Web-surfing the lexicon

M. Vilares D. Cabrero L. Docampo

M. Vilares
Computer Science Department
University of Corunna
Campus de Elviña s/n
15071 La Coruña
Spain.
E-mail: vilares@dc.fi.udc.es

D. Cabrero
Ramón Piñeiro
Research Center for Humanities
Estrada Santiago-Noia, Km. 3
A Barcia, 15896 Santiago de Compostela
Spain.
E-mail: dcabrero@cirp.es

L. Docampo
He made his masther thesis
within the *Cole* group

Abstract

This paper describes the efforts made in order to distribute an access language resources including the tagger generation tools, the lexicon from which the tagger is generated and the tagger itself. It also describes the evolution from a centralized, hardly portabel system to an open and widely accessible one.

Key Words: Tagging, Graphical User Interface, WWW, Maintenance.

1. Introduction

Natural language processing is a typical example of multi-disciplinary task, involving both capabilities from computer science and linguistics. In addition, the complexity of the problem often makes the participation of large sized research groups necessary.

In practice, this leads to work with different teams in different research centers over common questions, which clearly presents a strong case for the consideration of concurrent and distributed development environments. This is, for example, the case of the lexical database filled in for a tagger.

On the other hand, we can also apply the preceding reasoning from the point of view of the user and get similar conclusions. For example, focusing on the case of tagger design, there are few things more frustrating than spending a great deal of time looking for lexicons to build the final tool. In particular, it is advisable to provide a distributed access to this kind of resource, in order to guarantee a certain standardization for the tagging domain. This in turn gives us some additional advantages such as, typically, earlier error detection and, more generally, easier maintenance.

In concrete, our work is located in the conception of a tagging environment for both Spanish and Galician[1]. These projects integrate researchers from different universities and research centers, working on the same lexical resources. So, the priority of the group designing the tagging architecture was to make all programming tasks as transparent as possible. At the same time, we provide a distributed architecture allowing us to avoid the most frequent drawbacks in the context of the incremental development of taggers. In this sense, we have chosen to work in the domain of the finite automaton (*FA*) model [4, 7, 8, 9], hiding most mechanical issues, and using the WWW as privileged vehicle to share the information. This provides an increase in declarativeness and, therefore, a greater comfort for the user. Finally, this previously explained strategy allows us to benefit from the advantages of an operational model, the FA model, which is currently considered as the most efficient and general way to deal with the problem of tagging computationally.

Section 2 shows how the declarativeness of the tagger was improved without losing the core FA technology. In section 3 the overall system architecture is described, focusing on how this makes the system more user-friendly. Next, section 4 briefly introduces the most relevant facilities of the graphical interface. Finally we present our conclusions in section 5.

2. The tagger design

Actually we should speak about a *multi-tagger* as it provides each word with all its possible tags, given

Work partially supported by the Government of Spain under project HF97-223, and by the Autonomous Government of Galicia under projects XUGA10505B96 and XUGA20402B97.

[1] the native language of Galicia, the north-west region of Spain.

that we shall not be speaking about the statistical desambiguator here[2]. The following discussion will focus on the tagger developed for Spanish, a highly inflexional language with a great number of non-concatenative processes. The most outlining features:

- A highly complex conjugation paradigm, with nine simple tenses and nine compound tenses, all of which have six different persons. If we add the Present Imperative with two forms, Infinitive, Compound Infinitive, Gerund, Compound Gerund, and Participle with four forms, then 18 inflected forms are possible for each verb.

- Irregularities in both verb stems and endings. Very common verbs, such as `hacer` (*to do*), have up to seven different stems: `hac-er`, `hag-o`, `hic-e`, `haré`, `hiz-o`, `haz`, `hech-o`. Approximately 30% of Spanish verbs are irregular. We have implemented 38 groups of irregular verbs.

- Verbal forms with enclitic pronouns at the end. This can produce changes in the stem due to the presence of accents: `da` (*give*), `dame` (*give me*), `dámelo` (*give it to me*). We have even implemented forms with three enclitic pronouns, like `tráetemelo` (*you, bring it to me*). Here, the analysis has to segment the word and return four tokens.

- A highly complex gender inflection, with words with only one gender such as `hombre` (*man*) and `mujer` (*woman*), and words with the same form for both genders as `azul` (*blue*). In relation to words with separate forms for masculine and feminine, we have a lot of models:

  `autor`, `autora` (*author,authoress*); `jefe`, `jefa` (*boss*); `poeta`, `poetisa` (*poet,poetess*); `rey`, `reina` (*king,queen*) or `actor`, `actriz` (*actor,actress*).

  We have implemented 20 variation groups for gender. All the linguistic possibilities the language offers for nouns and adjectives are included, even the most infrequent and irregular.

- The inflexion of number is also highly complex, with words only being presented in singular form, such as `lunes` (*Monday*), and others where only the plural form is correct, such as `matemáticas` (*mathematics*). The construction of different forms does not involve as many variants as is the case for gender, but we can also consider a certain number of models:

  `rojo`, `rojos` (*red*); `luz`, `luces` (*light*); `animal`, `animales` (*animal*); `lord`, `lores` (*lord*); `frac`, `fraques` (*dress coat*);

  We have implemented 10 variation groups for number.

Field	Values	
Word	*The citation form present in the input text.*	
Lemma	*The canonical form of the word.*	
Category	Adjective	*With no type.*
	Adverb	Exclamative, modifier, nuclear, relative, interrogative *and* nuclear & modifier.
	Article	*With no type.*
	Conjunction	Coordinate *and* subordinate.
	Demonstrative	*With no type.*
	Indefinite	*With no type.*
	Interjection	*With no type.*
	Interrogative	*With no type.*
	Numeral	Cardinal, ordinal, partitive *and* multiple.
	Peripheral	Foreign word, formula, symbol, abbreviation, acronym *and* other.
	Preposition	*With no type.*
	Personal Pronoun	Tonic, proclitic atonic *and* enclitic atonic.
	Possessive	*With no type.*
	Punctuation Mark	Dot, comma, colon, semicolon, dash, quotes, open/close question mark, open/close exclamation mark, open/close parenthesis *and* dots.
	Relative	*With no type.*
	Substantive	Common *and* proper.
	Verb	*With no type.*
Subtype	Determiner, non-determiner *and* both.	
Gender	Masculine, feminine, both, neutral *and* non-applicable.	
Number	Singular, plural, both *and* non-applicable.	
Degree	Comparative *and* non-applicable.	
Person	First, second, third,first & third *and* non-applicable.	
Case	Nominative, accusative, dative, accusative & dative, prepositional case *and* nominative & prepositional case.	
Verbal tense	Present, preterite, co-preterite, future, post-preterite *and* non-applicable.	
Mode	Indicative, subjunctive, imperative, infinitive, gerund *and* participle.	

Table 1: Tag set

- Duplicated past participles, like `impreso` and `imprimido`, *printed.*

- Gaps in some verbs paradigms, in which some forms are missing or simply not used. For instance, meteorological verbs such as `nevar` (*to snow*) are conjugated only in the third singular person.

To achieve the goals mentioned above we implemented a well-known scheme of compilation of morphological rules into FA's. Starting out from a *class-paradigm* model we created a class for each category, and inside categories one subgroup for each group of words with the same inflexion mechanism. The result is the tagset in table 1.

Rules that are automatically compiled into FA transitions were created for each inflexional process, based on the hierarchy described. Furthermore a

[2] this facility is currently available for non-distributed versions, but it is still in a testing phase.

```
tr’aemelo
=> ["tr’ae", (V2spm02), Verb, second, sing, present, imperative,
    gender non-applicable, 2 enclitic pronouns, "traer"]
=> ["me", (Re1syy), Personal Pronoun enclitic atonic, first, sing,
    accusative & dative, masc & fem, "yo"]
=> ["lo", (Re3sam), Personal Pronoun enclitic atonic, third, sing,
    accusative, masc, "’el"]
sobre
=> ["sobre", (Vysps0), Verb, first & third, sing, present, subjunctive,
    gender non-applicable, "sobrar"]
   ["sobre", (Scms), Substantive common, masc, sing, "sobre"]
   ["sobre", (P), Preposition, "sobre"]
```

Figure 1: Example of tagger output.

process was defined to add words to the lexicon recognized by the tagger, just by situating them in their corresponding group and indicating their lemma and stem(s).

As an example, let’s consider the word `tráemelo` with one meaning: *bring it to me* and two enclitic pronouns, and the word `sobre` with three meanings: the preposition *on*, the noun *envelope* and the verb *to exceed*. The output of the tagger is shown in figure 1.

3. System architecture

At the first development stage, the efforts were focused on the *tagger* itself, without paying attention to the system interfaces. All the lexical entries were stored in plain ASCII files, and were accessed by a few scripts designed to add, remove, edit or query the entries (see Fig. 2).

As the tagger became more stable and powerful, the need appeared for a simple graphical interface for maintaining the lexical database. To achieve this goal, we wrote two interfaces in TCL/TK. Besides that, the lexical database had gown so much as not to be suitable for ASCII file storage, leading us to an implementation using a relational database. More exactly, we have used MSQL [3], dealing with 17.138 lemmas, a database that keeps growing every week.

At this point, a SQL server managed the requests from the application interfaces running in different machines, in a client-server fashion. However, we still needed to port the client part to every operating system-machine architecture combination, making maintenance a nightmare. Finally, another drawback in this development stage was the fact that there was no way the user could easily customize the interface. To overcome those limitations, the following goals

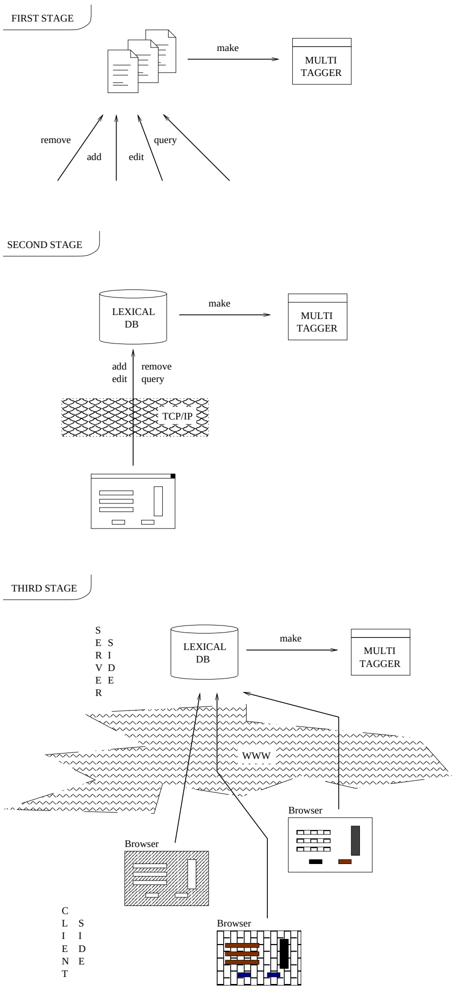
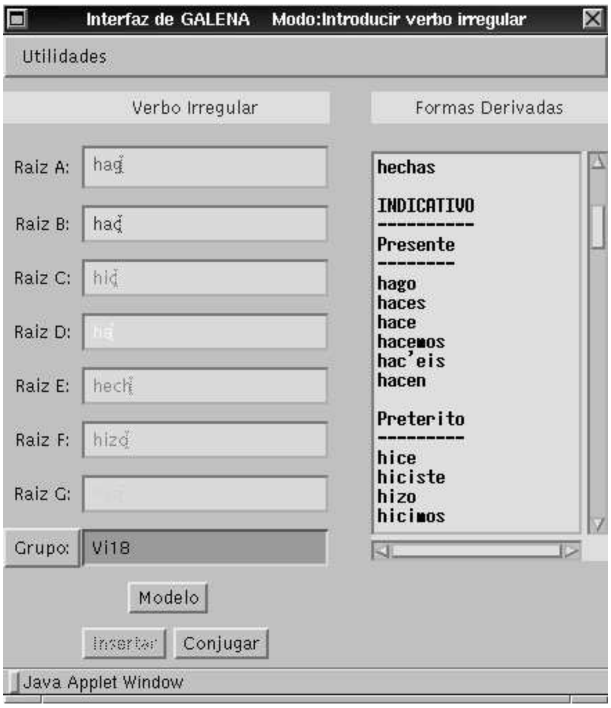


Figure 2: Stages of the system architecture.

were proposed, making the design shift to a WWW oriented one:

- Making the interface even more friendly.
- Improving system accessibility.
- The ability to use the system from any machine.
- Easiness of the interface code maintenance.

To achieve the proposed goals, we choose to keep the the relational database and to code the interface in JAVA [2, 6]. The most resource-consuming processes, the database-engine and the tagger compilation, are performed at the server-side part, while the interface is left in the client-side. By doing this, we allow users to access the system with whatever WWW browser they like the most, from their favorite machine, improving both system accessibility and easiness of use. Also, as we know, JAVA code is loaded directly from the WWW server every time the application is started, and therefore shifting to new versions of the interface is performed just by changing it in the server.



Raiz A: hag

Raiz B: hac

Raiz C: hic

Raiz D: ha

Raiz E: hech

Raiz F: hizo

Raiz G:

Grupo: Vi18

```
hechas

INDICATIVO
----------
Presente
--------
hago
haces
hace
hacemos
hac'eis
hacen

Preterito
---------
hice
hiciste
hizo
hicimos
```

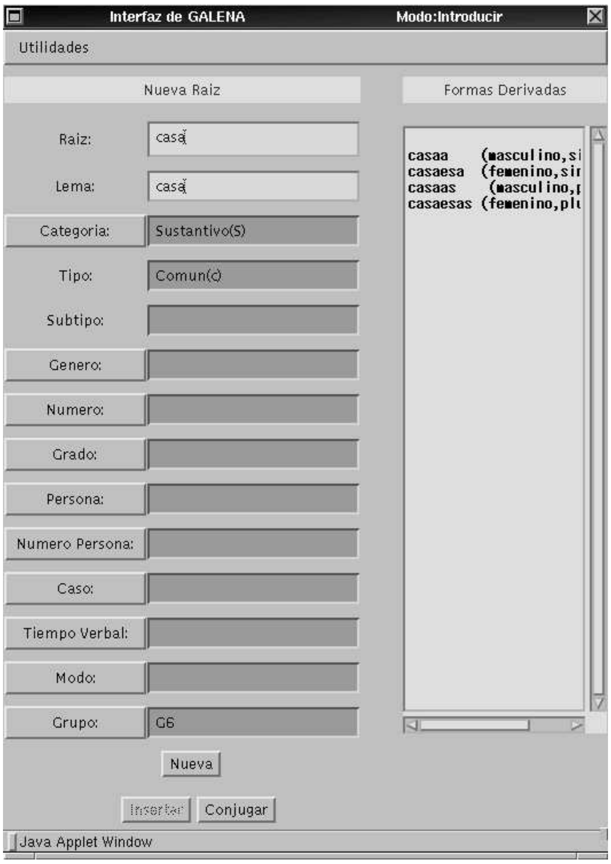
Figure 3: Inflexion of an irregular verb.

4. The graphical interface

Our goal now is to show how the graphical interface can help in maintaining the lexical database. A more detailed explanation can be found in [1]. The interface has three main windows, with three specific purposes that can be accessed from the main menu:

- Querying, editing or removing.
- Adding words with a regular or nearly regular inflexion paradigm.
- Adding irregular verbs.

Querying, editing and removing mode is shown in Fig. 5. As one would expect we can edit or remove whatever word results from a query. What is remarkable is that we can restrict queries by the values of the attributes of the tags or/and the categories and subgroups of the words, as well as by their lemmas and stems. The goal is to allow users to find words related in any morphological sense.



Raiz: casa

Lema: casa

Categoria: Sustantivo(S)

Tipo: Comun(c)

Subtipo:

Genero:

Numero:

Grado:

Persona:

Numero Persona:

Caso:

Tiempo Verbal:

Modo:

Grupo: G6

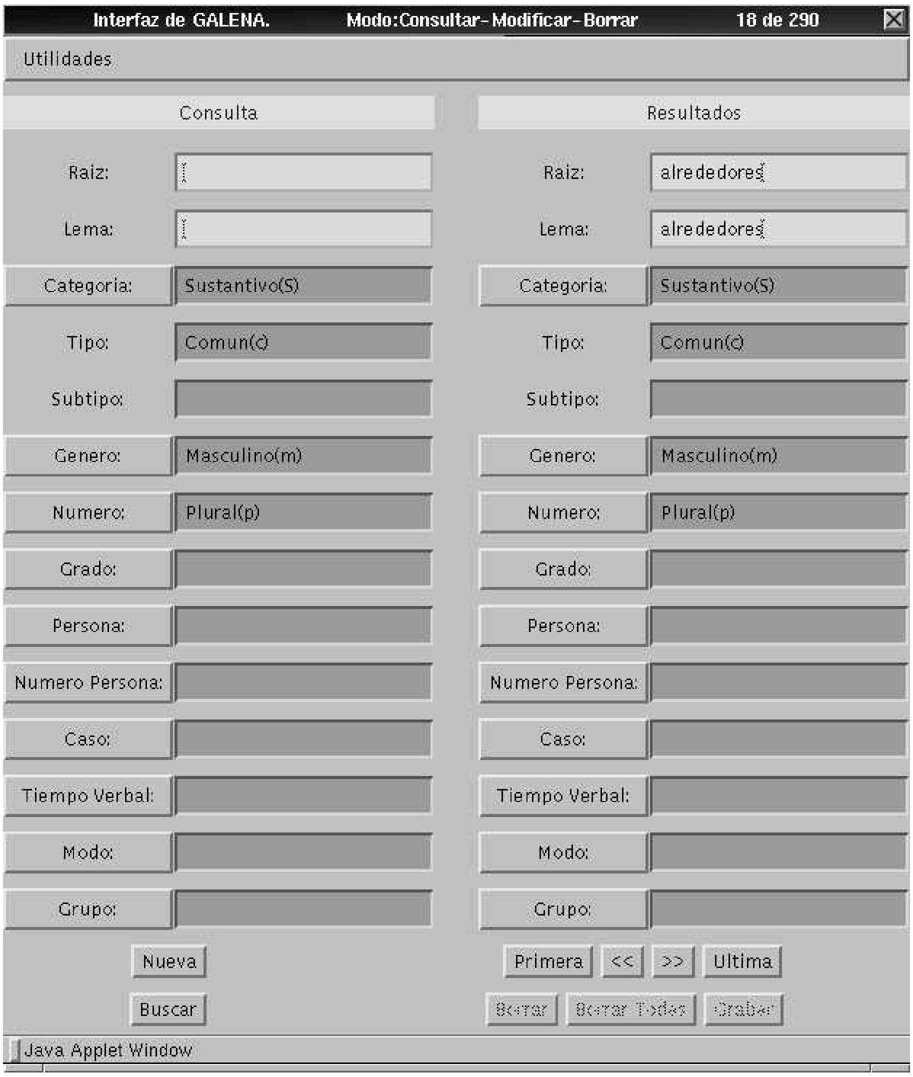
```
casaa     (masculino,si
casaesa   (femenino,sir
casaas     (masculino,p
casaesas  (femenino,plu
```

Figure 4: Example of an attempt to introduce a word in the wrong group.

As an example, in Fig. 5 the user has made the following query:

> “Give me all the common nouns with masculine gender and plural number”

The left-hand-side of the window shows the query (`Consulta`), and the right-hand-side the results



Interfaz de GALENA. Modo:Consultar-Modificar-Borrar 18 de 290

	Consulta	Resultados
Raiz:		alrededores
Lema:		alrededores
Categoria:	Sustantivo(S)	Sustantivo(S)
Tipo:	Comun(c)	Comun(c)
Subtipo:		
Genero:	Masculino(m)	Masculino(m)
Numero:	Plural(p)	Plural(p)
Grado:		
Persona:		
Numero Persona:		
Caso:		
Tiempo Verbal:		
Modo:		
Grupo:		

Figure 5: Example of query against the lexical database.

(`Resultados`). In Fig. 3 the user is in the adding irregular verbs mode. The example is the result of adding the highly irregular verb `hacer` (*to do*) which belongs to the subgroup VI18 in our classification, and has seven stems. What appears in the right half of the window are all the inflected forms of the verb. Another way in which the graphical interface can help the user can be seen in the next example.

When a user selects the category, subgroup, lemma and stem(s) of a word, he can then check if something is wrong by asking the system to show all the inflected forms. As an example, in Fig. 4 the user selected the wrong subgroup and checked out that the inflected forms were not what he expected.

Finally, due to the complexity of the irregular verb adding mode, another facility is supplied. It simply consists of the possibility of asking the system for the model of a subgroup. This is intended to help the user when he or she wants to add a new irregular verb but is not sure which paradigm it belongs to.

5. Conclusions

The design of taggers should respond to constraints of efficiency, safety and maintenance. The choice of the FA model as operational formalism assures computational efficiency. Safety is guaranteed by the separation between this operational kernel, and the high-level descriptive formalism.

In order to achieve the need for maintenance, a graphical interface for a distributed system following a client-server model was developed. The choice in this case was SQL plus WWW. Both involve well-proved and widely used techniques that provide: friendly graphical user interfaces, non-centralized execution of the system and world-wide accessibility.

The work described is not a closed research-line. It represents only a first approach to the problem of tagging. However if we look at both the efficiency of the resulting taggers and the satisfaction of the final users, we can see that we are going in the right direction.

Looking ahead from these promising results it clearly follows that work has to be done to design the parser generation tools and then the whole system with this distributed architecture.

6. References

[1] L. Docampo. Explotación remota del sistema galena. Master's thesis, Universidad de Informática, A Coruña, September 1997.

[2] H. Hanh. *Internet. Manual de referencia*. McGraw-Hill/Iberoamericana de Espña, S.A., 1997.

[3] Hughes Technologies Pty Ltd. *Mini SQL. A Lightweight Database Engine*, 1.0.1 edition, Jan 1996.

[4] K. Koskenniemi. Compilation of automata from morphological two-level rules. In *Proc. of the* 5^{th} *Scandinavian Conference of Computational Linguistics*, pages 143–149, Helsinki, Finland, 1985.

[5] Tim Ritchey. *Programming with Java!* New Riders Publishing, Carmel, IN, USA, 1996. CD-ROM includes: Java developers kit for Windows 95, Windows NT, and Sun Solaries 2.x, Java applet tools, example applets from the text.

[6] Tim Ritchey. *Programando con Java*. Prentice Hall HispanoAmericana, Mexico, DF, Mexico / Nueva York, NY, USA, 1997. Spanish translation of [5].

[7] G. Ritchie. On the generative power of two-level morphological rules. In *European Chapter of the ACL*, pages 51–57, Manchester, 1989.

[8] G. Ritchie, D. Pulman, Stephen, A.W. Black, and G.J Russell. *Computational Morphology*. The MIT Press, Cambridge, Massachusetts, U.S.A., 1991.

[9] E. Roche and Y. Schabes. Deterministic part-of-speech tagging with finite-state transducers. *Computational Linguistics*, 21(2):227–253, 1995.