NLP-based scripting for CALL activities

Georges Antoniadis, Sandra Echinard, Olivier Kraif, Thomas Lebarbé, Mathieu Loiseau, Claude Ponton

NLP-based scripting for CALL activities

Antoniadis G., Echinard S., Kraif O., Lebarbé T., Loiseau M., Ponton C.
LIDILEM, Stendhal University
Grenoble, France, F-38025
{Antoniadis; echinard; kraif; lebarbe; loiseau; ponton}@u-grenoble3.fr

Abstract

This article focuses on the development of Natural Language Processing (NLP) tools for Computer Assisted Language Learning (CALL). After identifying the inherent limitations of NLP-free tools, we describe the general framework of Mirto, an NLP-based authoring platform under construction in our laboratory, and organized into four distinct layers: *functions*, *scripts*, *activities* and *scenarios*. Through several examples, we explain how Mirto's architecture allows to implement state-of-the-art NLP *functions*, integrate them into easily handled *scripts* in order to create, without computing skills, didactic *activities* that could be recorded in more complex sequences or *scenarios*.

1 CALL: Conjugating NLP and language didactics

It is generally reckoned that computer science can prove itself to be a great aid in language learning, when in fact, most often computer scientists and didactics experts do not agree on the notion of "language". For the former, it corresponds to a sequence of codes, while for the latter it is a system of forms and concepts.

This divergence can easily be explained, when considering the fact that computer science, by definition, can only consider and process the form of the language independently of any interpretation, while, for language didactics, the form only exists through its properties and the concepts it is supposed to represent.

The consequences of these diverging approaches are "visible" in the great majority of language learning software. Many an imperfection of the latter's stem from the divergence mentioned above. Most language learning software are thought and implemented as computer products, only able to take into account a language form deprived of all semantics, or with extremely poor semantics. Caricaturely, rules as basic as that of the interpretation of the space remain ignored, which leads to unfortunate learning situations. For instance, if the learner answers "la casa" (sequence containing two spaces), his or her answer will not be accepted for the expected answer was "la casa" (sequence with one space). The pedagogical consequences of this poor "space processing" are obvious; the software teaches that the sequence of two spaces is not part of the language, and also, that all word preceded or followed by a space has nothing in common with the same word without the space! This down-to-earth example of the "spacebar syndrome" characterizes, in our opinion, the deficiencies of today's language learning software.

As (Chanier, 1998) and (Brun & al., 2002) point it out, and as (Antoniadis & Ponton, 2002) and (Antoniadis, 2004) have shown it, only the use of NLP methods and techniques allows to consider and process language as a system of forms and concepts. Considering them might lead to answers for two of the issues of existent CALL software.

The first concerns the rigidity of software: the data (instructions, examples, expected answers…) is to be predefined and, a few exceptions aside, can neither be modified nor enriched. Answer handling processes are intimately connected to this data. They are thus unable to consider new entries, unless they were explicitly anticipated.

The second problem concerns the inability of CALL software to adapt the course to the learners. Two types of courses are generally proposed. The first, the more classic, offers a predefined linear activity sequence. Whatever his (or her) answers and expectations, the learner will do (and do over) the same activities, using the same data. The second type of course offered is a "free" progression within a scenarized environment. It is the case of exploration software in which the learner is given a mission in a given environment (virtual reality). The dialogue, grammar or other activities are predefined, but will be performed in an order which will depend on the learner's mission completion process. This latter type of course, despite allowing a wider field of action for the learner (order of the mission, choice of activities…) does not offer real personalization or adaptation of the activities to the learner. Indeed, the course of action is independent of his or her answers for each stage, out of the incapacity of evaluating them. Last, we should bring to the

reader's attention that if the order in which the learner is confronted to the activities can vary according to his (or her) mission, the content of each activity remains invariable and will remain the same, whenever included in the course.

The last problem, which partly derives from the first two, characterizes current CALL software. As didactic products, this software should, *a priori*, be solely designed according to didactic solutions, expressed without constraints using pedagogical concepts. Now, current learning software are in fact computer products which require their users (language teachers, with little or no computing knowledge) to manipulate concepts and notions, which, *a priori*, do not belong to their language learning set of problems. Thus, instead of expressing pedagogic answers thanks to tools of their own discipline, they are forced to look for computerized solutions, which connect as much as possible with their own models or pedagogic aims. They might even have to give up on some pedagogical solutions, for they are unable to express them in a computer understandable way or because computer science is not able to handle them. To our knowledge, language didactics is presently able to imagine open pedagogic scenarios with exercises adapted according to each learner, examples changing when repeating the same activity within a given session, appropriate texts chosen to illustrate pedagogical contexts and, open and variable learning situations… Computer science is (and will be) unable to take into consideration these aspects with its own set of problems. Resorting to other knowledge (linguistics and language didactics) and to their modeling is essential. The use of NLP tools can constitute a way to resort to linguistic knowledge; the collaborative work of language didactics and NLP experts ought to provide answers concerning language didactics knowledge.

The problems that we have just presented explain, in our opinion, about the nature of language learning software so far. They were thought and implemented as computing problems and products which only use the aspects of language didactics that computer science is able to consider. The pedagogical solutions are often altered or truncated so that they can be computed. This approach), and also most of CALL software deficiencies, stem from computer science's narrow view of language (a simple sequence of codes.

Our approach towards the development of language learning software is radically different from those mentioned above. We consider that language learning software is above all a didactic product, a program which provides a didactic solution to a problem of language didactics, without altering, neither the solution nor, *a fortiori*, the problem. The design of such software requires that we should be able to adapt the possibilities of computer science to the implementation of pedagogical solution previously determined. In this approach, considering language properties, which are invariably present in every pedagogic solution concerning languages, is a must-have. Considering NLP methods, techniques and products only are capable of satisfying this condition, then a language learning software should be defined as the adaptation of NLP possibilities to the predefined didactic aims of language learning. In our opinion, such an approach is the only way to offer to language didactics experts not only tools that would not narrow the scope of treatment of their set of problems, but also tools with pedagogical added-value, capable of widening the set of problems of their discipline.

The use of NLP in the design of CALL software is not a new idea; systems like ELEONORE (Renié, 1995), ALEXIA (Chanier & Selva, 2000), or the EXILLS platform (Brun & al., 2002) resort to NLP methods and use NLP resources. Nevertheless, such examples remain marginal and concern non commercial products. Paradoxically, CALL and NLP, two fields centered on language, still seem to be ignoring each other. Most of the time, not using NLP is justified through the added cost resulting from its use. But more than the often-invoked extra cost, it is the lack of NLP culture, which should be held responsible for its absence.

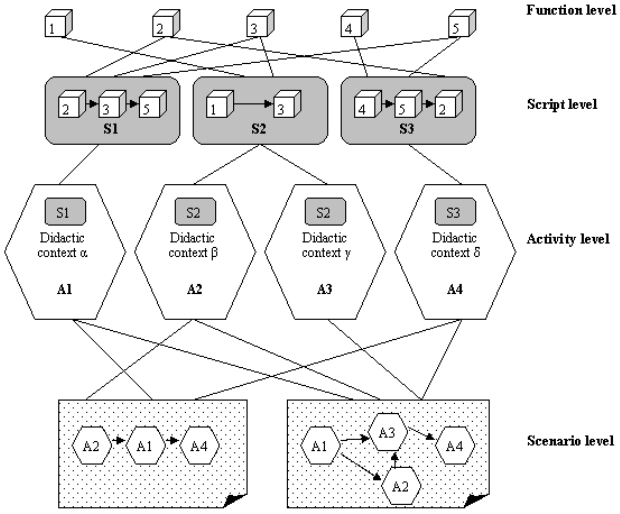
In the line of the systems mentioned above, the Mirto platform (Antoniadis & Ponton, 2004) (Forestier, 2002) is aiming at providing a global answer to the problems of CALL software, through an NLP approach on the one hand and on the other hand a collaborative work with didactics experts. More than a finished product, Mirto seeks to be a tool for the creation of didactic solutions for language learning. We present in the rest of the paper the aspects of the system, which describe our approach and its implementation.

2 Mirto description

The Mirto project is determinedly pluridisciplinary, and aims at giving an NLP toolbox to language teachers in order to design scenarios in their own pedagogical set of problems. The main goal of Mirto is to propose to the language teacher the possibility of designing pedagogical scenarios while fully taking advantage of NLP technologies in a user-friendly manner. Thus, those scenarios will be open (dynamical text database), will allow an individualized adaptation

according to the learner (automated generation of exercises, qualitative evaluation of the answers…) and should allow new possibilities (work on long texts, automated production of aids or exercises, design of non-linear scenarios …). The approach of Mirto is determinedly user-oriented since it is meant for language teachers who, *a priori* have little or no skill in computing nor in NLP. The technical nature of NLP has to be transparent to the language teacher and only the didactic aspects are to be visible and available to him.

In that way, four hierarchical levels (function, script, activity and scenario), associated with the text database, structure Mirto as it is illustrated on fig.1.



-Fig.1 – The MIRTO levels-

2.1 Function level

The functions (1 to 5 in fig.1) represent the Mirto lower level objects. They correspond to a basic NLP process such as tokenization (text splitting in forms) or language identification. Considering its technical nature and its independence from a didactic application, this level is not visible for any final users of Mirto (i.e. teachers and learners).

2.2 Script level

This level corresponds to the application of NLP functions to language didactics. A script (S1 to S3 in fig. 1) is a series of functions with a didactic purpose. So, this level needs both NLP and didactical competences and its design will be the result of an interdisciplinary work. For instance, the automated design of a gap-filling exercise is considered as a script because it connects the functions of language identification, tokenization, morphological analysis and gap creation depending on parameters chosen by the user.

2.3 Activity level

This level with the next one (scenario level) is the didactic core of Mirto. An activity (A1 to A4 in fig. 1) corresponds to the didactic contextualization of a script (previous level). Its goal is to associate a script with a text from the corpus database, some instructions, possible aids and an optional evaluation system. In order to create a gap-filling exercise, one only has to choose to apply the script of the previous example to a text while specifying the gaps criteria (for instance, hiding the preterit verbs and replacing them by their infinitive form), associating an instruction as “Fill in the blank with the preterit form” and specifying the evaluation form of the activity.

2.4 Scenario level

This level allows the teachers to define the sequence of activities in order to answer to their pedagogical objectives throughout the learner progression. This expected progression is not the same for each learner. Effectively, each of them will have a personal learning process linked to different factors. Mirto is dealing with that reality while proposing non-linear scenario creation. The path through the scenario depends on the individual process of each learner (learning course, evaluation…). That course is stored in a learners’ tracing database. For instance, according to his progress in a given scenario, a learner can be redirected to remediation activities, or retry an activity on another text or simply advance in the scenario.

2.5 Levels and users

There are three kinds of users in Mirto: NLP specialists, language specialists (didactic experts, linguists and teachers) and students. The following table shows the intervention level of each user of Mirto.

Level	**Use**	**User**
Function	Conception	NLP specialist
Script	Conception	NLP specialist + Language specialist
Activity	Conception	Language teacher
Scenario	Conception	Language teacher
	Playing	Student

Tab.1 – The intervention level of each user

This article deals more precisely with the NLP/CALL meeting, which takes place in the « script » level. However, before exposing the set

of problems of script designing, it is necessary to stress on the activity level, which uses that script level first.

3 Activity design

An activity is the implementation of a precise minimal pedagogical aim (for instance, having a work on a grammatical notion, revising conjugations, writing a paragraph, etc.). Activities are designed by language teachers through a specific interface: the activity editor. The activity editor (cf. Fig.2) is an authoring system. It allows to manipulate and format pedagogical objects such as texts (or text corpora), scripts and instructions.

In order to illustrate the steps of activities design, let us give the example of a teacher who wants to create an activity for the systematic revision of the preterit, using a gap filling exercise.

The design work is then broken up into five steps (cf. Fig.2). The first consists in selecting a script in the toolbox, which allows him to generate a gap filling exercise. The second is the definition of a didactic context for the script application. This script setting operation allows the teacher to select elements from a text base and determine the elements (criteria on the form, the category or/and morpho-syntactical features). These first two steps produce the desired gap-filling exercise, which will be integrated into the activity. Before the effective production of the activity, three steps remain: writing the instructions, precising the aids, which will be given to the learner, and finally specifying the evaluation criteria.

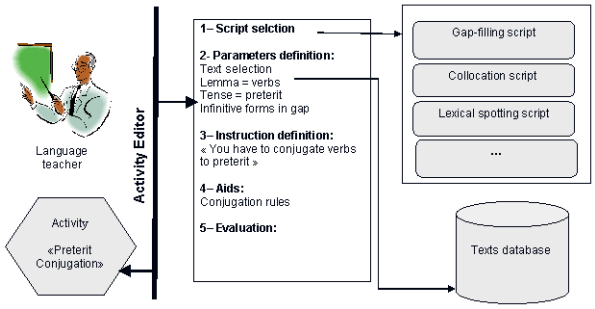


Fig. 2 – Example of activity design

4 CALL/NLP scripts

The script level represents the computing side of the didactic tools available in the Mirto environment. Scripts are integrated modules that implement one or several NLP standard resources and processes such as tagging, stemming, lemmatizing, parsing, dictionaries, etc. The standardization of these functions is an important aspect, because Mirto does not aim at developing new NLP techniques, but only at giving a framework to take advantage of the existing state of the art: Mirto is a car running with a NLP engine, and the engine may be changed, as a simple interchangeable part, if a new engine allows to get better performance.

Thus, scripts are the core of Mirto's architecture: their design should allow to transform the engine kinetic energy into movement and direction on the road of didactic activities, without requiring that the driver to have mechanic skills.

4.1 Parameters

As any computing module, a script will be directed by a set of parameters. These parameters shall not be accessible to the end-user *directly*, but through a *control panel*. This control panel shall be relevant from the didactic point of view; that is why the controls may be transcribed into a set of parameters. Let us take the example of the gap-filling exercise generator. By the mean of a simple form, the user may define:

a) which the units are to be removed from the text. Any linguistic feature should be used for this definition: lemma (e.g. *to drive*), part-of-speech (ex. *verb*), morphosyntactic description (ex. *past tense*), or even meaning (e.g. "car" semantic field - this functionality has not been implemented yet).

b) what information has to be given in the gap : *none*, the *lemma*, the *morphosyntactic features*, a *synonym*, a *definition* (not implemented yet) etc.

c) if the removed words should appear or not as an ordered list in the text header.

d) if the learner's answer should initiate a feed-back process immediately after it was entered.

On the user interface, the controls have to be:

- simple: two many features could discourage the user
- declarative: the user is not supposed to handle a tough formal language, so the control definition has to be intuitive and immediately understandable.
- user-friendly: the interface must allow to pick out the important information. For instance, a first form may present the standard settings for a control, and a second optional form may give access to advanced settings of the generator.

It is clear that the definition of linguistic features in a) involves a simple transcription process in order to determine the script parameters: the tagged and lemmatized texts handled by the generator use specific codes for morphosyntactic description. Declarative features as "Verbo, Prima coniugazione, Indicativo, Presente, Prima persona, Singolare" will be transcribed into a parameter set: "base=er$", ctag="verb", msd="IndP SG P1".

Even if this transcription process appears to be unavoidable, the script design must render the

Selection criterion	Script type	Example of activity	Expected answer	Involved NLP functions
Semantic	lexical spotting	Spot every word related to the "*car*" topic	Spotting of "*drive*", "*taxi*", "*engine*", "*road*", etc.	morphosyntactic tagging, lemmatization, semantic net interrogation
Semantic	lexical question	Give an Italian translation for "*to drive*"	Entering of "*guidare*"	morphosyntactic tagging, lemmatization, bilingual dictionary interrogation
Morpho-syntactic	gap-filling	Replace every infinitive verb in the gaps, using the appropriate tense	Replacement of "t*o wait*" by "*have been waiting*"...	morphosyntactic tagging, lemmatization
Morpho-syntactic	lexical question	What would be the contrary of the adverb "*lentement*"?	Entering of "rapidement"	morphosyntactic tagging, lemmatization, semantic net interrogation
Morphological	lexical spotting	Spot every word derived from the verb "*traduire*"	Spotting of "*traducteur*", "*traduction*", "*retraduite*", etc.	morphosyntactic tagging, lemmatization, stemming
Morphological	gap-filling	Fill every gap by a word of the "*traduire*" verb family	Entering of "*traducteur*", "*traduction*", "*retraduite*", etc.	morphosyntactic tagging, lemmatization, stemming

Tab.2 - Example of scripting for activity generation

parameters as close as possible to the user's control.

4.2 Incremental approach

It is impossible to determine from scratch what the exact form of a script must be. There are two reasons for this uncertainty:

- NLP functions are multifaceted, they may require complex sets of parameters to give an expected result, and the form of their input and output may have many different forms.
- the application field of NLP for a didactic use has been so far unexplored. New activities, new pedagogical habits, and new teachings are likely to emerge from these new technologies.

We strongly claim that only the pedagogical practice can pave the way.

Thus, designing the script, one may offer complex functionalities without real interest. Other scripts may appear to be very useful in some applications for which they were not initially designed. What we propose is to combine both top-down and bottom-up approaches: the proposed tools may offer wide possibilities, among which the pedagogical practice may select a few interesting features. Conversely, the practice may give rise to new needs that the technology will try to meet.

As suggested by (Kraif, 2003), to initiate the incremental process of script designing, we have chosen existing activities that may take advantage of simple improvements from NLP techniques. For these activities, we have tried to define scripts with a major modularity, i.e. scripts that may be reusable in different contexts and for a large spectrum of didactic applications. At last, another important criterion was given by the performances and limitations of the implemented functions: when a NLP task yields a 20% error rate, the results may not be valid for every kind of activity: erroneous information may be very confusing for a learner.

4.3 Examples of scripting

Most of the following examples are not implemented in the Mirto platform yet: but they are all realistic, given the current NLP state of the art, and may be added to Mirto in the short term.

The scripts fall into three categories

4.3.1 Activity generators

Given an input text, NLP techniques allow to select lexical units and expressions that bear specific lexical, idiomatic, grammatical or semantic features. This ability makes it possible to create a wide range of activities using generators for *gap-filling*, *lexical spotting* (i.e. identification of specific units of the text) or *lexical questions* (i.e. questions about units occurring in the text, concerning synonyms, contraries, translation equivalents, etc.). Table 2 shows various examples of generated activities.

Other scripts can be used upstream for the input text constitution: for instance a concordancer allows to extract from a corpus every unit (and the surrounding context) that satisfies the former selection criteria.

Such a concordance script, integrated with an appropriate interface, may give rise to a full activity, in order to allow the learner to search by him/herself examples (in context) that may help him/her solve a problem. A bilingual concordance script, involving an NLP aligning function, may also be very useful for this kind of text mining.

Similar activity generators may work without any input text, applying the selection criteria on a dictionary, and using, if necessary, a random draw to select a single unit:

- Conjugator: e.g. "Conjugate the expression *tomber en panne sèche* to : subjonctif imparfait, première personne du singulier"
- Lexical question: e. g. "Give a synonym for the word *phare*."
- Morphological question: e.g. "Give a noun derived from the verb *conduire*".

Another interesting application of NLP technique for activity generation is to implement a kind of "chat-bot", following the classical model of Elisa, able to simulate a conversation with a virtual interlocutor on a given subject.

4.3.2 Comprehension aid

For any kind of activity (reading a text, doing an exercise, etc.), it is possible to propose interactive aids for the learner. Most of the NLP tools available on the Exills platform belong to this category: at any time the learner can ask questions to a robot, that gives access to dictionary definitions (after a context sensitive disambiguation) and to a conjugator, or allows to find the correct form of a wrongly spelt word.

Such aids can be either *generic* (like dictionary or concordance consultation, grammar lessons, conjugator, phonetiser) or *context dependent* (a click on a word can give access to its morphological tags, lemma, syntactic function, definition and/or translation). As an example, we have implemented a contextual aid that automatically links to specific grammar lessons according to the morphosyntactic features of the clicked word: when an Italian verb is at the "*passato remoto*" tense, a hyperlink is automatically pasted in the contextual popup, giving access to the corresponding grammar lesson (see fig. 3).

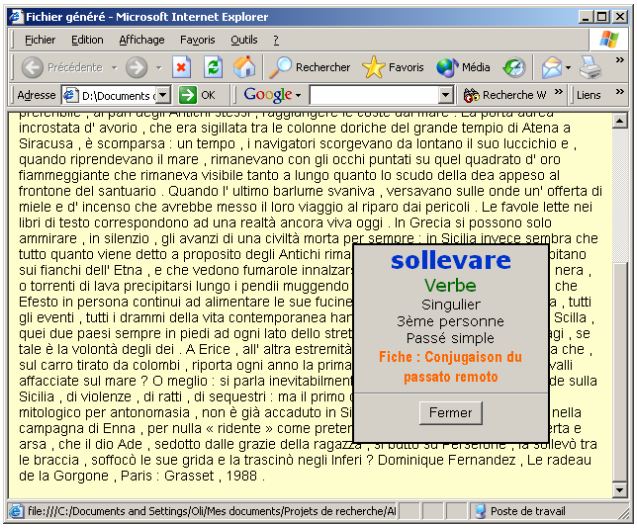


Fig.3 - Example of generated contextual aid

For the teacher, the handling of these scripts corresponds to specific settings of the final activity interface.

4.3.3 Automated evaluation

The learner production, in the framework of an activity, may have very various forms: clicks on check box, words, sentences or even texts.

The evaluation of sentences and texts is a tough problem: NLP techniques cannot really give reliable information about features that require a human interpretation (meaning, style, etc.). Even for the simplest task of error detection, the existing models are both silent and noisy at the same time: some errors are not detected, and correct expressions are wrongly pointed out as errors.

On the opposite, the evaluation of a multiple choice questionnaire is a trivial problem that does not need the expensive implementation of NLP tools.

For now, we think that the most realistic and promising application concerns the evaluation of simple lexical productions. We are currently studying a three levels protocol for the evaluation of a given answer with respect to the expected correct answer. If the given answer is different, three cases are considered:

1- Spelling error: if the entered chain does not exist in an inflected form dictionary, one can assume that it bears a spelling error. If the chain is very close to the correct answer, a message can be displayed, warning about the spelling error. Else, a list of resembling existing words can be proposed to the learner, asking him to make a choice.

2- Morphosyntactic level: at this stage, the answer is integrated in the linguistic context of the activity (for instance, the sentence where the gap was done, in a gap-filling exercise), in order to compute a morphosyntactic analysis with tagging

and lemmatization. If the lemma is the same than the lemma of the correct answer, a warning can be displayed about the difference in the morphosyntactic features (e.g. "wrong tense", "wrong number", etc.).

3- Semantic level: in the case of a different lemma, a semantic wordnet is searched in order to check whether a close semantic link (synonymy, hyperonymy, hyponymy, meronymy, antonymy) can be found between the given answer and the expected one. Then, a warning can be displayed such as "be more precise", "not exactly", etc.

In the global architecture, such a script could be useful for the evaluation of various activities: gap-filling, lexical questions, etc. According to the specific context and aim of a given activity, the feed-back to the learner may be very different. For instance, if a gap-filling exercise is designed to test the ability to conjugate verbs in a given tense, the fact that the lemma of the learner's answer is different is not very important, provided that the verbal flexion is correct.

Therefore, in the design of such an evaluation script, it is important to separate the comparison and the feed-back. We plan to implement too scripts:

- the comparison script that takes as an input: the linguistic context, the expected answer, the given answer; and returns a *difference code* such as:

0: no difference

1.1: spelling error on the expected answer

1.2: spelling error on another answer (with a list of close words)

2.1: different lemma

2.2: different morphosyntactic features

2.2.1, 2.2.2,...: different number, different gender, etc.

3.1,3.2,...: synonym, hyperonym, etc.

- the feed-back script that takes as an input the difference code, and returns a message, such as : "yes, but the spelling is wrong", "be more precise". Even if one can propose standard messages for each difference code, the teacher should obviously be able to edit an adapted message set depending on the didactic context of a given activity.

5 Current functionalities of Mirto and perspectives

The development of Mirto started about a year ago. A total of three years should be necessary to complete the first prototype. A handling period is to be foreseen in order to allow teachers to master the use of the product. We plan to integrate Mirto to the Stendhal Intranet for experimentation.

So far, the development of Mirto mainly concerned the script creation module. Completing this module allowed the integration of various NLP (and non-NLP) software. Other software, especially NLP-based, ought to be integrated. The choice of the number and nature of integrated software can only be done through a process of exchange involving both language teachers and NLP experts. We consider that, the software integrated so far allowed the creation of enough scripts for an experimental use of Mirto.

In order to perform tests and validate the global approach, a first version of the activity and scenario editor has been implemented. It allows the creation of almost every type of activity (excluding the evaluation) and the design of linear scenarios that will not trace the learner training history.

The definition of the approach underneath Mirto, along with making use of it, originated various research works, which are currently being carried out. Apart from the implementation of the prototype of the system, our efforts particularly concern the following aspects:

- the pedagogical annotation and indexation of texts towards the creation of a corpus to be used by language teachers (Loiseau, 2003)
- the automatic analysis and pedagogical analysis of the learners' answers using NLP based tools and techniques.
- scripting and interfacing for activity generation

The first results of these works should find their application in Mirto.

At the crossroads of three branches – language didactics, NLP and computer science – Mirto raises new problems, not only in each of these branches (advances in NLP for instance) but also problems for which no solution can be reached unless the branches (and their specialists) work in quasi osmosis. One can mention among others the examples of the automatic definition of the appropriate response for the learners' answers, the modeling and implementation of computer functions manipulating language didactics concepts (so as to provide language teachers with no specific computational skills with tools they can handle), the definition and pedagogical exploitation of the trace of the learners' activity or the modeling of non-linear scenarios… Unless we find answers to these problems, CALL will have to settle for the creation of pedagogical-added-value-less-products.

References

G. Antoniadis and C. Ponton. 2004. *Mirto : un système au service de l'enseignement des langues*, UNTELE'2004, Compiègne, France.

G. Antoniadis and C. Ponton. 2002. *Le TAL : une nouvelle voie pour l'apprentissage des langues*. UNTELE'2002, Compiègne, France.

C. Brun, T. Parmentier, A. Sandor and F. Segond., "Les outils de TAL au service de la e-formation en langues", *Multilinguisme et traitement de l'information*. F. Segond, ed., pages 223-250, Hermès Science Publications, Paris, France

T. Chanier. 1998. *Relations entre le TAL et l'ALAO ou l'ALAO un "simple" domaine d'application du TAL ?*. International conference on natural language processing and industrial application (NLP+IA'98), Moncton, Canada.

T. Chanier and T. Selva. 2000. "Génération automatique d'activités lexicales dans le système ALEXIA". *Sciences et Techniques Educatives* 7(2):385-412, Hermès, Paris, France

G. Forestier. 2002. *Plates-Formes pour l'enseignement des langues : le cas de Mirto*. Mémoire pour l'examen probatoire en Informatique, CNAM, Grenoble, France.

O. Kraif. 2003. "Propositions pour l'intégration d'outils TAL aux dispositifs informatisés d'apprentissage des langues, Intercompréhension en langues romanes". *LIDIL*, n° 28, C. Degache ed., pages 153-165, Grenoble, France.

M. Loiseau. 2003. *Vers la création d'une base de données de ressources textuelles indexée pédagogiquement pour l'enseignement des langues*. Mémoire de DEA en Sciences du Langage, Stendhal University, Grenoble, France.

D. Renié. 1995. *Modélisation informatique de l'acquisition des interrogatives directes en français langue seconde dans leur dimension pragmatique, proposition d'un environnement offrant un apprentissage collaboratif : ELEONORE*. PhD Thesis, Clermont II University, Clermont-Ferrand, France