CALL for Open Experiments

Roman Efimov, Maxim Mozgovoy* and John Brine*
The University of Aizu, Tsuruga, Ikki-machi, Aizuwakamatsu, Fukushima, Japan

Keywords: Computer-assisted Language Learning, Intelligent Systems, Virtual Labs.

Abstract: In this paper, we briefly describe the limitations of present CALL systems, caused both by technological factors and by the limited agenda of CALL developers, whose design goals tend not to result in software tools for practical everyday language learning activities. We also note the lack of creative new ways of using computers in language education and a gradual shift towards traditional teaching and learning practices, enhanced with common computer technologies such as multimedia content delivery systems and social media. However, computers can provide more options for interactive learning, as shown by the emergence of virtual labs or virtual sandboxes that support and encourage open experimentation. Such systems are well known in natural sciences, but still have had little impact on the world of CALL software. We believe that the same "free experimentation" approach used in natural sciences can be applied in CALL, and should have a positive impact on the quality of learning, being consistent with constructivist perspectives on language education. In the present paper, we briefly introduce our work-in-progress to develop a system that supports open experiments with words and phrases.

1 INTRODUCTION

When computers became commodities, terms like "computer-assisted *X*" lost some significant part of their initial meaning. We do not refer to "ballpoint pen-assisted writing" or "car-assisted traveling", and yet "computer-assisted language learning," or CALL, is still in common use. In regard to CALL, we should probably imagine dedicated educational systems that somehow "assist" learning in a nontrivial technologically-driven way, but ironically common definitions of CALL simply refer to the use of computers in language learning activities (Levy, 1997). In particular, using an electronic dictionary or watching a foreign-language clip on YouTube are perfect examples of "computer-assisted language learning", though neither an electronic dictionary nor a video-sharing website were explicitly designed to support language learning.

Furthermore, it also seems to us that such general-purpose software is the most widely used and most helpful for the learners. By contrast, there are hundreds if not thousands of available dedicated software packages for language acquisition, but strikingly they are rarely mentioned in numerous "language learning tips" found online (Leick, 2013; Hessian, 2012).

In general, computer technology holds a firm position as a helper within traditional teaching and learning practices. We learn language by listening, speaking, reading, writing, and doing (established) exercises, and computers provide unprecedented support and convenience in these activities. However, overall they still fail to provide fundamentally new teaching and learning practices, unavailable in traditional paper-and-pencil scenarios.

Even dedicated CALL systems (such as the ones developed by companies like Eurotalk, Berlitz or Rosetta Stone) are typically designed as integrated packages of traditional learning materials — audio/video clips, pictures, texts, exercises, and vocabularies. In other words, current CALL systems can be considered primarily as highly usable and modernized versions of traditional "book + tape" self-learning courses. The survey conducted by Hubbard in 2002 revealed that even the CALL experts are not convinced about the effectiveness of educational software. Hubbard notes: "*...it is interesting that questions of effectiveness still tend to dominate. In fact, the basic questions of "Is CALL effective?" and "Is it more effective than alternatives?" remain popular even among those*

* Supported by JSPS KAKENHI Grant #25330410

Efimov R., Mozgovoy M. and Brine J..
CALL for Open Experiments.
DOI: 10.5220/0004936704040408
In *Proceedings of the 6th International Conference on Computer Supported Education* (CSEDU-2014), pages 404-408
ISBN: 978-989-758-020-8
Copyright © 2014 SCITEPRESS (Science and Technology Publications, Lda.)

who have been centrally involved in the field for an extended period of time." (Hubbard, 2002).

We suggest that the reasons are both technological and psychological: many computer technologies relevant to language learning are indeed not mature enough to be used in practical CALL systems, and our traditional learning habits make it hard to design fundamentally new systems that would utilize the full power of today's computing hardware.

2 CALL MEETS TECHNOLOGICAL LIMITS

A number of language learning software insturuments can do more than merely support traditional learning activities, but their overall capabilities are still limited (Hubbard, 2009).

We can add that research efforts in this area are limited, too. For example, Volodina et al. observe that only three natural language processing-backed CALL systems have come into everyday classroom use (Volodina et al., 2012). Furthermore, as noted in (Amaral et al., 2011), "*the development of systems using NLP technology is not on the agenda of most CALL experts, and interdisciplinary research projects integrating computational linguists and foreign language teachers remain very rare*".

Possibly, the only "intelligent" technology that has made its way into some retail CALL systems is automated speech analysis, which is used to evaluate the quality of student pronunciation. Such an instrument is implemented, e.g., in commercial Rosetta Stone software, but its resulting quality is sometimes criticized (Santos, 2011).

We have to state that future development of ICALL systems crucially depends on significant achievements in the underlying technologies. Language learning is a sensitive area, where misleading computer-generated feedback may harm students. So it is impossible to expect any rise of intelligent CALL systems before the related natural language processing technologies improve vastly.

3 THE PROBLEM OF LIMITED AGENDA

However, computers can significantly improve learner experience even without advanced AI technologies, and provide "killer features" that are inherently computer-backed and cannot be easily reproduced in traditional environments. A good example of such an "inherently computer" system is any electronic dictionary, as it can implement a number of unique capabilities that create new use cases:

- approximate word search;
- partial search (find a word fragment);
- full-text search (find example phrases);
- arbitrary word form search;
- handwritten characters input.

Surprisingly, most popular dictionaries implement only a fraction of this list. It should be noted that none of the mentioned functions require the use of any immature research-stage technologies, and can be implemented with established methods.

Another example is spaced repetition-based flashcards software such as Anki (Elmes, 2013) or SuperMemo (Wozniak, 2013). While in spaced repetition can be exercised without a computer, it is a laborious process, hardly tolerable for most learners. So despite being relatively simple, these tools are efficient learning aids (as spaced repetition practices are proven to be effective (Caple, 1996)), and yet seldom mentioned in CALL-related papers.

So, it seems that CALL experts have not paid much attention to the development of everyday language learning tools. This situation is unfortunate, as it is inconsistent with the current trend of seamless integration of technologies into existing learning activities and with declarations of a preference for a student-centered approach that should presumably allow learners to follow their preferred learning styles or at least to ensure higher flexibility of the learning process.

4 VIRTUAL SANDBOXES

Such a technology-backed, student-centered approach is already implemented in a number of educational systems for the disciplines such as physics, chemistry, and computer science. Notably, there are sandbox-like environments (or "virtual labs") that do not restrict their users and do support open experimentation.

For example, Open Source Physics project (Christian et al., 2013) collects together a vast amount of interactive physical simulations with user-adjustable parameters. The 2D physics sandbox Algodoo is positioned by its authors as *"the perfect tool for learning, exploring, experimenting and laborating* [sic] *with real physics"* (Algoryx, 2013). The ChemCollective collection (Yaron et al., 2013)

includes a number of ready setups for chemical experiments as well as a virtual lab for open exploration. The JFLAP environment (Rodger, 2013) allows students to create, analyze and test finite-state machines — the devices that constitute the basis of computer science.

We consider such systems as great examples of well-grounded uses of computer technology in education. Virtual labs provide safe and controlled environments in which students can test their ideas, and in this sense they can be likened to flight simulation software, used to train pilots: the students perform predefined training routines, but also can experience the outcome of any arbitrary maneuver. Furthermore, virtual labs contribute to the modeling of the problem domain in the learner's mind, and thus are consistent with constructivist views on educational process.

It is interesting to note that from the technological point of view, virtual labs are not necessarily complex systems. The possibility of open experimentation outweighs many technical limitations and constraints.

Unfortunately, environments for open experiments are barely provided by the existing CALL systems. This perhaps can be attributed to the unclarity of the notion of an "experiment" in language learning. It is evident, however, that a large portion of active language learning is related to the process of combining words and phrases into meaningful sentences, and the analysis of the subsequent feedback. We learn a language both by comprehending other people's speech and writing, and by creating our own phrases that are to be tested for admissibility by our interlocutors.

Within such a concept of experiments, even a feature-rich electronic dictionary can be a powerful experimental tool in the hands of an avid learner. Indeed, with full-text search it is possible to check actual word use, test the correctness of certain word combinations, the compatibility of certain prefixes with certain stems, etc.

The ways in which students could do "experiments with the language" are still to be identified. Here we can only quickly introduce our own work-in-progress system that is intended to help language learners master basic grammatical rules.

5 TOWARDS WORDBRICKS

One of the most basic aims of language learning is to train the ability to formulate grammatically correct sentences with known words. Unfortunately, traditional exercises lack active feedback mechanisms: learners are unable to "play" with language constructions to find out which word combinations are admissible and which are not. The best (and maybe the only) way to train active writing skills is to *write* (essays, letters…), and to get the writings checked by the instructor. Some intelligent CALL systems, such as Robo-Sensei (Nagata, 2009), can assess students' writings by using natural language processing technologies, but the success of these instruments is limited.

We suggest that active skills of sentence composition can be improved by forming a consistent *model of language* in the learner's mind. Metaphorically speaking, the difference between a "consistent model" and a set of declarative grammar rules in this context is the same as the difference between a Lego construction kit and a lengthy manual describing which Lego bricks can be connected and in which ways. A child does not need manuals to play Lego: the rules of brick linkage can be easily inferred from brick shapes and with some trial-and-error process. Unfortunately, there is no such way to easily check whether it is correct to combine certain words in a sentence.

The idea of modeling syntactic rules with shaped bricks was implemented in the educational programming environment Scratch (Resnick et al., 2009). In Scratch, individual syntactic elements of a computer program are represented with shaped bricks that have to be combined to constitute a program (Figure 1a). While Scratch code may have logical errors, syntactically it is always correct, since it is impossible to combine mismatching bricks.

Scratch's graphical editor is not just a simpler way to write computer programs, helpful for the beginners. It can be treated as a *construal* (Gooding, 1990) that forms a model of a programming language in the learner's mind, though this aspect is not explicitly emphasized in Scratch.

In our research, we are working towards implementation of a similar scheme for natural language sentences. Undoubtedly, natural language grammar is much more complex and less formal than the syntax of any programming language. However, for the purposes of novice language learners, it is reasonable to teach restricted grammar (as it happens in traditional language teaching), which is technologically feasible.

Even in the case of Scratch, the design of brick linkage principles is not trivial. One important problem is to make sure that the links between the bricks reflect *actual* structure of the corresponding computer program. For example, a loop control

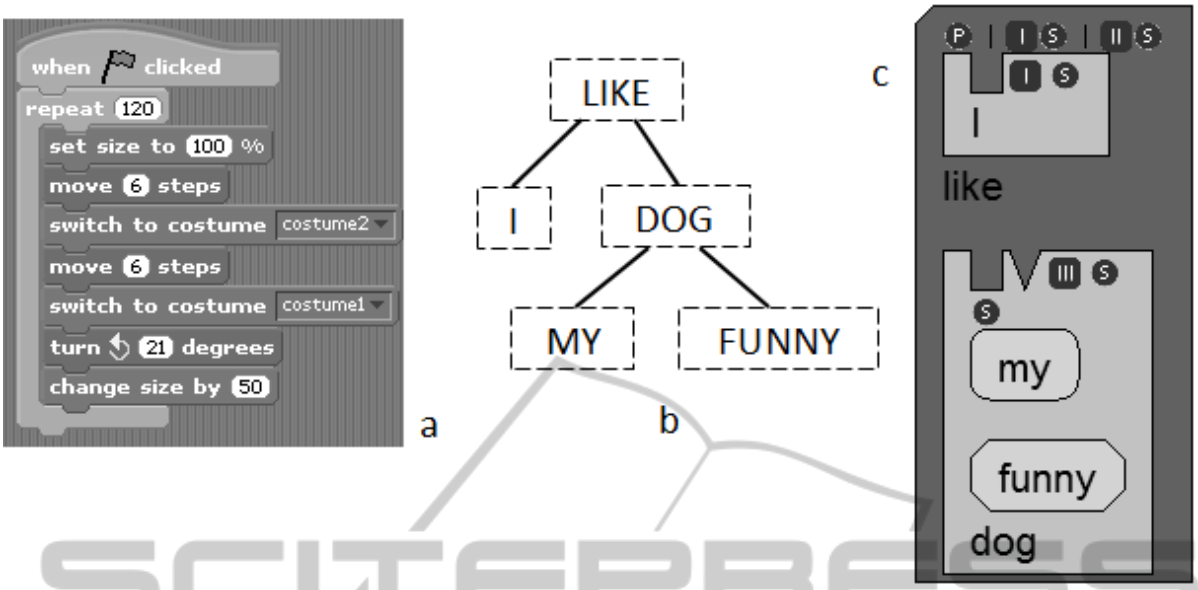

Figure 1: a) A fragment of Scratch program; b) Dependency tree of the phrase "I like my funny dog."; c) Dependency tree of the same phrase in the form of 2D puzzle.

SCITEPRESS
SCIENCE AND TECHNOLOGY PUBLICATIONS

structure can be represented with the separate "Begin Loop" and "End Loop" bricks that surround bricks that constitute the loop body; however, such a design would make a false impression that "Begin Loop" and "End Loop" are independent program elements. Instead, a loop in Scratch is represented with a single C-shaped brick that embraces the loop body.

It is much harder to identify a consistent set of rules that control such linking principles of a natural language-based system. However, they are actually considered in a number of linguistic theories. In particular, we base our rules on the principles of *dependency grammars* (Nivre, 2005). Existing guidelines, such as the Stanford Typed Dependencies Manual (Marneffe & Manning, 2008) describe in detail how the words in the given sentence should be linked to form a structure consistent with the ideology of dependency grammars. For example, a subject and an object should be directly connected to their head verb; an adjective should be directly connected to its head noun (Figure 1b).

The resulting structure of a sentence is represented with an *n*-ary tree. While this structure is linguistically correct (according to the theory of dependency grammars), it arguably might be difficult for learners to master it. Therefore, it is our challenge to represent such trees as two-dimensional brick puzzles. Furthermore, dependency grammars do not express word order, while it has to be reflected in the resulting brick structure (Figure 1c).

The proposed learning environment can be used in different scenarios, but we would emphasize again the possibility to perform open experiments. Learners will be able to test which word combinations are admissible and why.

We should also note that it is an open question whether language learners (at least in the early stages of learning) should study sentence structure. However, we believe that some gentle exposure is fruitful, especially for learning languages with rich morphology, where a single change in one word may trigger changes in several of its dependent words.

6 CONCLUSIONS

Computer technologies are widespread in modern language education. Some directions in CALL research, such as intelligent systems, have not yet been as fruitful as anticipated, while other developments, such as multimedia and networking capabilities, have surpassed our expectations.

It seems that the present agenda of CALL research is primarily focused on exploring recent technologies such as ubiquitous computing or Web 2.0. However, we see that even basic language learning tools, such as electronic dictionaries or flashcard software, would benefit from greater attention by CALL developers. Ubiquitous and mobile computing technologies stimulate learner's independence, but language learners still lack tools that support

independent language exploration and make use of computing hardware not just as a platform for the delivery of multimedia data.

We would especially favor more developments in open experimentation language software. This direction has promising advancements in a variety of scientific fields, but not yet in CALL.

REFERENCES

Algoryx, 2013. *Algodoo: 2D Physics sandbox*, www.algodoo.com.

Amaral, L., Meurers, D. & Ziai, R., 2011. Analyzing learner language: towards a flexible natural language processing architecture for intelligent language tutors. *Computer Assisted Language Learning* 24 (1), 1–16.

Caple, C., 1996. The Effects of Spaced Practice and Spaced Review on Recall and Retention Using Computer Assisted Instruction. Ann Arbor, MI.

Christian, W., Belloni, M. & Brown, D. et al., 2013. *Open Source Physics*, www.opensourcephysics.org.

Nivre, J., 2005. Dependency grammar and dependency parsing. *MSI Report* 05311, Växjö University.

Elmes, D., 2013. *Anki: Friendly, intelligent flash cards*, www.ankisrs.net.

Gooding, D., 1990. *Experiment and the making of meaning: Human agency in scientific observation and experiment*. Kluwer Dordrecht.

Hessian, J., 2012. Tips for Studying Foreign Languages, University of Illinois at Chicago. www.uic.edu/depts/ace/foreign_languages.shtml.

Hubbard, P., 2002. Survey of unanswered questions in Computer Assisted Language Learning, Stanford University. www.stanford.edu/~efs/callsurvey/index.html.

Hubbard, P., 2009. A General Introduction to Computer Assisted Language Learning. In: Hubbard, P. (ed.) *Computer-Assisted Language Learning (Critical Concepts in Linguistics)*, New York: Routledge, pp. 1–20.

Leick, V., 2013. Tips on managing your language learning, University of Birmingham. www.birmingham.ac.uk/facilities/cml/learnersupport/skills/managing.aspx.

Levy, M., 1997. *Computer-assisted language learning: Context and conceptualization*. Clarendon Press, Oxford [u.a.].

Marneffe, M.-C. de & Manning, C. D., 2008. Stanford typed dependencies manual. *Stanford University*.

Nagata, N., 2009. Robo-Sensei's NLP-based error detection and feedback generation. *Calico Journal* 26 (3), 562–579.

Resnick, M., Silverman, B. & Kafai, Y. et al., 2009. Scratch: Programming for All. *Communications of the ACM* 52 (11), 60-67.

Rodger, S., 2013. *JFLAP*, www.jflap.org.

Santos, V., 2011. Rosetta Stone Portuguese (Brazil) levels 1, 2, & 3 Personal Edition Version 4 (TOTALe). *Calico Journal* 29 (1), 177–194.

Volodina, E., Borin, L., Loftsson, H., Arnbjörnsdóttir, B. & Leifsson, G. Ö., 2012. Waste not, want not: Towards a system architecture for ICALL based on NLP component re-use. In: *Proc. of the SLTC 2012 workshop on NLP for CALL*, pp. 47–58.

Wozniak, P., 2013. *SuperMemo*, www.supermemo.com.

Yaron, D., Ashe, C., Karabinos, M., Williams, K. & Ju, L., 2013. *ChemCollective*, www.chemcollective.org.

SCITEPRESS SCIENCE AND TECHNOLOGY PUBLICATIONS