 13 shows one of the rules that are needed in the verbal model `MV9a`. Provided that γ is any consonant sequence, it computes an allomorph from an infinitive form when it finishes in -eγir, by changing *e* to *ie* and deleting the *ir* ending, or, if the infinitive form ends in *-oγir* it changes the *o* to *ue*, deleting the ending also. These two productions can compute the allomorphs *sient*, *requier*, *muer* or *duerm* from the forms *sentir* (to feel), *requerir* (to require), *morir* (to die) and *dormir* (to sleep), respectively.

4.5 Type checking

This particular section has been designed to provide some kind of type checking. Open (any value) and closed (fixed set of values) features have to be declared here. For closed features all the possible values have to be declared too. We show some example declarations in figure 14.

---

[14] UNIX is a registered trademark of the X/OPEN Company Ltd.

```
stem  =
pers  =  1 2 3
agr   =  @(gen num) @(num pers)
```

Fig. 14. Feature declaration.

`stem` is an open feature that can take any atomic value (including character strings), `pers` is a closed one, and its legal values are only `1`, `2` or `3`, `agr` is a closed feature that can take only a feature structure as its value, and some restrictions are declared on its possible feature components: `gender` and `number`, or `number` and `person`[15].

This section is of special interest for consistency checking throughout the whole source lexical base, for detecting misspellings of feature names and values, and as reference for lexicographer editors. It is used also by some tools to improve the efficiency of the deliverable products, as closed feature values can be compiled out because they form a finite set.

4.6 Object dictionary generation

In this section of the Source Lexical Base, rules are given for building the Object Dictionary. Each rule is a sequence of tree manipulating operators that can be used to modify, filter out or add some branches to the tree structure of the feature bundles.

The section specifies a set of rules for each of the sections containing lexical entries (lemmas, words and morphemes). From the point of view of Object Dictionary construction all these three sections are equivalent, and it is because of these rules that they behave differently.

This section is split into three subsections, each one headed by one of the labels `LEXEMES`, `MORPHEMES` or `WORDS`. The rules in each subsection will be applied to the entries defined in the relevant section of the Source Lexical Base. For each rule succesfully applied, a new entry will be generated in the Object Dictionary.

Each rule consists of a sequence of equations. Their left hand side refers to the entry generated in the Object Dictionary and the right hand side to the entry under consideration in the Source Lexical Base. The special tokens $$ and @ refer to the EN and to the ES respectively. All rules must have a value assigned to $$ and to @, and the rule is successful if an effective value is assigned to $$ at runtime.

Tree branches can be accessed by their path from @, and assignments to non-existing branches are considered tree augmenting. Deleting a branch is done by specifying an incomplete copy: on the right hand side of an equation, after a subtree specification, a sequence of paths to eliminate is written in parenthesis, preceding each

---

[15] It is possible that only one of the features appears, but not `gen` and `pers` at the same time.

```
LEXEMES                                    MORPHEMES

$$       =  @ alo 1 stem                   @          =  @
@        =  @ alo 1 (- stem)               $$         =  $$
@        =  @ (- alo - aux)
@ lex    =  $$

$$       =  @ alo 3 stem                   WORDS
@        =  @ alo 3 (- stem)               @          =  @
@        =  @ (- alo - aux)                $$         =  $$
@ lex    =  $$                             @ concat   =  w
```

Fig. 15. Rules for Object Dictionary compilation.

one with a minus (−) sign. Rules are invoked sequentially, and non-monotonically: an equation can override a value assigned by previous equations.

For each possible allomorph that can be included in a lemma entry, a rule should be included in this section. We have not considered iteration to enhance this tree manipulating language to cope with an indeterminate number of allomorphs, because we have always found a manageable number of them.

A sample of the rules is shown in figure 15. The entries in the *morphemes* sections are just copied, while the `concat` feature, with the value `w`, is added to those in the *words* section. For the *lemmas* the same set of operations is repeated for each of the possible allomorphs (we show just the relevant rules for the first and the third ones). The allomorph is converted to the EN and the older EN becomes the feature `lex` of the target ES. Some deleting is also carried out.

Figure 16 shows the resulting entries in the Object Dictionary obtained by applying such rules to the lemma entry *salir* (cf. figure 12).

4.7 The lexicon

The lexicon for this platform has been derived from different sources:

- The concise version of the COLLINS English-Spanish dictionary.
- The work done at New York University by A. Moreno and C. Olmeda in the framework of the PROTEUS project (Grishman *et al.* 1993).
- The tag-set proposed by A. Martín (1994) at Universidad Autónoma de Madrid for morpho-syntactic tagging of Spanish texts. See also Martín and Goñi (1995).
- Corpora of Spanish texts compiled by the King's College of London and by the International Telecommunications Union. (A small part of the first corpus has been tagged manually by Martín.)
- Collections of texts from Spanish newspapers (ABC, *El Mundo*, etc.) and lists of words.

Morphological information has been derived in a semi-automatic way:

```
sal
cat          =  v
conj         =  3
concat       =  vl
stt          =  0 12 13 14 15 16 21 22 23 24 25 26 31 32 33 34 35 36 \
                61 62 63 64 65 66 85 90 99
sut          =  reg
lex          =  salir

salg
cat          =  v
conj         =  3
concat       =  vl
stt          =  11 51 52 53 54 55 56 83 86
sut          =  reg
lex          =  salir

sald
cat          =  v
conj         =  3
concat       =  vl
stt          =  41 42 43 44 45 46 71 72 73 74 75 76
sut          =  fut_cond
lex          =  salir

sal
cat          =  v
conj         =  3
concat       =  w
agr pers     =  2
agr num      =  sing
vinfo mood   =  imper
lex          =  salir
```

Fig. 16. Expanded entries.

1. Tools were implemented to classify words (nouns, adjectives and verbs) according to morphological models from their surface form.
2. A Prolog generator, GRAMPAL (cf. section 5.4.3), was used to produce all the inflected forms (or the most representative ones in the case of verbs) for each dictionary entry. This was devised as a way to check manually the results of the classifiers.
3. Finally, the inflected forms were checked manually by groups of semi-volunteer students[16], and the entries were fixed accordingly.

---

[16] They deserve special mention, but their names cannot be included here by obvious reasons.

5 Tools

5.1 Verb classifier

As previously stated, morpho-graphemics for Spanish is fairly regular, and most verbs can be classified in paradigmatic models attending to surface patterns. To help dictionary building we have developed a verb classifier[17] for verbal models by means of regular expression patterns. Entry lemmas are matched against the patterns in sequential order. When the first match happens, the lemma is arranged into the conjugation model associated with the pattern.

These patterns are built based on empirical criteria and the evolution of Spanish verb morphology thus allowing us to state classification rules confidently. As an example we will show some of those patterns:

```
   /pezar$/      {v("3a"); continue}
   /aizar$/      {v("3a"); continue}
/comenzar$/      {v("3a"); continue}
  /gonzar$/      {v("3a"); continue}
   /orzar$/      {v("3a"); continue}
     /zar$/      {v("2") ; continue}
```

With such rules, any verb ending in *zar* is classified into the model named 2, except if the ending is one of the more specific ones, previously declared. In that case it is classified into the model 3a.

Thus, verbs in the conjugation model 2 have two allomorphs: the first one is regular (stripping the *-ar* ending), and the second one changes from *z* to *c* for some inflected forms, depending on the first character of the morpheme to be added[18].

In contrast, verbs belonging to the 3a model present four allomorphs: the first is the regular one, and the last three ones account for vowel changes in the stem, such as diptongation or marked diacritical stress, whether combined with the */z/* to */c/* spelling change or not.

With such a scheme, any particular verb can be classified into one of 34 conjugation models. After that, all the information needed for morphological recognition or generation, including all the relevant allomorphs, can be obtained from this paradigmatic model by means of inheritance devices. Note that we have been able to extend the dictionary automatically from mere verb lists, and that we will be able to add new verbs automatically to the dictionary.

We do not claim that this approach to verb classification can be generally applicable to many languages, but only to those whose inflectional classes can be predicted from the surface form of a word. In that sense, it could be useful to test whether the classifier can be adapted to languages such as Catalan, Portuguese, French or Italian.

---

[17] In fact, it is a *script* for the program '*awk*', included in any UNIX operating system.

[18] In this case, the change is produced when the morpheme starts in *e* or *é*, because *ze* is not possible under Spanish spelling rules.

5.2 Dictionary compiler

Object dictionary generation is achieved by specialized tools that are guided by the filtering rules seen above. These are a set of rules of a tree-manipulating language that allows the filtering, addition or modification of the branches of the feature bundle attached to each entry, as well as the computing of the different allomorphs for each lemma and their assignment of the relevant feature bundle. The tools (all of them implemented in C/C++) deal with:

1. Interpretation of filtering rules.
2. Inheritance searching algorithms, to assign the relevant feature bundle to each entry in the compiled dictionary.
3. Interpretation of regular expression rules to compute the allomorphs. The *GNU regex* package (Hargreaves and Berry 1992) has been used for such a purpose.
4. Building the *letter-tree* index (*trie*) for efficient retrieval. This indexing system will be described in the next section.

5.3 Dictionary interface

A flexible software access interface has been implemented to allow the applications a fast retrieval of lexical entries. The implementation language adopted is C/C++ also. A *trie* index is built while the object dictionary is compiled. The storage inefficiency of the *vanilla trie* index is solved with some simple modifications of the algorithm of the double-array *trie* described in Aoe *et al.* (1992). This indexing mechanism allows the integration in this interface of the word segmentation needed for morphological analysis. Thus, word segmentation is no longer a blind process, since it is guided by the entries included in the lexicon. This module also includes a software library for feature structure manipulation.

5.4 Parsing tools

Applications can be built in a modular way, selecting the software modules that are best suited for them. Software interfaces between the different modules and tools are cleanly defined to ease software integration and tool combination in a flexible and modular way. Figure 17 shows the system architecture, in particular inter-module dependencies within the implemented tools. In the following we describe all of them briefly.

5.4.1 Unification modules

Two modules were built for the processing of feature structures contained in PATR-II rules:

- A full unification module that allows real value sharing among features. The algorithm implemented is an enhancement of the *quasi-destructive graph unification algorithms* described by Wroblewski (1987) and Tomabechi (1991).

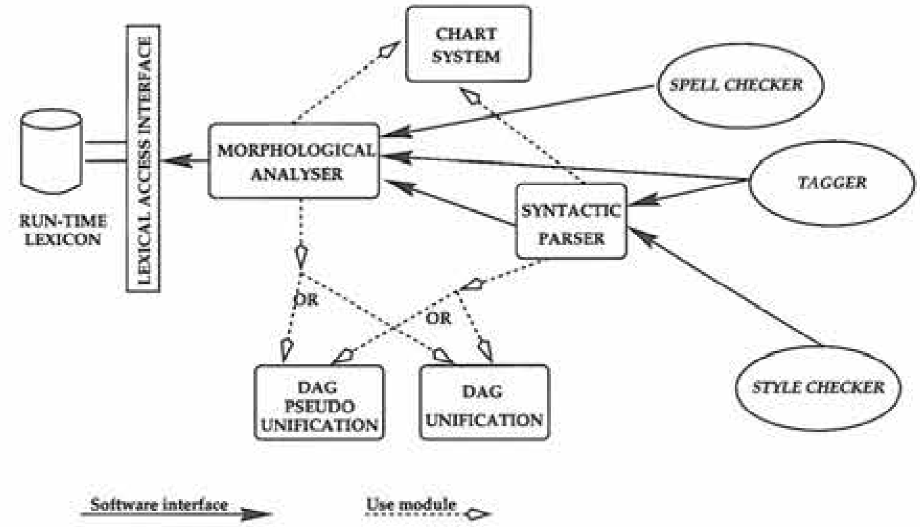

Fig. 17. Parsing tools architecture.

- A pseudo-unification module that treats feature bundles as trees, without value sharing. Our implementation is a modification of the pseudo-unification algorithm described by Tomita *et al.* (1988). A lot of enhancements have been added to maintain consistency in the unified feature structures, guaranteeing that all the values are exactly the same in the (pseudo) unified paths, independently of the order in which the equations are given[19].

5.4.2 *The modular chart parsing system*

Our modular chart parsing system, called NUCLEO, is a module that serves as a basis for the development of chart parsers in C programming language. It is based on the CMCHART architecture proposed in Thompson (1983), and includes facilities for implementing parsers with different searching strategies, rule invocation strategies, and top-down, bottom-up or mixed strategies.

5.4.3 *Tools for morphological analysis/generation*

A morphological analyser has been implemented using NUCLEO, the pseudo-unification module[20] and the lexical access interface (see figure 17). It is a modified chart parser that interprets word formation rules like those in figures 5 and 6. This implementation is reported in González *et al.* (1995).

We also developed a prototype of morphological analyser/generator in PROLOG

---

[19] Such enhancements include the smart sorting of the equations and double evaluation of the equalities.

[20] We chose this implementation for reasons of efficiency, following the claim of Tomita *et al.* (1988) that best performance is achieved by such an algorithm.

programming language, called GRAMPAL. It is, basically, a DCG grammar for word formation. The advantages of this prototype are declarativity and bidirectionality. The lexicon needed for its operation is automatically derived from our source lexical base. GRAMPAL is fully described in Moreno and Goñi (1995).

5.4.4 *The parser*

The parser is built on top of the NUCLEO system, and is integrated with the full unification module. The grammars have to be written in a PATR-II like format to feed the parser. A parser with the pseudo-unification module is also available, if so desired.

5.5 *Application tools*

5.5.1 *The tokenizer*

A tokenizer for the Spanish language has been developed using *flex*, the GNU generator of lexical analysers (Paxson 1995). This tokenizer is just a prototype, used for evaluating the possibilities of a tool of this kind. It is planned to implement a fully functional tokenizer in C in the near future.

The main goal of the tokenizer is to split a text into tokens, which most of the time will be simple words, although multi-word tokens are allowed. The tokenizer returns a token per line (which makes the post processing of the output easier) and each of these tokens may be marked with a tag in order to differentiate between types of tokens: proper nouns, text between quotes or parenthesis, beginning and end of sentences, punctuation marks, paragraph delimiters or strange words (those which contain both numeric and alphabetic characters, or non Spanish characters); when the tokenizer returns a *normal* word, it is not tagged.

5.5.2 *The stochastic tagger*

In the context of the CRATER project[21], the Xerox Tagger (Cutting *et al.* 1992) has been adapted to Spanish. The Xerox Tagger uses a statistical method for text tagging. In this kind of systems, ambiguity of assignment of a tag to a word is solved on the basis of the most likely interpretation. A form of Markov model is used that assumes that a word depends probabilistically on just its part-of-speech, which in turn depends on the category of the preceding two words.

One of the main advantages of the Xerox Tagger is that it does not need any tagged corpus; only a lexicon with the more common words and their corresponding tags, and a suffix lexicon which stores the set of tags which may correspond to a

---

[21] This project (*C*orpus *R*esources *A*nd *T*erminology *E*xt*R*action) is funded by the Commission of the European Communities under contract MLAP-93/20. Our group contributes to this project as subcontractor of *Laboratorio de Lingüística Informàtica, Universidad Autónoma de Madrid*, Spain, with Fernando Sánchez being the project leader of the Spanish team.

particular suffix[22] are necessary. With these elements it is possible to train, even with an untagged corpus, the Hidden Markov Model (HMM) in which the tagger is based.

Once the tagger has been trained, it can be fed with a string or a text to be tagged. First of all, the input is converted into a sequence of tokens which are passed to the lexicon; if the token is in the lexicon, it is annotated with the tag(s) provided by the lexicon. If it is not, but the token can be split into a verb form plus some clitic pronouns, this decomposition is done, and each of the elements resulting from the splitting is tagged as if it were a token originally obtained from the input text. If the token is neither in the lexicon nor can be split, the suffix table is looked up in the lexicon for a matching; if there is any, the set of tags corresponding to the matched entry is returned. Finally, if all of this fails, a default set of tags is returned.

The output of the tagger could be used as input for a syntactic parser, or could be used for compiling statistics on order and relations between different categories of words.

The modifications made to the Xerox Tagger in order to adapt it to Spanish have to do mainly with suffix handling and processing of clitic pronouns (Sánchez and Nieto 1995).

5.5.3 The verbal subcategorization tool

Verbal subcategorization frames are the different syntactic structures that a particular verb can admit: types of complements (and required prepositions), non-finite forms, reflexive forms, auxiliaries, etc. A tool, called SOAMAS (Monedero *et al.* 1995), has been developed to automatically obtain the frames corresponding to Spanish verbs from tagged texts. The system includes three grammars. The first one, for the analysis of the verb phrase, identifies main and auxiliary verbs (as well as the possible intermediate prepositions or conjunctions and clitic pronouns). The second one recognizes nominal, adjective and prepositional phrases inside the verb phrase. The last grammar describes the structure of the verbal complements in Spanish. Grammars have been implemented from the work in Hallebeek (1992).

There were 25 different subcategorization frames considered, including simple, composed and auxiliary frames. The system has not been integrated yet with the stochastic tagger and the current version of the morpho-syntactic analyser, but it has been tested by using a 10,000 word text manually tagged by Martín (1994), but no statistical analysis has yet been carried out.

5.6 Demonstrators

E-mail and Web interfaces have been built to show the capabilities of the tools developed. E-mail requests can be sent to `aries@mat.upm.es`, with one of the following subjects:

---

[22] This suffix lexicon has been generated manually by Fernando Sánchez; the original tagger has the possibility of generating suffixes automatically from a corpus, but this method produces poor results in the case of Spanish.

- `ayuda`: to get instructions (in Spanish) on the use of the demonstrator.
- `comprueba`: takes at most 10 lines of the message body or the first 100 words and returns to the petitioner the lists of unknown and unanalysed words.
- `analiza`: to get the morpho-syntactic analysis of one word (the first word in the message body). For instance, the answer obtained for the word *volvíais* is:

  ```
  lex = volver
  cat = v
  agr pers = 2
  agr num = plu
  vinfo tense = impf
  vinfo mood = ind
  ```

  that is, this word is the plural second, imperfect tense, indicative mood form of the verb *volver*.

Using a Web browser, our ARIES page can be reached at the URL:

`http://www.mat.upm.es/~aries`

where forms for the e-mail interface are also available, in addition to on-line papers, licensing and other information. A special interactive interface is also provided for morphological analysis.

6 Evaluation

A first evaluation of the ARIES platform has been carried out regarding its ability for spell checking purposes. Lexical coverage has been the only aspect under consideration: comparison with other morphological analysers is difficult because of the differences in their tag sets. What we are using as a spell checker is in fact a morpho-syntactic analyser that generates, for recognized words, all their possible analysis[23]. No alternative spelling is provided for unrecognized words.

The current version of the platform is relatively efficient in terms of space needed for the storage of object dictionaries. No effort has been made however until now to improve time efficiency. Anyway, an improvement of at least one order of magnitude is considered a feasible goal in the very short term.

Evaluation has been carried out by using a collection of texts from the Culture Supplement of the Spanish newspaper *ABC*. The text was previously tokenized to filter out proper nouns, acronyms, quoted and bracketed expressions, etc. Of course, the filtered output still contained foreign words and typographical errors. A specialized dictionary of culture terms including around 400 words (less than 1200 inflected forms) was created to supplement the core dictionaries of the four spell checkers used for evaluation: ARIES on the one hand and the Spanish versions of

---

[23] Over-acceptance is marginal by design. Of course, the lexicon acquisition process is prone to errors and some non-words might have been accepted.

Table 3. *Results of the evaluation*

Spell checker	Unrecognized distinct words	% error[a]	Unrecognized words	% error[b]
ARIES	3,167	7.5%	4,264	0.6%
WORD 7	5,174	12.2%	9,128	1.3%
ISPELL	5,630	13.3%	9,919	1.4%
WP 5.1	9,245	21,8%	18,622	2.6%

[a] Over 42,364 distinct words.
[b] Over 714,124 words.

the WP 5.1, Word 7 and ISPELL 3.1.00 spell checkers on the other[24]. The size of the corpus used, after tokenization, is 714,124 words long. The number of distinct forms is 42,364.

The results of the evaluation are shown in Table 3.

7 Conclusions and future work

The need for the development of our own lexical framework arose from the fact that other existing formalisms presented some inadequacies in order to cover the goals we stated at the first section of this paper, like verbosity of strong ideological commitment to particular linguistic theories. Although we tried to minimize ideological commitment, some definitive decisions had to be made, like abandoning the two-level morphology approach found almost everywhere, and adopting a new morphological model. This has proved useful, since the formalism has been successfully applied to account for the morphology of the Spanish language with an extensive coverage (Goñi *et al.* 1995).

But this framework would be useless if it were computationally intractable. A set of software tools has beed designed around it, setting-up the basis of our lexical platform. Extensive work has been carried out at our site to develop a loosely coupled, highly modular environment that allows us to integrate this set of tools (González *et al.* 1995*a*). Among them we will cite efficient dictionary access libraries, conversion tools between source and object formats (this includes regular expression rule interpretation, multiple inheritance management, etc.), and morphological analyser and generator.

The current dictionary has a considerable size: around 38,000 lemma entries, including 21,000 nouns, 7300 verbs, and 10,000 adjectives, and around 500 entries for prepositions, conjunctions, articles, adverbs and pronouns. The model could be

---

[24] WordPerfect is a trademark of Corel Corporation. Word is a trademark of Microsoft Corporation. The Spanish ISPELL dictionaries used for this evaluation (version 1.4, December 1994) have been developed by Rodríguez and Carretero (1996) at the Facultad de Informática, Universidad Politécnica de Madrid, Spain.

used for derivational morphology and compounds as well, but this has not been done fully yet[25], since further linguistic analysis must be done to specify the features needed to permit derivatives and compounds.

The full power of context-free grammars is not required for morpho-syntactic processing, so we are planning to write morphological rules in a less elegant, but more efficient, *possible continuations* or regular grammar implementation. This can be done as the full recursion that context-free grammars provide us with is not needed for morphology, so a regular grammar can be used, thus allowing us to achieve further performance. For practical reasons, we opted firstly to test intensively the morphological model with a Prolog prototype using the built-in DCG package (Moreno and Goñi 1995), and secondly to integrate morphological and syntactical processing in existing parsing tools. In any case, the morphological model will remain the same in both implementation approaches.

References

Alarcos, E. (1994) *Gramática de la Lengua Española.* Colección Nebrija y Bello. Real Academia Española. Espasa Calpe.

Aoe, J., Morimoto, K. and Sato T. (1992) An efficient implementation of trie structures. *Software–Practice and Experience* **22**(9): 695–721.

Badía, T., Egea, M. A. and Tuells, A. (1996) SEGMORF: Un formalismo para analizadores morfológicos de dos niveles. *Actas de la XII Conferencia de la Sociedad Española para el Procesamiento de Lenguaje Natural (SEPLN'96)*, pp. 63–71. Sevilla, Spain.

Bear, J. (1986) A morphological recogniser with syntactic and phonological rules. *Proceedings of the 11th International Conference on Computational Linguistics (COLING'86)*, pp. 272–276.

Briscoe, T., de Paiva, V. and Copestake, A. (1993) *Inheritance, Defaults and the Lexicon.* Studies in Natural Language Processing. Cambridge University Press.

Carulla, M. and Oosterhoff, A. (1996) El tratamiento de la morfología flexiva del castellano mediante reglas de dos niveles en una gramática de unificación. *Actas de la XII Conferencia de la Sociedad Española para el Procesamiento de Lenguaje Natural (SEPLN'96)*, pp. 72–80. Sevilla, Spain.

Cutting, D., Kupiec, J., Pedersen, J. and Sibun, P. (1992) A Practical Part-of-Speech Tagger. *Technical Report SSL-92-01*, Xerox PARC.

Daelemans, W., De Smedt, K. and Gazdar, G. (1992) Inheritance in Natural Language Processing. *Computational Linguistics* **18**(2): 205–218.

González, J. C., Goñi, J. M. and Nieto, A. F. (1995*a*) ARIES: A ready for use platform for engineering Spanish-processing tools. *Digest of the Second Language Engineering Convention*, pp. 219–226. Department of Trade and Industry, London, UK.

González, A. L., Goñi, J. M. and González, J. C. (1995*b*) Un analizador morfológico para el castellano basado en chart. *Actas de la VI Conferencia de la Asociación Española para la Inteligencia Artificial (CAEPIA'95)*, pp. 343–352. Alicante, Spain.

Goñi, J. M. and González, J. C. (1995*a*) A framework for lexical representation. In *Proceedings of AI'95: Fifteenth International Conference. Language Engineering '95*, pp. 243–252. Montpellier, France.

Goñi, J. M. and González, J. C. (1995*b*) Manual de Referencia de la Plataforma Léxica ARIES. Versión 5.0. Infome Técnico, GSI/DIT/UPM 25/95.

---

[25] Although it is not reported here, clitic attachment to verbs and some derivational processes are captured in the rule component.

Goñi, J. M., González, J. C. and Moreno, A. (1995) A lexical platform for Spanish. In F. Verdejo (ed.), *Proceedings of The Computational Lexicon Workshop (ESSLLI'95)*, pp. 61–65. Barcelona, Spain.

Grishman, R., Moreno, A. and Olmeda, C. (1993) Developing a multi-lingual information extraction system. *Symposium on Advanced Information Processing & Analysis.* Tysons Corner, USA.

Hallebeek, J. (1992) *A Formal Approach to Spanish Grammar.* Language and Computers: Studies in Practical Linguistics, volume 7. Rodopi.

Hargreaves, K. A. and Berry, K. (1992) Regex Manual. Edition 0.12a. Free Software Foundation.

Hausser, R. (1991) Principles of Computational Morphology. *Technical Report*, Laboratory for Computational Linguistics, Carnegie Mellon University.

Karttunen, L., Kaplan, R. M. and Zaenen, A. (1992) Two-level morphology with composition. *Proceedings of the 14th International Conference on Computational Linguistics (COLING'92)*, pp. 141–148.

Karttunen, L. (1994) Constructing lexical transducers. *Proceedings of the 15th International Conference on Computational Linguistics (COLING'94)*, pp. 406–411.

Kernighan, B. W. and Pike, R. (1984) *The UNIX Programming Environment.* Prentice-Hall.

Koskenniemi, K. (1983) Two-level morphology: A general computational model for word-form recognition and production. *PhD Thesis*, Department of Linguistics, University of Helsinki.

Lau, P. and Perschke, S. (1987) Morphology in the Eurotra base level concept. *Proceedings of the 3rd Conference of the European Chapter of the Association for Computational Linguistics (EACL'87)*, pp. 19–25.

Martín, A. (1994) Una Propuesta de Codificación Morfosintáctica para Corpus de Referencia en Lengua Española. *Tesis doctoral*, Universidad Autónoma de Madrid, Departamento de Lingüística, Lenguas Modernas, Lógica y Filosofía de la Ciencia.

Martín, A. and Goñi, J. M. (1995) Una propuesta y un etiquetador de codificación morfosintáctica para corpus de referencia en lengua española. *Actas del XIII Congreso Nacional de la Asociación Española de Lingüística Aplicada* (in press). Castellón, Spain.

Monedero, J., González, J. C., Goñi, J. M., Iglesias, C. A. and Nieto, A. F. (1995) Obtención automática de marcos de subcategorización verbal a partir de texto etiquetado: el sistema SOAMAS. *Actas de la XI Conferencia de la Sociedad Española para el Procesamiento de Lenguaje Natural (SEPLN'95)*, pp. 241–254. Bilbao, Spain.

Moreno, A. (1991) Un Modelo Computacional Basado en la Unificación para el Análisis y la Generación de la Morfología del Español. *Tesis Doctoral*, Universidad Autónoma de Madrid, Departamento de Lingüística, Lenguas Modernas, Lógica y Filosofía de la Ciencia.

Moreno, A. and Goñi, J. M. (1995) GRAMPAL: A morphological model and processor for Spanish implemented in Prolog. In: M. Sessa and M. Alpuente (eds.), *Proceedings of the Joint Conference on Declarative Programming (GULP-PRODE'95)*, pp. 321–331. Marina di Vietri, Italy.

Paxson, V. (1995) *Flex Manual.* Edition 2.5. Free Software Foundation.

Pirrelli, V. and Battista, M. (1996) Monotonic paradigmatic schemata in Italian verb inflection. *Proceedings of the 16th International Conference on Computational Linguistics (COLING'96)*, pp. 77–82.

Real Academia Española. (1992) *Diccionario de la Lengua Española (vigésima primera ed.).* Espasa Calpe.

Rodríguez, S. and Carretero, J. (1996) Formal approach to Spanish morphology: the COES tools. *Actas de la XII Conferencia de la Sociedad Española para el Procesamiento de Lenguaje Natural (SEPLN'96)*, pp. 118–126. Sevilla, Spain.

Ritchie, G. D., Pulman, S. G., Black, A. W. and Russell, G. J. (1987) A computational framework for lexical description. *Computational Linguistics* **13**(3–4): 290–307.

Russell, G. J., Carroll, J. and Warwick-Armstrong, S. (1992) A practical approach to multiple default inheritance for unification-based lexicon. *Computational Linguistics* **18**(3): 311–337. (Also in Briscoe *et al.* (1993), pp. 137–147.)

Sánchez, F. and Nieto, A. F. (1995) Development of a Spanish Version of the Xerox Tagger. *Technical Report*, Universidad Autónoma de Madrid. (Recoverable as http://xxx.lanl.gov/list/cmp-lg/9505035.)

Shieber, S. M. (1986) *An Introduction to Unification-Based Approaches to Grammar.* CSLI Lecture Notes, number 4. Center for the Study of Language and Information, Stanford University.

Thompson, H. S. (1983) MCHART: A flexible modular chart parsing system. *Proceedings of the National Conference on Artificial Intelligence (AAAI'83)*, pp. 408–410.

Tomabechi, H. (1991) Quasi-destructive graph unification. *Proceedings of the 29th Annual Meeting of the Association for Computational Linguistics (ACL'91)*, pp. 315–322.

Tomita, M., Mitamura, T., Musha, H. and Kee, M. (1988) The generalized LR parser/compiler. Version 8.1. User's Guide. *CMU-CMT-88-MEMO*, Center for Machine Translation, Carnegie Mellon University.

Trost, H. (1990) The application of two-level morphology to non-concatenative German morphology. *Proceedings of the 13th International Conference on Computational Linguistics (COLING'90)*, **2**, pp. 371–376.

Tzoukermann, E. and Liberman, M. Y. (1990) A finite-state morphological processor for Spanish. *Proceedings of the 13th International Conference on Computational Linguistics (COLING'90)*, **3**, pp. 277–281.

Wroblewski, D. A. (1987) Nondestructive graph unification. *Proceedings of the 6th National Conference on Artificial Intelligence (AAAI'87)*, pp. 582–587.