AutoLearn's authoring tool: a piece of cake for teachers

Martí Quixal[1], Susanne Preuß[3], David García-Narbona[2], Jose R. Boullosa[2]

[1] Voice and Language Group,
[2] Advanced Development Group
Barcelona Media Centre d'Innovació
Diagonal, 177, E-08018 Barcelona, Spain
{marti.quixal,david.garcian,
beto.boullosa}@barcelonamedia.org

[3] GFAI
Martin-Luther-Str. 14
Saarbrücken, Germany
susannep@iai.uni-sb.de

Abstract

This paper[1] presents AutoLearn's authoring tool: AutoTutor, a software solution that enables teachers (content creators) to develop language learning activities including automatic feedback generation without the need of being a programmer. The software has been designed and implemented on the basis of processing pipelines developed in previous work. A group of teachers has been trained to use the technology and the accompanying methodology, and has used materials created by them in their courses in real instruction settings, which served as an initial evaluation.

The paper is structured in four sections: Section 1 introduces and contextualizes the research work. Section 2 describes the solution, its architecture and its components, and specifically the way the NLP resources are created automatically with teacher input. Section 3 describes and analyses a case study using the tool to create and test a language learning activity. Finally Section 4 concludes with remarks on the work done and connections to related work, and with future work.

1 Introduction

Over the past four decades there have been several hundreds of CALL (Computer-Aided Language Learning) projects, often linked to CALL practice (Levy 1997), and within the last twenty years a considerable number of them focused on the use of NLP in the context of CALL (Amaral and Meurers, in preparation). Despite this, there is an appalling absence of parser-based CALL in real instruction settings, which has been partially attributed to a certain negligence of the pedagogical needs (Amaral and Meurers, in preparation). In contrast, projects and systems that were pedagogically informed succeeded, yielded and are yielding interesting results, and are evolving for over a decade now (Nagata 2002; Nagata 2009; Heift 2001; Heift 2003; Heift 2005; Amaral and Meurers, in preparation). According to Amaral and Meurers successful projects were able to restrict learner production in terms of NLP complexity by limiting the scope of the learning activities to language-oriented (as opposed to communicative-oriented) or translation exercises, or by providing feedback on formal aspects of language in content oriented activities, always under pedagogical considerations –focus on form.

Our proposal is a step forward in this direction in two ways: a) it allows for feedback generation focusing both on formal and content (communicative-oriented) aspects of language learning activities, and b) it provides teachers with a tool and a methodology –both evolving– for them to gain autonomy in the creation of parser-based CALL activities –which by the way has a long tradition in CALL (Levy 1997, chap. 2). The goal is to shape language technologies to the needs of the teachers, and truly ready-to-hand.

1.1 Related work and research context

The extent to which pedagogues appreciate and require autonomy in the design and creation of CALL activities can be traced in the historical

[1] Research funded by the Lifelong Learning Programme 2007-2013 (AUTOLEARN, 2007-3625/001-001).

overview offered by (Levy 1997, 16, 17, 19, 23 and 38). Moreover, parallel research shows that the integration of CALL in the learning context is

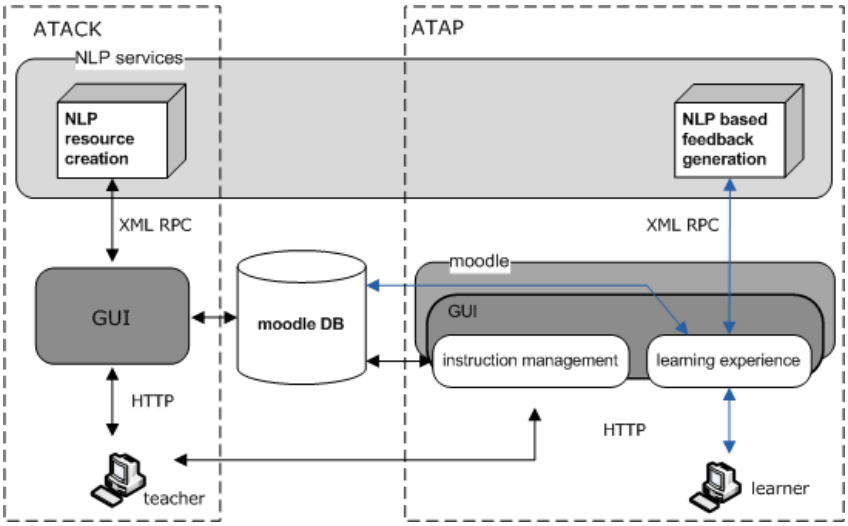

Figure 1. AutoTutor software architecture.

critical to ensure the success of whatever materials are offered to learners (Levy 1997, 200-203; Polisca 2006).

AutoTutor goes beyond tools such as Hot Potatoes, eXelearning or JClic[2] in that it offers the possibility of authoring NLP-based CALL activities. It is also more ambitious than other authoring tools developed for the creation of activities in intelligent tutoring systems. Chen and Tokuda (2003) and Rösener (2009) present authoring tools for translation exercises, where expected learner input is much more controlled (by the sentence in the source language).

Heift and Toole (2002) present Tutor Assistant, which enables to create activities such as build-a-sentence, drag-and-drop and fill-in-the-blank. An important difference between AutoTutor and Tutor Assistant is that the latter is a bit more restrictive in terms of the linguistic objects that can be used. It also presents a lighter complexity in the modelling of the underlying correction modules. However, the system underlying Tutor Assistant provides with more complex student adaptation functionalities (Heift 2003) and would be complementary in terms of overall system functionalities.

2 AutoTutor: AutoLearn's authoring software

AutoTutor is a web-based software solution to assist non-NLP experts in the creation of language learning activities using NLP-intensive processing techniques. The process includes a simplified specification of the means to automatically create the resources used to analyse learner input for each exercise. The goal is to use computational devices to analyse learner production and to be able to go beyond "yes-or-no" answers providing additional feedback focused both on form and content.

This research work is framed within the AutoLearn project, a follow up of the ALLES project (Schmidt et al., 2004, Quixal et al., 2006). AutoLearn's aim was to exploit in a larger scale a subset of the technologies developed in ALLES in real instruction settings. Estrada et al. (2009) describe how, in AutoLearn's first evaluation phase, the topics of the activities were not attractive enough for learners and how learner activity decreased within the same learning unit across exercises. Both observations –together with what it has been shown with respect to the integration of independent language learning, see above – impelled us to develop AutoTutor, which allows teachers to create their own learning units.

As reflected in Figure 1, AutoTutor consists primarily of two pieces of software: AutoTutor Activity Creation Kit (ATACK) and AutoTutor Activity Player (ATAP). ATACK, an authoring

[2] http://hotpot.uvic.ca/, http://sourceforge.net/apps/trac/exe/wiki, http://clic.xtec.cat/es/jclic/index.htm.

tool, provides teachers with the ability to create parser-based CALL exercises and define the corresponding exercise specifications for the generation of automated feedback. ATAP allows teachers to insert, track and manage those exercises in Moodle (http://moodle.org), giving learners the possibility to visualize and answer them. Both ATACK and ATAP share a common infrastructure of NLP services which provides the basic methods for generating, storing and using NLP tools. Access to those methods is made through XML-RPC calls.

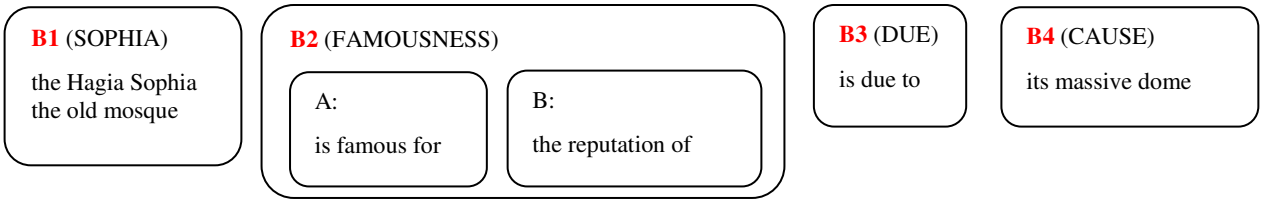

Figure 2 Blocks as specified in AutoTutor GUI.

2.1 AutoTutor Activity Creation Kit

ATACK is divided in two components: a GUI that allows content creators to enter the text, questions and instructions to be presented to learners in order to elicit answers from them; and an NLP resource creation module that automatically generates the resources that will be used for the automated feedback. Through the GUI, teachers are also able to define a set of expected correct answers for each question, and, optionally, specific customized feedback and sample answers.

To encode linguistic and conceptual variation in the expected answers, teachers are required to turn them into linguistic patterns using blocks. Blocks represent abstract concepts, and contain the concrete chunks linked to those concepts. Within a block one can define alternative linguistic structures representing the same concept. By combining and ordering blocks, teachers can define the sequences of text that correspond to the expected correct answers –i.e., they can provide the seeds for answer modelling.

Modelling answers

Given an exercise where learners are required to answer the question "From an architecture point of view, what makes Hagia Sophia in Istanbul so famous according to its Wikipedia entry?", the following answers would be accepted:

1. {The Hagia Sophia/The old mosque} is famous for its massive dome.
2. The reputation of {the Hagia Sophia/the old mosque} is due to its massive dome.

To model these possible answers, one would use four blocks (see Figure 2) corresponding to WHO (*Hagia Sophia*), WHAT (*Famousness*), and WHY (*Dome*), and complementary linguistic expressions such as "is due to". Thus, the possible correct block sequences would be (indices corresponding to Figure 2):

a) B1 B2.A B4
b) B2.B B1 B3 B4

Block B1 is an example of interchangeable alternatives (*the Hagia Sophia* or *the old mosque*), which do not require any further condition to apply. In contrast, block B2 is an instance of a syntactic variation of the concept. *Famousness* can be expressed through an adjective or through a verb (in our example), but each of the choices requires a different sentence structure.

Alternative texts in a block with no variants (as in B1) exploit the paradigmatic properties of language, while alternative texts in a block with two variants as in B2 account for its syntagmatic properties, reflected in the block sequences. Interestingly, this sort of splitting of a sentence into blocks is information-driven and simplifies the linguistic expertise needed for the exercise specifications.

2.2 Automatic generation of exercise-specific NLP-resources

Figure 3 shows how the teacher's input is converted into NLP-components. Predefined system components present plain borders, and the resulting ones present hyphenised borders. The figure also reflects the need for answer and error modelling resources.

NLP resource generation process

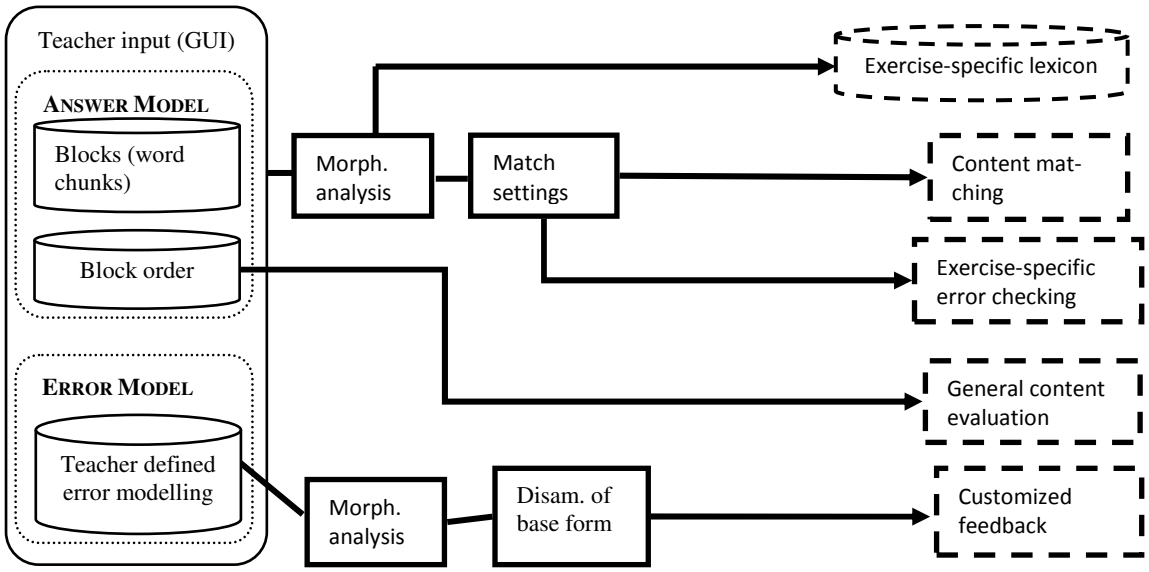

Figure 3. Processing schema and components of the customizable NLP resources of ATACK

The generation of the NLP resources is possible through the processing of the teacher's input with three modules: the morphological analysis module performs a lexicon lookup and determines unknown words that are entered into the exercise-specific lexicon; the disambiguation of base form module, disambiguates base forms, e.g. "better" is disambiguated between verb and adjective depending on the context in preparation of customized feedback.

The last and most important module in the architecture is the match settings component, which determines the linguistic features and structures to be used by the content matching and the exercise-specific error checking modules (see Figure 4). Using relaxation techniques, the parsing of learner input is flexible enough to recognize structures including incorrect word forms and incorrect, missing or additional items such as determiners, prepositions or digits, or even longish chunks of text with no correspondence the specified answers. The match settings component contains rules that later on trigger the input for the exercise-specific error checking.

The match settings component consists of KURD rules (Carl et al. 1998). Thus it can be modified and extended by a computational linguist any time without the need of a programmer.

Once the exercise's questions and expected answers have been defined, ATACK allows for the generation of the NLP resources needed for the automatic correction of that exercise. The right-hand side of Figure 3 shows which the generated resources are:

- An *exercise-specific lexicon* to handle unknown words
- A *content matching module* based on the KURD formalism to define several linguistically-motivated layers with different levels of relaxation (using word, lemma, and grammatical features) for determining the matching between the learner input and the expected answers
- A *customized feedback module* for teacher-defined exercise-specific feedback
- An *exercise-specific error checking module* for context-dependent errors linked to language aspects in the expected answers
- A *general content evaluation component* that checks whether the analysis performed by the content matching module conforms to the specified block orders

2.3 AutoTutor Activity Player (ATAP)

With ATAP learners have access to the contents enhanced with automatic tutoring previously created by teachers. ATAP consists of a) a client GUI for learners, integrated in Moodle, to answer exercises and track their own activity; b) a client GUI for teachers, also integrated in Moodle, used to manage and track learning resources and learner

activity; and c) a backend module, integrated into the AutoTutor NLP Services Infrastructure, responsible for parsing the learner's input and generating feedback messages.

Figure 4 describes the two steps involved in the NLP-based feedback generation: the NLP components created through ATACK –in hyphenised rectangles– are combined with general built-in NLP-based correction modules.

2.4 The feedback generation software

Feedback is provided to learners in two steps, which is reflected in Figure 4 by the two parts, the upper and lower part, called General Checking and Exercise Specific Checking respectively. The former consists in the application of standard spell and grammar checkers. The latter consists in the application of the NLP resources automatically generated with the teacher's input.

Content matching module

The text chunks (blocks) that the teacher has entered into ATACK's GUI are converted into KURD rules. KURD provides with sophisticated linguistically-oriented matching and action operators. These operators are used to model (predictable) learner text. The content matching module is designed to be able to parse learners input with different degrees of correctness combining both relaxation techniques and mal-rules. For instance, it detects the presence of both correct and incorrect word forms, but it also detects incorrect words belonging to a range of closed or open word classes –mainly prepositions, determiners, modal verbs and digits– which can be used to issue a corresponding linguistically motivated error messages like "Preposition wrong in this context", in a context where the preposition is determined by the relevant communicative situation.

Error types that are more complex to handle in technical terms involve mismatches between the amount of expected elements and the actual amount of informational elements in the learner's answer. Such mismatches arise on the grammatical level if a composite verb form is used instead of a simple one, or when items such as determiners or commas are missing or redundant. The system also accounts for additional modifiers and other words interspersed in the learner's answer.

The matching strategy uses underspecified empty slots to fit in textual material in between the correct linguistic structures. Missing words are handled by a layer of matching in which certain elements, mainly grammatical function words such as determiners or auxiliary verbs, are optional.

Incorrect word choice in open and closed word classes is handled by matching on more abstract linguistic features instead of lexeme features.

The interaction between KURD-based linguistically-driven triggers in the content matching module and the rules in the exercise-specific error checking (see below) module allows for specific mal-rule based error correction.

Customized feedback

Teachers can create specific error messages for simple linguistic patterns (containing errors or searching for missing items) ranging from one or two word structures to more complex word-based linguistic structures. Technically, error patterns are

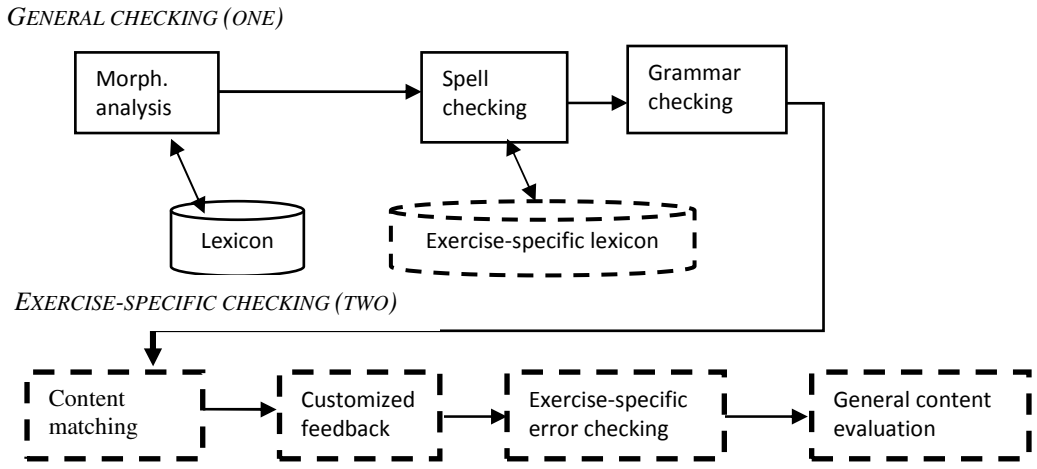

Figure 4. Processing schema of the NLP resources to generate automatic feedback.

implemented as KURD rules linked to a specific error message. These rules have preference over the rules applied by any other later module.

Exercise-specific error checking

Teachers do not encode all the exercise-specific errors themselves because a set of KURD rules for the detection of prototypical errors is encoded –this module uses the triggers set by the content matching component. Exercise-specific linguistic errors handled in this module have in common that they result in sentences that are likely to be wrong either from a formal (but context-dependent) point of view or from an informational point of view.

General content evaluation

Since the contents are specified by the blocks created by teachers, the evaluation has a final step in which the system checks whether the learner's answer contains all the necessary information that belongs to a valid block sequence.

This module checks for correct order in information blocks, for blending structures (mixtures of two possible correct structures), missing information and extra words (which do not always imply an error). The messages generated with this component pertain to the level of completeness and adequacy of the answer in terms of content.

3 Usage and evaluation

AutoTutor has been used by a group of seven content creators –university and school teachers– for a period of three months. They developed over 20 activities for learning units on topics such as business and finance, sustainable production and consumption, and new technologies. Those activities contain listening and reading comprehension activities, short-text writing activities, enabling tasks on composition writing aspects, etc. whose answers must be expressed in relatively free answers consisting of one sentence. In November 2009, these activities were used in real instruction settings with approximately 600 learners of English and German. Furthermore, an evaluation of both teacher and learner satisfaction and system performance was carried out.

We briefly describe the process of creating the materials by one of the (secondary school) teachers participating in the content creation process and evaluate the results of system performance in one activity created by this same teacher.

3.1 Content creation: training and practice

To start the process teachers received a 4-hour training course (in two sessions) where they were taught how to plan, pedagogically speaking, a learning sequence including activities to be corrected using automatically generated feedback. We required them to develop autonomous learning units if possible. And we invited them to get hold of any available technology or platform functionality to implement their ideas (and partially offered support to them too), convinced that technology had to be a means rather than a goal in itself. The course also included an overview of NLP techniques and a specific course on the mechanics of ATACK (the authoring tool) and ATAP (the activity management and deployment tool).

During this training we learned that most teachers do not plan how activities will be assessed: that is, they often do not think of the concrete answers to the possible questions they will pose to learners. They do not need to, since they have all the knowledge required to correct learner production any place, any time in their heads (the learner, the activity and the expert model) no matter if the learner production is written or oral. This is crucial since it requires a change in normal working routine.

After the initial training they created learning materials. During creation we interacted with them to make sure that they were not designing activities whose answers were simply impossible to model. For instance, the secondary school teacher who prepared the activity on sustainable production and consumption provided us with a listening comprehension activity including questions such as:

1) *Which is your attitude concerning responsible consumption? How do you deal with recycling? Do you think yours is an ecological home? Are you doing your best to reduce your ecological footprint? Make a list with 10 things you could do at home to reduce, reuse o recycle waste at home.*

All these things were asked in one sole instruction, to be answered in one sole text area. We then talked to the teacher and argued with her the kinds of things that could be modelled using simple one-sentence answers. We ended up reducing the input provided to learners to perform the activity to one

video (initially a text and a video) and prompting learners with the following three questions:

1) *Explain in your words what the ecological footprint is.*
2) *What should be the role of retailers according to Timo Mäkelä?*
3) *Why should producers and service providers use the Ecolabel?*

Similar interventions were done in other activities created by other content creators. But some of them were able to create activities which could be used almost straightforwardly.

3.2 System evaluation

The materials created by teachers were then used in their courses. In the setting that we analyse learners of English as a second language were Catalan and Spanish native speakers between 15 and 17 years old that attended a regular first year of *Batxillerat* (first course for those preparing to enter university studies). They had all been learning English for more than five years, and according to their teacher their CEF level was between A2 and B1. They were all digital literates and they all used the computer on a weekly basis for their studies or leisure (80% daily).

We analyse briefly the results obtained for two of the questions in one of the activities created by the school teacher who authored the learning unit on sustainable production and consumption, namely questions 1) and 2) above. This learning unit was offered to a group of 25 learners.

Overall system performance

Table 1 reflects the number of attempts performed by learners trying to answer the two questions evaluated here: correct, partially correct and incorrect answers are almost equally distributed (around 30% each) and non-evaluated answers are roughly 10%. In non-evaluated answers we include basically answers where learners made a bad use of the system (e.g., answers in a language other than the one learned) or answers which were exactly the same as the previous one for two attempts in a row, which can interpreted in several ways (misunderstanding of the feedback, usability problems with the interface, problems with pop-up windows, etc.) that fall out of the scope of the current analysis.

Table 2 and Table 3 show the number of messages issued by the system for correct, partially correct and incorrect answers for each of the two questions analyzed. The tables distinguish between Form Messages and Content Messages, and Real Form Errors and Real Content Errors –a crucial distinction given our claim that using AutoTutor more open questions could be tackled.[3]

QST	CORR.	PART.	INCORR.	INV.	TOT
1ST	36	23	12	2	**73**
2ND	14	29	36	21	**100**
ALL	**50 (29%)**	**52(30%)**	**48(28%)**	**23(13%)**	**173**

Table 1. Correct, partially correct and incorrect answers.

Table 2 and Table 3 show that the contrast between issued feedback messages (most commonly error messages, but sometimes rather pieces of advice or suggestions) and real problems found in the answers is generally balanced in formal problems (31:15, 8:7 and 41:39 for Table 2; and 6:8, 29:18, and 20:21 for Table 3) independently of the correctness of the answer.

On the contrary, the contrast between issued messages and content problems is much more unbalanced in correct and partially correct answers (139:71 and 84:42 for Table 2; and 45:20 and 110:57 for Table 3) and more balanced for incorrect answers (30:18 for Table 2; and 93:77 for Table 3).

	MESSAGES		REAL ERRORS	
	Form	Cont	Form	Cont
CORRECT ANSWERS	31	139	15	71
PARTIALLY CORRECT	8	84	7	42
INCORRECT ANSWERS	41	30	39	18
TOTAL ANSWERS	80	253	61	131

Table 2. Messages issued vs. real errors for question 1 in the answers produced by learners.

	MESSAGES		REAL ERRORS	
	Form	Cont	Form	Cont
CORRECT ANSWERS	6	45	8	20
PARTIALLY CORRECT	29	110	18	57
INCORRECT ANSWERS	20	93	21	77
TOTAL ANSWERS	55	248	47	154

Table 3. Messages issued vs. real errors for question 2 in the answers produced by learners.

This indicates that generally speaking the system behaved more confidently in the detection of formal errors than in the detection of content errors.

[3] A proper evaluation would require manual correction of the activities by a number of teachers and the corresponding evaluation process.

System feedback analysis

To analyze the system's feedback we looked into the answers and the feedback proposed by the system and annotated each answer with one or more of the tags corresponding to a possible cause of misbehaviour. The possible causes and its absolute frequency are listed in Table 4.

The less frequent ones are bad use of the system on the learner side, bad guidance (misleading the learner to an improper answer or to a more complex way of getting to it), connection failure, and message drawing attention on form when the error was on content.

MISBEHAVIOUR	QUESTION 1	QUESTION 2
CONN-FAIL	1	0
BAD-USE	1	1
FRM-INSTOF-CONT	2	1
BAD-GUIDE	4	2
OOV	11	13
WRNG-DIAG	11	20
FRM-STRICT	33	20
ARTIF-SEP	0	61
SPECS-POOR	1	62

Table 4. Frequent sources of system errors.

The most frequent causes of system misbehaviour are *out-of-vocabulary words*, *wrong diagnoses*, and corrections too *restrictive* with respect to *form*.

Two interesting causes of misbehaviour and in fact the most frequent ones were *artificial separation* and *poor specifications*. The former refers to the system dividing answer parts into smaller parts (and therefore generation of a larger number of issued messages). For instance in a sentence like (as an answer to question 2)

> *The **retailers** need to make sure that **whatever** they label **or they put** in shelf is understandable to consumers.*[4]

the system would generate six different feedback messages informing that some words were not expected (even if correct) and some were found but not in the expected location or form.

In this same sentence above we find examples of too *poor specifications*, where, for instance, it was not foreseen that *retailers* was used in the answer. These two kinds of errors reflect the flaws of the current system: artificial separation reflects a lack of generalization capacity of the underlying parser, and poor specifications reflect the incompleteness of the information provided by novice users, teachers acting as material designers.

4 Concluding remarks

This paper describes software that provides non-NLP experts with a means to utilize and customize NLP-intensive resources using an authoring tool for language instruction activities. Its usability and usefulness have been tested in real instruction settings and are currently being evaluated and analyzed. Initial analyses show that the technology and methodology proposed allow teachers to create contents including automatic generation feedback without the need of being neither a programmer nor an NLP expert.

Moreover, system performance shows a reasonable confidence in error detection given the immaturity of the tool and of its users –following Shneiderman and Plaisant's terminology (2006). There is room for improvement in the way to reduce false positives related with poor specifications. It is quite some work for exercise designers to foresee a reasonable range of linguistic alternatives for each answer. One could further support them in the design of materials with added functionalities –using strategies such as shallow semantic parsing, as in (Bailey and Meurers, 2008), or adding functionalities on the user interface that allow teachers to easily feed exercise models or specific feedback messages using learner answers.

The architecture presented allows for portability into other languages (English and German already available), with a relative simplicity provided that the lexicon for the language exists and contains basic morpho-syntactic information. Moreover, having developed it as a Moodle extension makes it available to a wide community of teachers and learners. The modularity of ATACK and ATAP makes them easy to integrate in other Learning Management Systems.

In the longer term we plan to improve AutoTutor's configurability so that its behaviour can be defined following pedagogical criteria. One of the aspects to be improved is that a computational linguist is needed to add new global error types to be handled or new linguistic phenomena to be considered in terms of block order. If such a system is used by wider audiences, then statistically driven techniques might be employed gradually, probably

[4] One of the expected possible answers was "They need to make sure that whatever they label and whatever they put in the shelves is understood by consumers".

in combination with symbolic techniques –the usage of the tool will provide with invaluable learner corpora. In the meantime AutoTutor provides with a means to have automatic correction and feedback generation for those areas and text genres where corpus or native speaker text is scarce, and experiments show it could be realistically used in real instruction settings.

Acknowledgments

We want to thank the secondary school teachers who enthusiastically volunteered in the creation and usage of AutoLearn materials: Eli Garrabou (Fundació Llor), Mònica Castanyer, Montse Padareda (Fundació GEM) and Anna Campillo (Escola Sant Gervasi). We also want to thank their learners, who took the time and made the effort to go through them. We also thank two anonymous reviewers for their useful comments.

References

Amaral, Luiz A., and Detmar Meurers. On Using Intelligent Computer-Assisted Language Learning in Real-Life Foreign Language Teaching and Learning (Submitted).

Bailey, Stacey and Detmar Meurers (2008) Diagnosing meaning errors in short answers to reading comprehension question. In Proceedings of the Third ACL Workshop on Innovative Use of NLP for Building Educational Applications, pages 107–115, Columbus, Ohio, USA, June 2008.

Carl, Michael, and Antje Schmidt-Wigger (1998). Shallow Post Morphological Processing with KURD. In Proceedings of NeMLaP'98, Sydney.

Chen, Liang and Naoyuki Tokuda (2003) A New Template-Template-enhanced ICALL System for a Second Language Composition Course. CALICO Journal, Vol. 20, No. 3: May 2003.

Estrada, M., R. Navarro-Prieto, M. Quixal (2009) Combined evaluation of a virtual learning environment: use of qualitative methods and log interpretation to evaluate a computer mediated language course. In Proceedings of International Conference on Education and New Learning Technologies, EDULEARN 09. Barcelona (Spain), 6th-8th July, 2009.

Heift, Trude. 2001. Intelligent Language Tutoring Systems for Grammar Practice. Zeitschrift für Interkulturellen Fremdsprachenunterricht 6, no. 2. http://www.ualberta.ca/~german/ejournal/ heift2.htm.

———. 2003. Multiple learner errors and meaningful feedback: A challenge for ICALL systems. CALICO Journal 20, no. 3: 533-548.

———. 2005. Corrective Feedback and Learner Uptake in CALL. ReCALL Journal 17, no. 1: 32-46.

Heift, Trude, and Mathias Schulze. 2007. Errors and Intelligence in Computer-Assisted Language Learning: Parsers and Pedagogues. New York: Routledge.

Levy, Michael. 1997. Computer-Assisted Language Learning. Context and Conceptualization. Oxford: Oxford University Press.

Nagata, Noriko. 2002. BANZAI: An Application of Natural Language Processingto Web based Language Learning. CALICO Journal 19, no. 3: 583-599.

———. 2009. Robo-Sensei's NLP-Based Error Detection and Feedback Generation. CALICO Journal 26, no. 3: 562-579.

Polisca, Elena. 2006. Facilitating the Learning Process: An Evaluation of the Use and Benefits of a Virtual Learning Environment (VLE)-enhanced Independent Language-learning Program (ILLP). CALICO Journal 23, no.3: 499-51.

Quixal, M., T. Badia, B. Boullosa, L. Díaz, and A. Ruggia. (2006). Strategies for the Generation of Individualised Feedback in Distance Language Learning. In Proceedings of the Workshop on Language-Enabled Technology and Development and Evaluation of Robust Spoken Dialogue Systems of ECAI 2006. Riva del Garda, Italy, Sept. 2006.

Rösener, C.: "A linguistic intelligent system for technology enhanced learning in vocational training – the ILLU project". In Cress, U.; Dimitrova, V.; Specht, M. (Eds.): Learning in the Synergy of Multiple Disciplines. 4th European Conference on Technology Enhanced Learning, EC-TEL 2009 Nice, France, Sept. 29 – Oct. 2, 2009. Lecture Notes in Computer Science. Programming and Software Engineering, Vol. 5794, 2009, XVIII, p. 813, Springer, Berlin.

Schmidt, P., S. Garnier, M. Sharwood, T. Badia, L. Díaz, M. Quixal, A. Ruggia, A. S. Valderrabanos, A. J. Cruz, E. Torrejon, C. Rico, J. Jimenez. (2004) ALLES: Integrating NLP in ICALL Applications. In Proceedings of Fourth International Conference on Language Resources and Evaluation. Lisbon, vol. VI p. 1888-1891. ISBN: 2-9517408-1-6.

Shneiderman, B. and C. Plaisant. (2006) Strategies for evaluating information visualization tools: multi-dimensional in-depth long-term case studies. BELIV '06: Proceedings of the 2006 AVI workshop on Beyond time and errors: novel evaluation methods for information visualization, May 2006.

Toole, J. & Heift, T. (2002). The Tutor Assistant: An Authoring System for a Web-based Intelligent Language Tutor. Computer Assisted Language Learning, 15(4), 373-86.