OAL[1]: A NLP Architecture to Improve the Development of Linguistic Resources for NLP

Javier Couto*†, Helena Blancafort*‡, Somara Seng*, Nicolas Kuchmann-Beauger*ǂ, Anass Talby*, Claude de Loupy*†

* Syllabs
15, rue Jean Baptiste Berlier, 75013 Paris, France
{jcouto, blancafort, seng,beauger,talby,loupy}@syllabs.com

† MoDyCo
CNRS - Université Paris X
200, Av. de la République, Nanterre, France

‡ Universitat Pompeu Fabra
Roc Boronat 138, 08018 Barcelona, Spain

ǂ Laboratoire MAS - Ecole Centrale Paris
Grande Voie des Vignes, Châtenay-Malabry, France

Abstract

The performance of most NLP applications relies upon the quality of linguistic resources. The creation, maintenance and enrichment of those resources are a labour-intensive task. In this paper we present the NLP architecture OAL, designed to assist computational linguists in the whole process of the development of resources in an industrial context: from corpora compilation to quality assurance. To add new words more easily to the morphosyntactic lexica, a guesser that lemmatizes and assigns morphosyntactic tags as well as inflection paradigms to a new word has been developed. Moreover, different control mechanisms are set up to check the coherence and consistency of the resources. Today OAL manages resources in five European languages: French, English, Spanish, Italian and Polish. Chinese and Portuguese are in process. The development of OAL has followed an incremental strategy.

1. Introduction

One key issue in Natural Language Processing (from now on NLP) is related to the quality of the linguistic resources (e.g. morphosyntactic lexicons, semantic lexicons, specialized corpora) that are at the heart of NLP software. The creation, maintenance an enrichment of those linguistic resources are labour-intensive tasks, especially when no tools are available. Very often, in industrial NLP contexts, the linguist develops lexicographic resources without any specific tool but just using a text editor and several processing tools such as web crawlers, text content extractors (scrapers), tools to import and export information, tools to connect to resources such as GeoNames or Wikipedia or engines such as Yahoo! Boss, and scripts to generate inflections and to resolve text encoding problems. Thus, the information is often stored in textual format. Text file manipulation with Unix commands or scripts written in PERL or Python are a common practice.

To improve of the development of linguistic resources and provide quality assurance, Syllabs has designed OAL, an architecture to provide a user-friendly environment to develop lexicographic resources. The linguists are assisted during the whole process –from corpora compilation to the quality check of the resources. Corpora may be acquired from the web by focused crawling or from selected RSS feeds. Quality assurance is provided by several quality control mechanisms such as regression tests, automatic and semi-automatic procedures to evaluate the consistency of each linguistic resource. Today OAL manages resources in five European languages: French, Esnglish, Spanish, Italian and Polish. Chinese and Portuguese are in process. German will be included in the near future. From the beginning, we wanted the OAL platform to be language independent, even if we know that it is impossible to define a single formalism for any language in the world and that it will be necessary to add new features for new languages. The ongoing integration of Polish resources proves that the platform (i.e. the formalisms and tools developed) is not only useful for Romance languages or English. Chinese is a major challenge for OAL, in particular for the SylLex formalism (see section 5).

For now, the architecture and all the tools and resources were designed and developed for internal use at Syllabs. Thus, resources definitions and tools are proprietary. The design and development of OAL are carried out following an incremental strategy. For example, semantic lexica are at present being developed. This implies the extension of the SylLex formalism and the lexicon editor (see section 5). In addition, a named entities guesser is

[1] OAL stands for French “Outil d’aide au linguiste”, this is, a tool to assist linguists

under construction, which involves the API enrichment of the linguistic resources server (see section 3).

In this paper we give an overview of the whole OAL architecture and describe in detail its use for morphosyntactic lexica management (formalism and tools). However, this is only a part of the system. The paper is organized as follows. First, some related works are presented. Then, the OAL architecture is described. The fourth section briefly presents a morphosyntactic guesser, an example of an OAL component. The fifth section describes in detail the morphosyntactic lexicon editor, another example of an OAL component. Finally, we draw some conclusions and further work is presented.

2. Related Work

There are works related to each OAL component. For example, *Lexicon Creator* (Fontenelle et al. 2008), *TshwaneLex* (Joffe and Schryver 2004) and Word Manager (Hacken 2002, Hacken et al. 1994) are lexicon editors with powerful functionalities. Nevertheless, they are not integrated in a global processing architecture. An example of such architectures, we can mention the Victoria project (Nicolas et al. 2009) and Nooj (Silberztein 2005).The main difference of OAL is its client-server approach empowered by a rich lexicon editor, and the integration of a whole process that may start with the focused crawling of textual content from the Web and end by the automatic suggestion of term candidates to be validated by the linguist.

3. OAL Architecture

The OAL architecture (cf. figure 1) draws its inspiration from (Loupy and Gonçalves 2008).

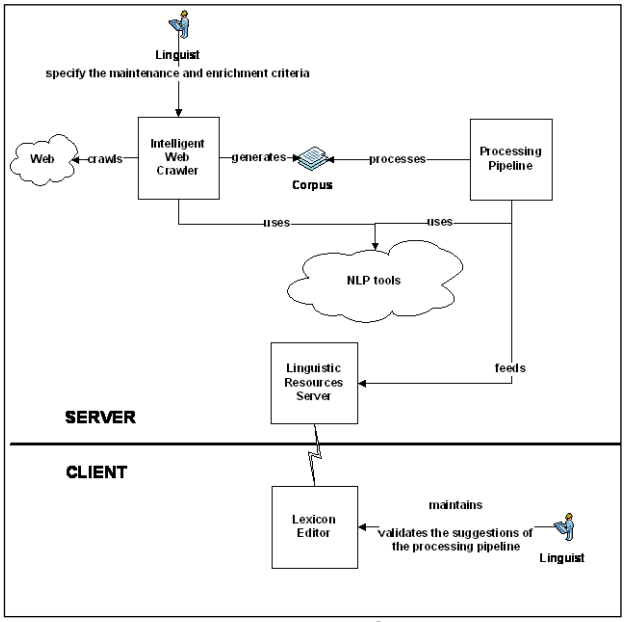

Figure 1: OAL Architecture.

The main components of this client-server architecture are:

- an intelligent web crawler,
- a processing pipeline,
- a linguistic resources server,
- specialized editors for creating, maintaining and enriching linguistic resources.

The processing pipeline role is to extract from a raw corpus the elements that are potentially relevant for an existing resource from a raw corpus. When developing a morphosyntactic lexicon, for example, the pipeline cleans the input (deletion of boilerplate texts) and extracts unknown words (not in the lexicon). This is not a trivial task, as a large amount of noise is generated during this automatic procedure. This is why a guesser tool has been developed (cf. section 4). The corpora are collected in three different ways: 1) manually built, 2) crawling, or 3) RSS feed. The processing pipeline relies on several tools developed internally at Syllabs or distributed as freeware: tokenizers, guessers (see next section), POS taggers, KWIC, etc.

The linguists do not have direct access to results obtained from the focused crawler and the processing pipeline. These components transfer the results, via web services, to the linguistic resources server. The server offers API to integrate the results and to access the resources. Let us consider the development of a domain-specific lexicon in the field of economics, first economic news are crawled to compile a corpus in this domain. Then, the pipeline cleans the texts, extracts new terms and transfers the results to the server, which integrates them as suggestions to the server version of the economy lexicon. Next, the server distributes the suggestions between the lexicon versions of the users who are in charge of maintaining the resource. Finally, when an authorized user checks out the economy lexicon, he can manually validate the suggestions. Managing the resources implies managing the server and local user versions, allowing the merging of different user versions, resolving conflicts when merging, but also other aspects such as user permissions, logs and statistical information concerning the word frequencies run on lexica and corpora.

A specialized lexicon editor provides an easy-to-use environment to create, maintain and enrich lexica. The editor is a smart client that interacts with the linguistic resources server (cf. section 5).

4. Using a Morphosyntactic Guesser

Lexicon enrichment means adding new words. When dealing with morphosyntactic lexica, the procedure implies assigning an inflection paradigm to a lemma or running a program to generate all the inflections of a lemma with their corresponding morphosyntactic tag.

Usually, the linguist is the one who has to organize the words to be added in different files. Each file corresponds to an inflection class to associate to each word or list of words. It is possible of course to create sublists with words classified automatically depending on the word ending of the word. Suffix information may indicate the inflection class of a word, but some noise can be

generated, as some suffixes may correspond to more than one word category In French, for instance, words ending with the suffix *–ment* can be an adverb or a noun, as the adverb *joliment* and the noun *jugement*. But still, suffixes are not enough information to associate automatically an inflection class to a word. This is why linguists do have to check the list manually.

A solution to speed up this process and make it more user-friendly is to integrate a guesser. Today, most PoS taggers include a guessing procedure to guess the PoS. However, a morphosyntactic lexicon needs more information. This is why we have developed a more advanced guesser to associate one or more triplet (lemma, inflection paradigm, confidence score) to an unknown word. The guesser uses a hybrid approach: a statistical model inspired from (Mikheev 1997) that computes probabilities from words endings and a set of heuristics. The heuristics includes the use of prefixes, the detection of foreign words and the verification over Internet of the forms guessed (Kuchmann-Beauger 2009, Kuchmann-Beauger et al. 2010).

As the guesser has been trained with the inflection paradigms of our lexicon, for each word it suggests a lemma and an inflection paradigm corresponding to the inflections paradigms of our lexicon. To illustrate, for the word *commercialization* the guesser proposes the following suggestions:

RG	Paradigm suggested	Probability
1	N_Fem-s	0.679254835
2	N_Masc-s	0.041687166
3	N_Masc-NULL	0.002511341
4	N_MascFem-NULL-s-s	0.002426684
5	A-NULL	1.5267522 E-4

Table 1: paradigms suggested by the guesser for the word *commercialization*.

5. Editing the lexicon

In this section we present the morphosyntactic lexicon editor implemented into the OAL framework. As we said before, similar works exist (Fontenelle et al. 2008, Joffe and Schryver 2004, Hacken 2002). Two main differences have to be stressed: (1) OAL editor is a rich client that communicates with a linguistics resources server and (2) the editor implements SylLex (Blancafort et al. 2010), a morphosyntactic lexicon formalism.

SylLex organizes every morphosyntactic lexicon in three components: a list of lemmas, a set of inflection paradigms and a set of patterns. Patterns help the linguist to create inflection paradigms: they contain the information about all morphosyntactic tags, all possible inflection forms of a lemma and further morphological information (e.g., stems and suffix). All this information is needed for word inflection and is specified by the linguist will when he/she creates an inflection paradigm. Each language has its own SylLex model, thus its own set of patterns and flexion paradigms.

When a new lexicon is created, the patterns are first outlined and then inflection paradigms are defined by filling the corresponding fields: stems and suffixes for Romance Languages. For German, for instance, a further field "prefix" is required for verbal inflection for the inflectional prefix *ge* as well as a specific attribute "particle" for separable verbs. SylLex formalism models the notion of word variants (as spelling and geographical variants) and compound words. See (Blancafort et al. 2010) for more details. For now, the morphosyntactic lexica are implemented as a set of text files but a migration to a database scheme is planned. Using a database will facilitate the integration of phonetic features and semantic lexica.

5.1. Lemma and paradigm view

The user can navigate the lexicon in two ways: lemma-based or inflection-paradigm-based navigation. As shown in figure 2, he/she can visualize all the lemmas in the left panel (a simple research option is proposed) and for each lemma, the system shows in the right panel all the inflection paradigms applied to that lemma. Visualization and maintenance are possible at the same time. Thus, as the user visualizes the inflection paradigm *VREG/e* applied to the lemma *amaze*, he/she can modify both lemma and inflection paradigm (upper buttons offer several functionalities).

As shown in figure 3, the paradigm view shows all the inflection paradigms on the left panel and, on the right panel, for each paradigm the resulting forms that are generated when applying this paradigm to a lemma prototype are visualized.

Both views are coordinated. As a result, if the user double clicks over a lemma in the lemma view, the system switches the view and shows the relevant paradigm for the lemma. The interaction done the other way round is more powerful: if the user double clicks over an inflection paradigm name, the editor switches to the lemma view and shows, by automatically filtering (see section 5.3), all the lemmas that have the paradigm applied.

5.2. Adding new words: Import

Adding new lemmas to the lexicon can be carried out by the *import* function. The linguist imports a text file with a list of lemmas corresponding to the same inflection paradigm, he/she selects the corresponding paradigm out of a list of paradigms that can be filtered by word category (noun, adjective, verb, adverb or specific function word). A pop-up window shows an alert if one or more of the words are already in the lexicon and shows the word and the paradigm assigned in the lexicon. If any conflict appears, the user can undo the import of the lemmas by removing them.

5.3. Finding words by criteria: Advanced filtering

As lexicons are usually big and continue to grow, searching and filtering are essential functions. The basic search functionalities are not enough powerful when the user wants to access all the forms that match several conditions (e.g. all the noun forms generated from a lemma that finishes by "ness" whose plural is generated from the inflection paradigm N-es). This is why the editor offers an advanced filtering tool. The user can define conditions over:

- the lemma type (simple, compound, both)
- the lemma and form string (regex or literal)
- the complete tags (regex allowed)
- the variants type
- the inflection paradigms: number of paradigms applied to a lemma, set of paradigms applied to a lemma (at least one, all, exactly all).

5.4 Lexicon formatting: Export

The aim of the editor is to enhance the maintenance and handling of the lexicon. To export the whole lexicon, there is a functionality to compile the lexicon in different formats (cf. table 2).

1) form-lemma-tag format				
improvement	improvement	Nc-s--		
improvements	improvement	Nc-p—		
2) form and lemma-tag couples				
abandon	abandon	Nc-s--	abandon	Vfb-----
abandons	abandon	Nc-p--	abandon	Vf-p3s--
3) lemma and inflection paradigms format				
[S] tunnel	N-s	V_REG/l_distilled		
[S] turban	N-s			
[S] Turk	N-s			

Table 2: Different export format examples.

The export format depends on the application that will use the lexicon. The exported lexicon does not necessarily contain all the forms. Indeed, the user can use the advanced filtering (see previous section) to decide which forms are to be exported. For example, table 2 shows three different formats: (1) form-lemma-tag format with a single couple lemma-tag per form, (2) form-lemma-tag format with one or more lemma-tag couple and (3) lemma type (**S**imple or **C**ompound)-form-tag-flexion paradigm. The [S] stands for simple lemma as opposed to a compound word.

Until now we have not needed a XML format, or to follow standards such as ISO TC37/SC4. However, in case of need, new output formats are easy to integrate.

5.4. Quality check

A further functionality checks the consistency of the paradigms. Thus, when the user clicks the ladybug button (cf. figure 5), the editor shows all the lemmas that have a consistency problem when a paradigm is applied. This is especially useful after a massive import or an automatic integration of guessed items.

Another useful functionality is the search on the internet of all the forms generated for a lemma. The user can access, for each form, to the hits count and to the matches (showed as a KWIC). This is particularly useful for neologisms, compound words and low frequent terms.

6. Conclusions and further work

In this paper we have presented the NLP architecture OAL, which helps the linguist to develop linguistic resources in an industrial context. We saw that OAL provides different functionalities to assist the linguist during the whole process of lexicon enrichment: a relevant corpus may be crawled from the Web, a linguistic pipeline processes corpora and includes a guessing tool to help the linguist to add new words by suggesting possible inflection paradigms for each new lemma for simple nouns, adjectives and verbs. This user-friendly environment improves the development of linguistic resources and assures its quality.

The resources definitions and tools are proprietary, as they have been designed and developed for internal use at Syllabs. The development of OAL has followed an incremental strategy. At present, semantic lexica, a named entities guesser and a named entities phonetizer are being developed. These new functionalities imply the enrichment of the SylLex formalism and the migration of the lexicons to an unified database scheme (for morphosyntactic and semantic lexicons). Further work will concentrate on how to integrate aligned multilingual lexica to the lexicon editor.

References

Blancafort H., Couto J., Seng S. (2010). "Morphosyntactic Lexica in the OAL Framework: Towards a Formalism to Handle Spelling Variants, Compounds and Multi-words." *Euralex 2010*, Leeuwarden, Netherlands.

Fontenelle T., Cipollone N., Daniels M., Johnson I. (2008). "Lexicon Creator: A Tool for Building Lexicons for Proofing Tools and Search Technologies." *Euralex 2008*, Barcelona, Spain.

Hacken P. ten, Bopp, S., Domenig, M., Holz, D.; Hsiung, A., Pedrazzini, S. (1994). "A Knowledge Acquisition and Management System for Morphological Dictionaries". In *Proceedings of COLING 94*. Kyoto, Japan.

Hacken, P. ten (2002). "Word Formation and the Validation of Lexical Resources". in González Rodríguez, M. And Paz Suárez Araujo, C. (eds.). In *Proceedings LREC 2002*. Las Palmas, Spain.

Joffe D., de Schryver G-M. (2004). "TshwaneLex. A State-of-the-Art Dictionary Compilation Program", *Euralex 2004*, Lorient, France.

Kuchmann-Beauger N. (2009). "Maintenance de lexiques à partir du Web par caractérisation de mots inconnus". Master Thesis. Université de Paris 13. Paris, France.

Kuchmann-Beauger N., Couto J., Blancafort H., de Loupy C. (2010). “Rich morphosyntactic guessing in the OAL framework”, to be submitted.

Loupy C. de, Gonçalves S. (2008). “Aide à la construction de lexiques morphosyntaxiques.” *Euralex 2008*, Barcelona, Spain.

Mikheev A. (1997). “Automatic rule induction for unknown-word guessing”. Computational Linguistics, 23(3):405-423.

Nicolas L., Molinero M., Sagot B., Sánchez E., de la Clergerie E., Alonso M., Farré J., Verges J.M. (2009). “Producción eficiente de recursos lingüísticos: el proyecto Victoria.” *SEPLN 2009*, Donostia, Spain.

Silberztein M. (2005) “Nooj’s Dictionaries”. *LTC 2005*.

Annexe 1: Lexicon editor figures

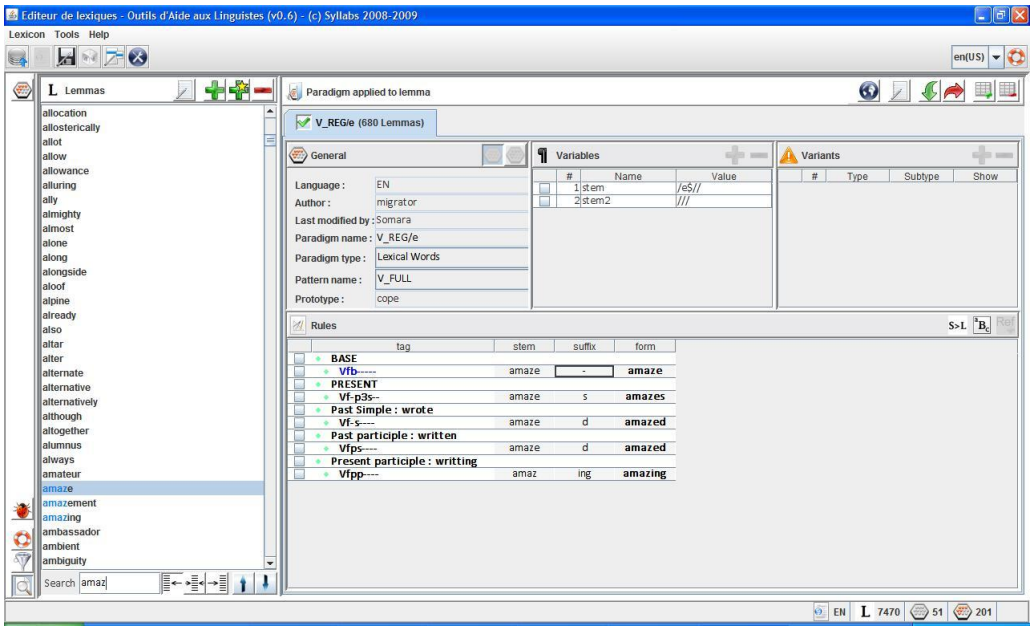

Figure 2: Lemma view.

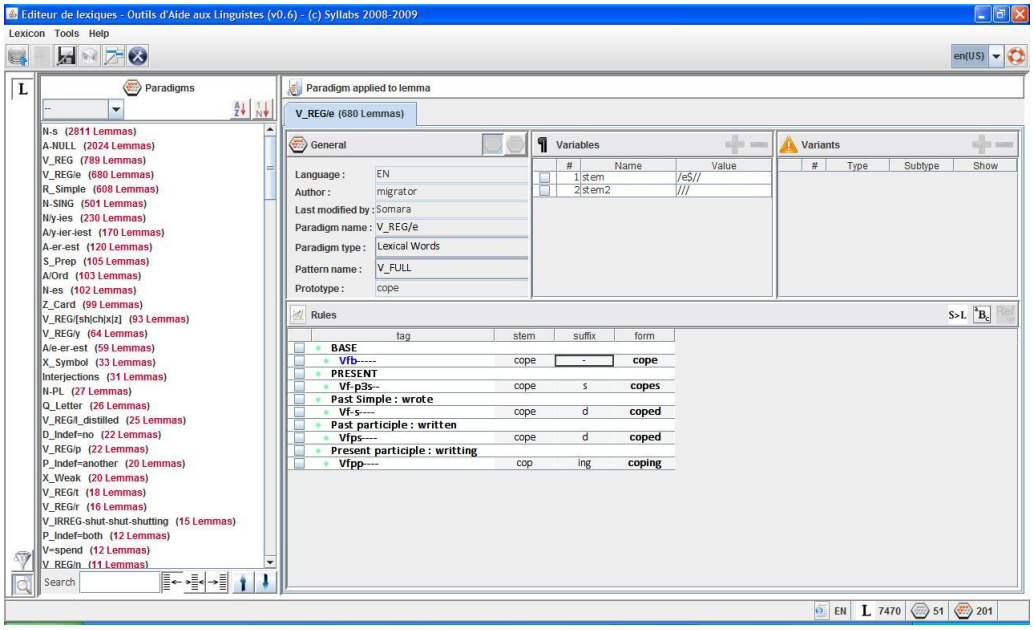

Figure 3: Paradigm view.

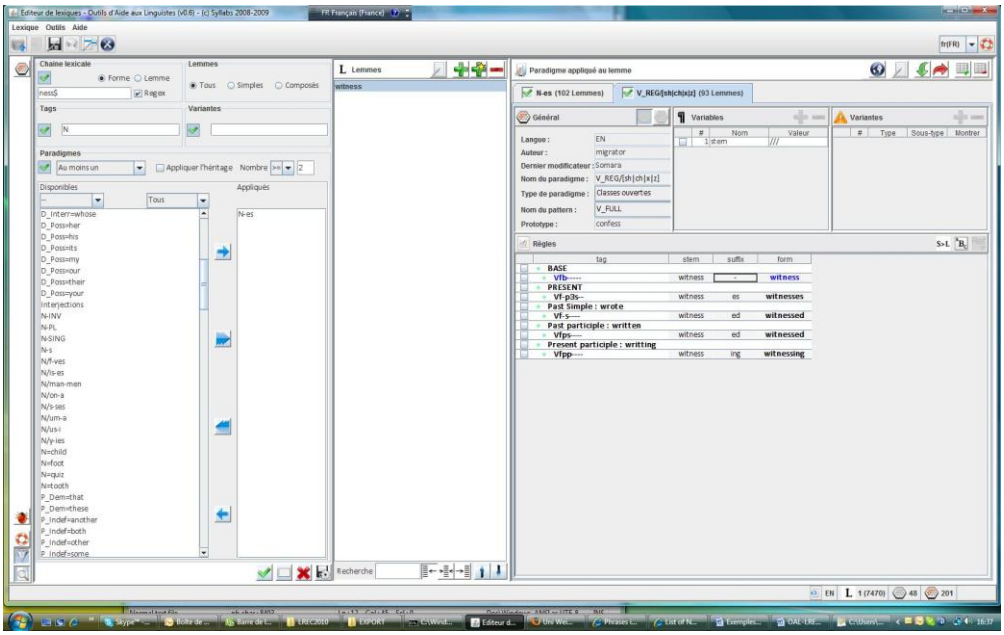

Figure 4: Lexicon filter.

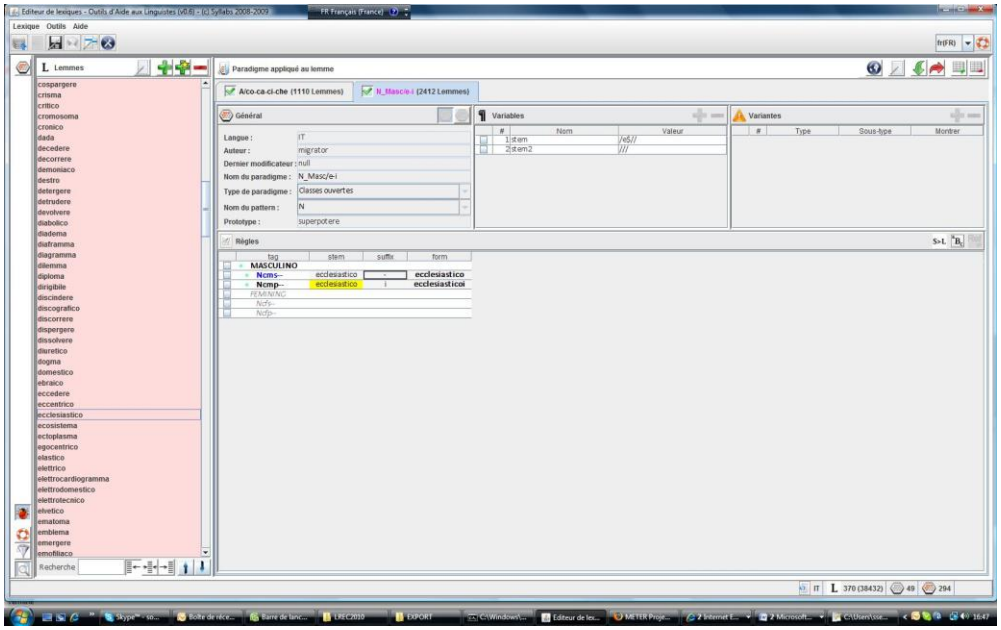

Figure 5: Consistency of the inflection paradigms.