AN ASSISTANT TOOL FOR VERSE-MAKING IN BASQUE BASED ON TWO-LEVEL MORPHOLOGY

AUTHOR: Bertol Arrieta

AFFILIATION: University of the Basque Country

E-MAIL: jiparkob@si.ehu.es

AUTHOR : Iñaki Alegria

AFFILIATION : University of the Basque Country

E-MAIL: i.alegria@si.ehu.es

AUTHOR : Xabier Arregi

AFFILIATION: University of the Basque Country

E-MAIL: jiparipx@si.ehu.es

CONTACT ADDRESS: 649 Postakutxa / 20080 Donostia / Basque Country

FAX NUMBER: 34-943219306

PHONE NUMBER: 34-943015064

Abstract

In this paper we present a specialised word generator, which has been aimed as an assistant tool for Basque troubadours. Such a tool allows verse-writers to generate all the words that match with a given word termination. We coped with some interesting aspects, i.e. the dimension of the generated list and the need of establishing an order of relevance among the listed items.

This work can be seen as a way of reusing computational linguistic tools in the context of the Basque cultural means of expression. The technical foundations of this tool lie on a two-level morphological processor. The way in which words must be generated (starting from the end of the word) leads us to inverse the generation process.

‘Bertsolaritza’: What Is It?

‘Bertsolaritza’ (Basque term for verse-making) is an oral or written literature form with old tradition and great popularity in the Basque Country. Similar forms are manifested in other countries like Cuba.

While the written mode is similar to poetry, the oral mode has a peculiarity: troubadours sing verses without knowing previously the theme. In other words, a theme is given to the troubadour and in a few seconds they have to think a set of verses adjusted to the theme and sing them. These verses must hold the formal conditions (measurement and rhyme) of the discipline.

Here it is an example of a verse in Basque. In this occasion, the troubadour was asked for singing a verse, which would describe the word 'burua' (head). Egaña (the troubadour) spent only 20 seconds thinking this verse, before beginning to sing it (see the translation in Fig. 3). Note that even sentences rhyme (see Fig. 1) and that it has a fixed measurement (see Fig. 2).

‘Batzutan da argia
bestetan il**una**
batzutan pilakoa
bestetan ast**una**
kanpotik ilea du
barruan gar**una**
lepo gainean denok
daukagun lag**una**
gehiegik gutxiegi
erabiltzen d**una**’

Figure 1 (Egaña, 97)

‘Ba-tzu-tan da ar-gi-a	(7 syll.)
bes-te-tan i-lu-na	(6 syll.)
ba-tzu-tan pi-la-ko-a	(7 syll.)
bes-te-tan as-tu-na	(6 syll.)
kan-po-tik i-le-a du	(7 syll.)
ba-rru-an ga-ru-na	(6 syll.)
le-po gain-e-an de-nok	(7 syll.)
dau-ka-gun la-gu-na	(6 syll.)
gehi-e-gik gu-txi-e-gi	(7 syll.)
e-ra-bil-tzen du-na’	(6 syll.)

Figure 2 (Egaña, 97)

' Sometimes is clever
sometimes is dark
sometimes is ordinary
sometimes is lazy
it is covered with hair
brain is in
the friend we all have
on the neck
which is used too few times
by too many people.'

Figure 3 (Egaña, 97)

It has to be taken into account that this is an improvised verse. The verse-maker did not know previously the word he had to describe. He had to think all the verse at the moment and make a verse adjusted to a measurement and where even sentences rhymed.

Let us see another example. In this case, Lazkao Txiki (1926-1993) had to think 3 verses after an object was given to him. Lazkao Txiki was a single man, small in height and he was in his 60's when he was asked for singing this verses. The object given was a mirror.

‘Ispilu txiki eder polit bat	‘Is-pi-lu txi-ki e-der po-lit bat (10)	‘A little pretty mirror
didate aurrera eka**rri**	di-da-te_au-rre-ra e-ka-rri (8)	has been brought to me.
ekarri dunak jakingo zuen	e-ka-rri du-nak ja-kin-go zu-en (10)	The one who brought it
mutilzar zar honen be**rri**	mu-til-zar zar ho-nen be-rri (8)	must know about this old confirmed bachelor.
gazte dan arte hau izaten da	gaz-te dan arte hau i-za-ten da (10)	While you are young , it (the mirror) is
gazte guztien pozga**rri**	gaz-te guz-ti-en poz-ga-rri (8)	usually a reason for joy,
zartutakoan ez da beiratzen	zar-tu-ta-ko-an ez da bei-ra-tzen (8)	but when you get older you do not look at it
hau gaztetan bezain sa**rri**	hau gaz-te-tan be-zain sa-rri (10)	as often as when you were young.
Hola jarrita bota behar dit	Ho-la ja-rri-ta bo-ta be-har dit (10)	As I am here, I will sing you
bertso koxkor bat edo **bi**	ber-tso kox-kor bat e-do bi (8)	one or two little verses
behingoan jarri geranez gero	behin-go-an ja-rri ge-ran-ez ge-ro (10)	since we are finally
biok aurpegiz aurpe**gi**	bi-ok aur-pe-giz aur-pe-gi (8)	face to face.
neri begira hortik daduzkak	ne-ri be-gi-ra hor-tik da-duz-kak (10)	There you have, looking at me,
alperrikako bi be**gi**	al-pe-rri-ka-ko bi be-gi (8)	those two useless eyes.
hik ez nauk noski ni ikusiko	hik ez nauk nos-ki ni i-ku-si-ko (10)	You might not see me, of course,
baina nik ikusten haut **hi**	bai-na nik i-kus-ten haut hi (8)	but I can see you very well.
Neri begira jarrita motel	Ne-ri be-gi-ra ja-rri-ta mo-tel (10)	Why are you
zergatik hago horr**ela**?	zer-ga-tik ha-go ho-rre-la? (8)	looking at me like that?
ta pentsatzen det aspalditxotik	ta pen-tsa-tzen det as-pal-di-txo-tik (10)	I feel as if I knew you
ezagututzen haut**ela**	e-za-gu-tu-tzen hau-te-la (8)	since a long time ago.
mutilzarraren moko horrekin	mu-til-zarr-a-ren mo-ko ho-rre-kin (10)	With this bachelor-*beak*
ez dek ematen mot**ela**	ez dek e-ma-ten mo-te-la (8)	you do not look bad at all
azal zimurtzen ari haiz motel	a-zal zi-mur-tzen a-ri haiz mo-tel (10)	but you are getting wrinkled, man,
Lazkao Txiki bez**ela**.’	Laz-ka-o Txi-ki be-ze-la.’ (8)	just like Lazkao Txiki. ’

Figure 4 (Lazkao Txiki, 92) Figure 5 (Lazkao Txiki, 92) Figure 6 (Lazkao Txiki, 92)

In the first verse, Lazkao Txiki is singing to the mirror, but in the second one and in the third one, he has the brilliant idea to sing to the figure of the mirror (himself) (see Fig. 6). Note that the verses have a fixed measurement (see Fig. 5) and that even sentences rhyme (see Fig 4).

As it can be seen, this verse-making task is quite difficult, and more if the verses are improvised. So great expertise is required. Because of that, some schools devote to teach how to improvise this type of verses. As far as we concern, the tool here presented could be quite useful in the verse-schools.

Since some decades, an oral verse-making competition is organized in the Basque Country every four years. The high diffusion of this event (thousands of Basque people follow this competition with great interest live or from TV) is a clear demonstration of the importance of this discipline.

From this background was formed the idea of designing the tool here presented. Noting that one of the most difficult things troubadours must learn is to find words that rhyme, we did an application tool for that purpose. We hope that such an application will be a useful assistance-tool in the task of finding rhymes, namely for those inexperienced troubadours.

Obtaining the Inverse Generator

To get an inverse generator tool, a morphological analyzer/generator for Basque, integrated in several tools as spelling correctors and ICALL systems (Maritxalar et al., 97) and developed a few years ago (Alegria et al., 96), was reused. The morphological description is based on the Koskenniemi's Two Level Morphology Model (Koskenniemi, 83).

Koskenniemi's Two-Level Morphology Model

The two-level system is based on two main components:

1. A lexicon where the morphemes (lemmas and affixes) and the possible links among them (morphotactics) are defined. The lexicon is divided into different sublexicons and each lexicon-entry specifies its morphotactical information bymeans of a continuation class which is

a set of sublexicons (see Fig. 7). Combining sublexicons (nodes) and continuation classes (arcs) the graph of morphotactics is defined.

SUBLEXICONS	CONTINUATION CLASSES
*Regular_verbs stay RV_CC try RV_CC believe RV_CC ask RV_CC … *Irregular_verbs take IV_CC get IV_CC see IV_CC fly IV_CC buy IV_CC … *Third_person s NULL *Regular_past ed NULL	*RV_CC Third_person; Regular_past; … *IV_CC Third_person; …

Figure 7: Example of some sublexicons and some continuation classes

2. A set of rules which controls the mapping between the lexical level and the surface level (changes at surface level when morphemes are linked) due to the morphonological transformations (morphophonemics).

To control these morphonological transformations with the sublexicons and continuation classes above, we would need some rules. Here it is one of them, which will control the third person of the verbs ending by ‘y’ and having a consonant before this ‘y’ character. In this case this ‘y’ character will be converted to ‘ie’.

Y:i ⇔ Cons _ +: 0:e s;

So with these sublexicons, these continuation classes (see Fig. 8) and the necessary rules we would obtain the results shown in Fig. 9.

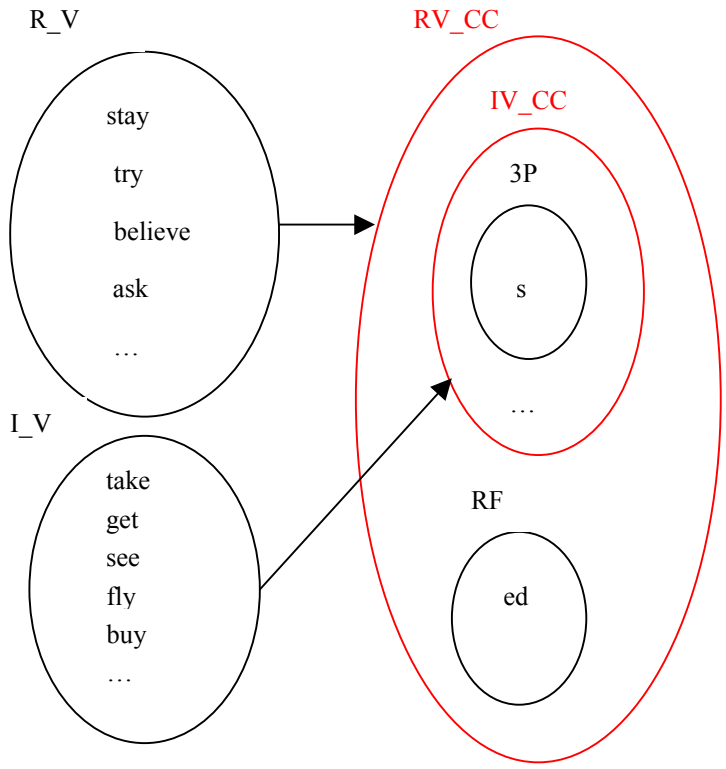

Figure 8: Sublexicons and continuation classes

stays tries (The rule before is applied here) believes asks … stayed tried believed asked …	takes gets sees flies (The rule before is applied here) buys …

Figure 9: Words generated

Reversing of the Morphological Description

In order to get our inverted morphological analyzer/generator for Basque language we needed to reverse this morphological description. The goal is to build an inverted morphological generator for Basque language, which will control the order of the proposals according to their suitability for being a rhyme. The inverted morphological generator will obtain all the possible forms corresponding to a known ending, instead of generating the possible forms corresponding to the beginning. We took into account two choices to reverse the morphological description.

1. Manipulating the Automata

This first option consists on manipulating the automata that are created from the morphological description of the Basque language. The algorithm of this manipulation would be as follows:

-converting the final states to initial states

-converting the initial states to final states

-changing the direction of the transitions

Let us see what automaton would be if the rule described before (Y:i⇔Cons_ +: 0:e s) is considered (see Fig. 10 and Fig. 11) and what kind of inverted automaton would be obtained if the algorithm mentioned is applied (see Fig. 12).

	y i	y =	Cons Cons	+ =	0 e	s s	= =
1:	0	1	2	1	1	1	1
2:	3	6	2	1	1	1	1
3.	0	0	0	4	0	0	0
4.	0	0	0	0	5	0	0
5.	0	0	0	0	0	1	0
6:	0	1	2	7	1	1	1
7:	0	1	2	1	8	1	1
8:	0	1	2	1	1	0	0

: → final states
. → non final states

Figure 10: The table of the automaton

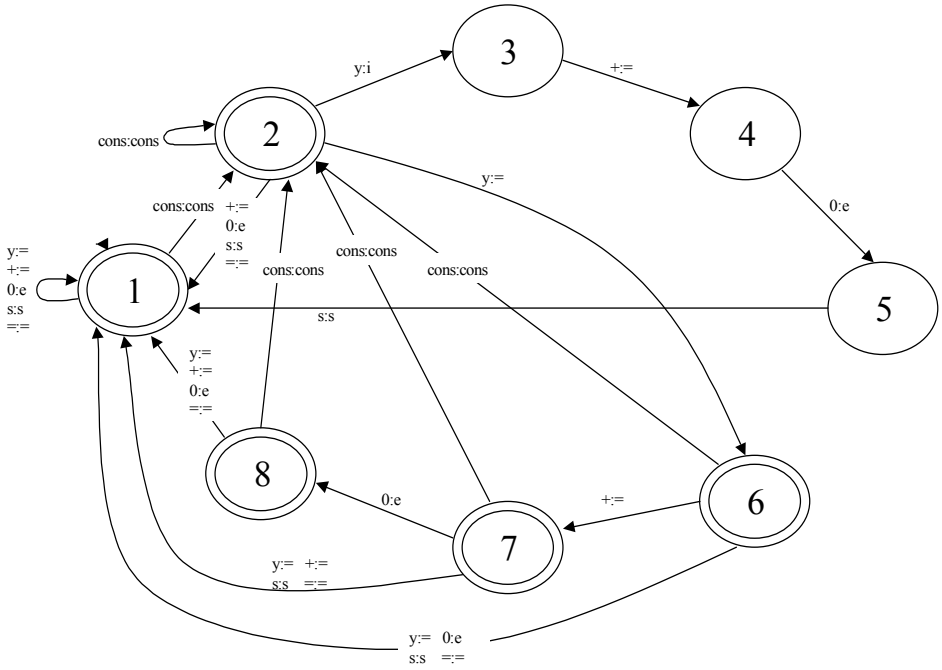

Figure 11: The resulting automaton

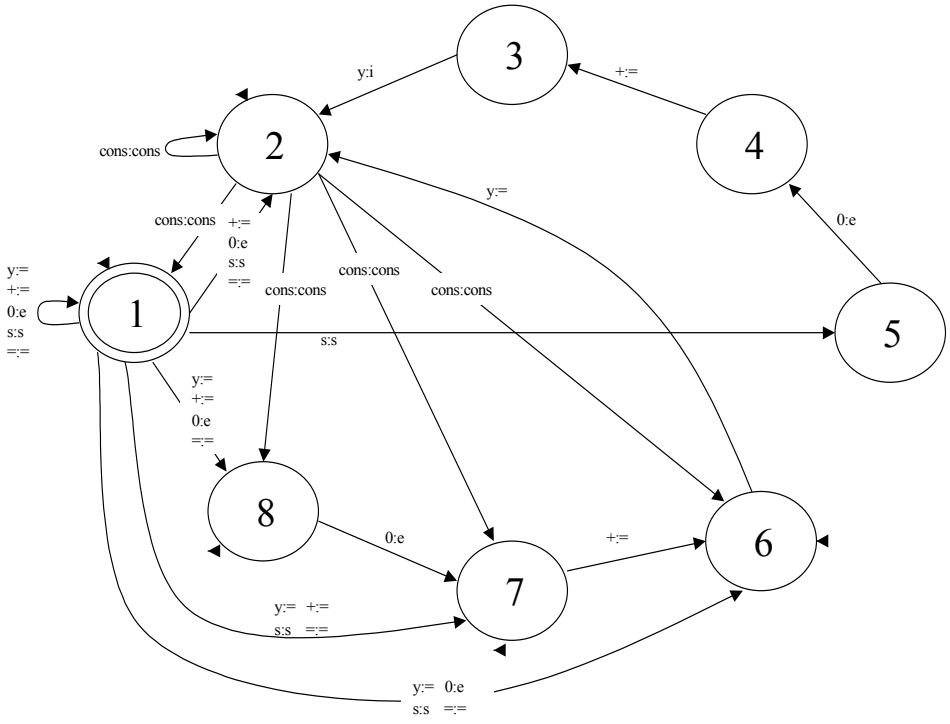

Figure 12: The inverted automaton

This option looked initially good, because we did not need to manipulate the lexicon and the rules; we only had to manipulate the automata. But analyzing this process, we realized that after applying our inversion algorithm the Deterministic Finite Automata (DFA) would become Non-Deterministic Finite Automata (NDFA) and trying to re-convert the NDFA in DFA would cause a combining explosion, since this transformation has an exponential order (Hopcroft and Ullman, 1979).

2. Manipulating the Morphological Representation

The second option consists on manipulating and reversing the lexicon and the rules directly before using the compilers (Karttunen and Beesley, 1992) (Karttunen, 1993). So, this approach involves the implementation of the programs that invert the lexicon, the morphotactics and the phonological rules in an automatic way.

Considering the problems of the first choice, to develop the second method was decided. This process was divided into three steps:

2.1 Reversing the lexicon

This task deals with the inversion of all the morphemes. The order of the characters inside the morphemes is inverted in all the sublexicons (see Fig. 13).

SUBLEXICONS

```
*Regular verbs
        yats      RV_CC
        yrt       RV_CC
        eveileb   RV_CC
        ksa       RV_CC
        …
*Irregular verbs
        ekat      IV_CC
        teg       IV_CC
        ees       IV_CC
        ylf       IV_CC
        yub       IV_CC
        …
*Third_person
        s         NULL
*Regular_past
        de        NULL
```

Figure 13: Sublexicons with the inverted lexicon-entries

2.2 Converting the continuation classes in 'backward classes'

The base of the morphotactics in the two-level model are the continuation classes (Koskenniemi, 1983). A script to convert the continuation classes in 'backward classes' was programmed so as to get a group of morphemes that goes **before** an inverted morpheme. It may look easy, but it has some problems. Lexicons containing final classes have to be defined as root lexicons, and consequently the backward class of the original root lexicons must be null.

The algorithm is quite simple. If the lexicon-entries (morphemes) of sublexicon_i have a continuation class (which contains also one or more sublexicons in), each sublexicon of the continuation class would have the sublexicon_i into its backward class. The problem begins when some entries of one sublexicon have one continuation class and others have another continuation class, in other words, when one sublexicon has two or more continuation classes. This would mean each sublexicon would have to be separated in so many pieces as continuation classes it had. Thus, it is going to be possible to get the correct group of entries in each backward class.

For instance, the sublexicon SUBLEXICONi would have to be divided into two sublexicons, SUBLEXICON__1 and SUBLEXICON__2, because different continuation classes are in (see Fig. 14), and after, we would have to invert the entries and give to each entry a backward class (see Fig. 15).

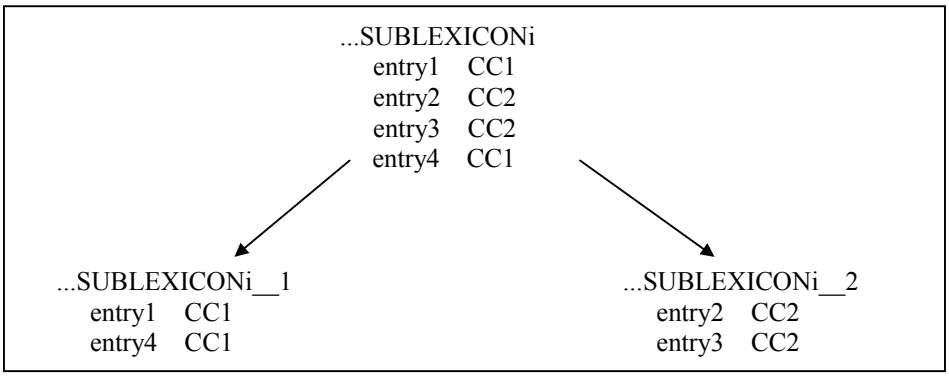

Figure 14: Sublexicons separated depending on the continuation classes of each lexicon-entry

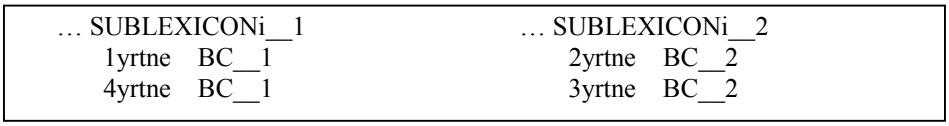

Figure 15: Separated sublexicons, with inverted lexicon-entries and backward classes

Sublexicons in the example mentioned before (see Fig. 7) would be like in the Fig. 16.

SUBLEXICONS		*BACKWARD CLASSES*
*Regular_verbs		
yats	NULL	*TP_BC (third_person backward class)
yrt	NULL	Irregular verbs;
eveileb	NULL	Regular verbs;
ksa	NULL	
…		
		*RP_BC (regular_past backward class)
*Irregular_verbs		Regular verbs;
ekat	NULL	
teg	NULL	
ees	NULL	
ylf	NULL	
yub	NULL	
…		
*Third_person		
s	TP_BC	
*Regular_past		
de	RP_BC	

Figure 16: Sublexicons with the lexicon-entries inverted and their backward classes

2.3 Reversing the rules

Each part has to be analyzed to decide what has to be inverted (see Fig. 17).

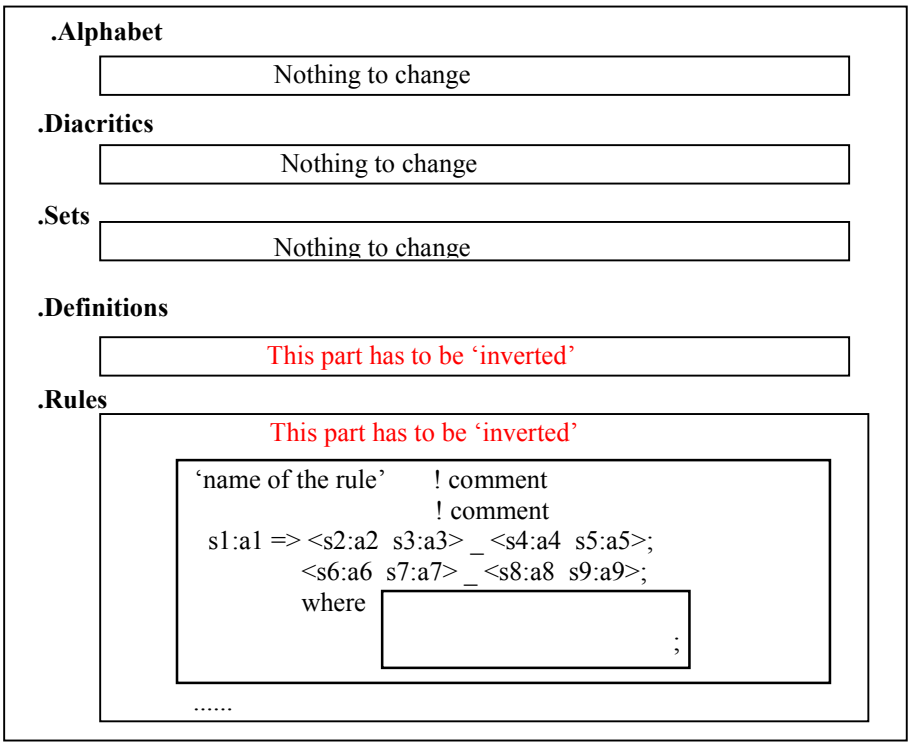

Figure 17: Rules file in the Xerox tools

In the case of **Rules** the name and the comments have not to be changed. Let us see how the own rule has to be changed.

The rules are expressed as follows:

<correspondence> <operator> <left context> _ <right context>

To reverse the rules only contexts have to be changed, interchanging between them and reversing each one (see Fig. 18). The contexts are regular expressions and it is necessary to distinguish between data (to be reversed) and regular operators and reserved characters.

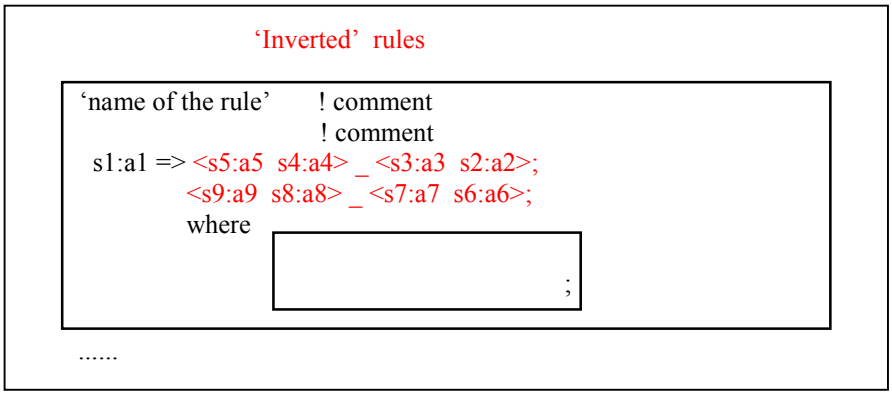

Figure 18: Inversion of each rule

The clause 'Where' can not have changes, because this clause makes reference to the correspondence of the rule (s1:a1) and this part does not change.

Besides, in the rules, different operations may appear. So, what has to be done in each operation has to be decided (see Fig. 19).

-complement (~): ~a:b ===> ~a:b NOTHING TO CHANGE
-term complement (\): \a:b ===> \a:b NOTHING TO CHANGE
-containment ($): $a:b ===> $a:b NOTHING TO CHANGE
-kleene star (*): (a:b)* = (a:b a:b a:b) ===> (a:b a:b a:b) = (a:b)*
NOTHING TO CHANGE
-kleene plus (+): (a:b)+ ===> (a:b)+ NOTHING TO CHANGE
-ignore (/): (a:b c:d) / e:f = (e:f)* a:b (e:f)* c:d (e:f)*
===> (e:f)* c:d (e:f)* a:b (e:f)* = (c:d a:b) / e:f
The Ignore operation HAS NOT TO BE CHANGED,
only the concatenation, as we are seeing now.
-concatenation( ): a:b c:d ===> c:d a:b THERE HAVE TO BE CHANGED
-union (|): a:b | c:d ===> a:b | c:d NOTHING TO CHANGE
-intersection (&): a:b & c:d ===> a:b & c:d NOTHING TO CHANGE
-minus (-): a:b - c:d ===> a:b - c:d NOTHING TO CHANGE

Figure 19: Operations that could appear in the rules

As the concatenation operation is the only one to change, set and definitions have not to be changed. A gap is a concatenation operation so when one is found, both sides of the gap would have to be exchanged. If no gap is found, nothing changes.

Let us see a real example. The rule mentioned before, 'Y:i ⇔ Cons _ +: 0:e s; ' will be converted to ' y:i <=> s 0:e +: _ Cons;'. So, with these sublexicons, these backward classes (see Fig. 20) and the inverted rules these words would be created (see Fig. 21).

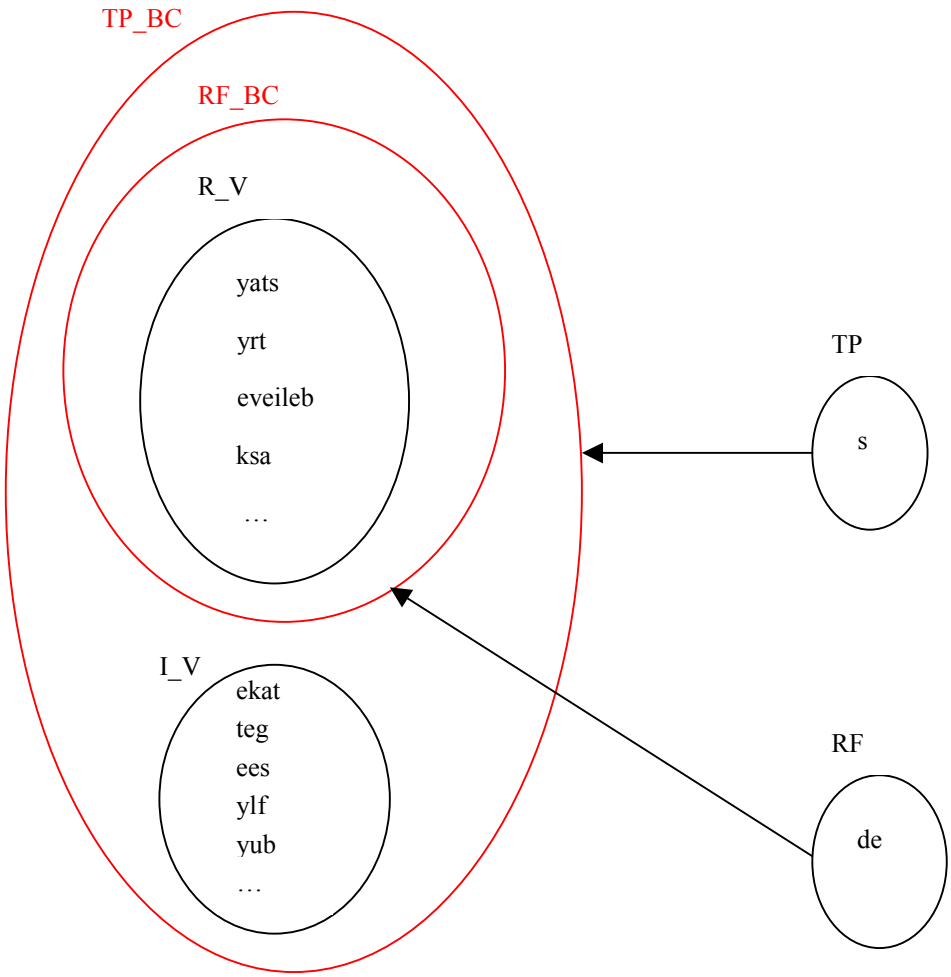

Figure 20: Sublexicons with inverted lexicon-entries and backward classes

syats seirt seveileb sksa … deyats deirt deveileb deksa …	sekat steg sees seilf syub

Figure 21: Words generated

Application to 'Bertsolaritza': Finding Words that Rhyme with an Ending

Once the inverted analyzer/generator for Basque language was developed, we tried to reuse it in an application that got the rhymes departing of a final part of a word. We needed to invert the character sequence given by the user and then launch the generation with our inverted morphological generator tool.

But to start the generation process the lexicon form of the ending given by the user is needed. So we had to convert this word-ending (which was of course at the surface form) to the lexicon form. The morpheme analyzer was introduced for this aim.

This morpheme analyzer is a little lexicon joined with the normal ('non-inverted') rules. This little lexicon recognizes all kind of morphemes added to any characters, so that it is going to be possible to analyze any word-ending given by the user.

Another question to front was the infinite generation. The Basque language has two peculiar declensions '–en' and '–ko' which have the possibility to join with another declension '-a' and get an endless generation. For instance, the Basque word 'ama' means in English 'mother'. If the morpheme '-en' is added, 'amaren' is got (it means "mother's" in English and it is the genitive form). Joining the morpheme 'a' then, results in 'amarena', that means "mother's", but in a direct object sense. And these two additions can be done as much as is wanted. Thus, all these forms can be obtained:

ama+en → amaren

ama+en+a → amarena

ama+en+a+en → amarenaren

ama+en+a+en+a →amarenarena

ama+en+a+en+a+en → amarenarenaren

ama+en+a+en+a+en+a →amarenarenarena

… and so on.

So, let us think that the user writes 'en' as an ending character sequence. It would be an endless list unless it is cut somewhere. In such a manner, two rules were created to cut this endless generation and to show only one of these forms (the first one, which is the less complex form).

k:k /<= (E: | E: t: a:| t: a:) k: o: (=:=)* (E: | E: t: a:| t: a:) _ o;

M:n /<= (E: M: | a: r: e: M:| r: e: M:) (=:=)* (E: | a: r: e: | r: e:) _ ;

Besides, another rule was implemented not to allow more than two suffixes one behind the other.

%+:= /<= %+: %+: (=:=)+ %+: (=:=)+ _ ;

After these problems were solved we realized that the list generated was still too long with almost all the word-endings. So, in order to improve the usefulness of the application, we considered necessary to face this problem. Two solutions were implemented:

1. Establishing of a kind of categorization or class-partition among the morphemes, so that only one example (representative of the class) is returned when all the elements of the class are suitable to be shown. For instance, if the input is 'ed', instead of returning all the regular verbs with the past form ‘ed’ added (too long!)

stay + ed → stayed

try + ed → tried

believe + ed → believed

ask + ed → asked

...

the application will return only one example and a short explanation:

TRY+ ed --> tried (REGULAR VERB + ed)

2. Returning of words sorted in the order that verse-makers appreciate more. The quality of the rhyme is generally better considered if the word is not composed or declined. At any rate, using compose or declined words is mainly bad considered when more than one word with the same structure are used to get the rhyme in the same verse (Amuriza, 1981). So, words that are not composed or declined are shown first in our application, but immediately after, the categorization of words with the same structure is shown. In the example above, our application would show first words like ‘red’, ‘bed’,… and after the categorization (regular_verb + ‘ed’) of words like ‘tried’, ‘believed’, … that have the same structure, would be shown.

Finally, we had to make a Tcl-Tk screen to show all the output generated. And as we got inverted words with our ‘inverted’ generator tool, we had to reverse these words before

showing them. In this way, the tool here presented returns all the Basque words that have the same final as the sequence of characters given by the user. The result is that the application finds all the words that rhyme with the word-ending given by the user.

Conclusions and Future Improvements

Basque is a pre-Indo-European language with an unknown origin and it is quite different from the surrounding European languages. The declension of the Basque language has fourteen different forms for each singular, plural and undefined form. All of these forms are added at the final of the words. Besides, it is an agglutinative language, which accepts morphemes to be added to other morphemes. These characteristics show us the relevance of the final parts of the Basque words. That reason leads us to think that the inverted morphological analyzer/generator will be useful for different applications. We have found an interesting utility for such a generator in the world of the 'bertsolaritza'. Given that final parts of words (rhymes) are very important in verses, the inverted morphological analyzer/generator can be an important assistant tool for writing verses. Furthermore, an automatic method for inverting the morphological description has been defined. Such a method can be reused in any other language, always starting from a two-level description. The only thing to be taken into account is that words with the same ending in Basque always rhyme, but not in other language like English or French. In these languages the pronunciation of the characters is different depending on the characters around. 'Asked' and 'red', for instance, have the same ending ('ed') but they do not rhyme.

Here there are the future works we have considered i) to improve the categorization; ii) to return words with the assonance rhyme; iii) to deal with semantics in the selection module, in order to improve the order of presentation, and iv) to publish the application on a web site.

Acknowledgements

We would like to thank Xerox for letting us use their tools, and specially to Lauri Karttunen.

References

Alegria I., Artola X., Sarasola K., Urkia M. (1996) *Automatic Morphological Analysis of Basque. Literary and Linguistic Computing 11 (4): 193-203.* Oxford University Press. 1996.

Amuriza X. (1981) *Hiztegi Errimatua, Hitzaren Kirol Nazionala*. Bizkaiko Bertsozale Elkartea, 1981, 1993 (2nd ed.).

Egaña A. (1997) *Bertsolari Txapelketa Nagusia 97.* Euskal Herriko Bertsozale elkartea. Elkarlanean Fundazioa, 1998.

Hopcroft John E., Ullman Jeffrey D. (1979) *Introduction to Automata Theory, Languages, and Computation.* Addison-Wesley Publishing Company.

Karttunen L. and Beesley K. R. (1992). *Two-Level Rule Compiler.* Xerox ISTL-NLTT-1992-2.

Karttunen L. (1993). *Finite-State Lexicon Compiler.* Xerox ISTL-NLTT-1993-04-02.

Karttunen L. (1994). *Constructing Lexical Transducers,* Proc. of COLING´94, 406-411.

Koskenniemi K. (1983). *Two-level Morphology: A general Computational Model for Word-Form Recognition and Production*, University of Helsinki, Department of General Linguistics. Publications nº 11.

Lazkao Txiki (1992). *Lazkao Txiki*. Sendoa produkzioak. CD format.

Lekuona et al. (1980) *Bertsolaritza*. Jakin 14. eta 15. Donostia.

Maritxalar M., Diaz de Ilarraza A., Oronoz M. (1997) *From Psycholinguistic Modelling of Interlingua to a Computational Model*. Proc. Of CONLL97 Workshop
(ACL Conference). Madrid 1997.