Waste not, want not: Towards a system architecture for ICALL based on NLP component re-use

Elena Volodina, Lars Borin
Språkbanken (Swedish Language Bank)
University of Gothenburg, Sweden
first.last@svenska.gu.se

Hrafn Loftsson
School of Computer Science
Reykjavík University, Iceland
hrafn@ru.is

Birna Arnbjörnsdóttir
School of Humanities
University of Iceland, Iceland
birnaarn@hi.is

Guðmundur Örn Leifsson
School of Engineering and Natural Sciences
University of Iceland, Iceland
gol1@hi.is

Abstract

It is a surprising fact that, despite the existence of various mature Natural Language Processing (NLP) tools and resources that can potentially benefit language learning, very few projects are devoted to development of Intelligent Computer-Assisted Language Learning (ICALL) applications. This paper presents an on-going collaborative project whose overall aim is to develop an open-source system architecture for supporting ICALL systems that will facilitate re-use of existing NLP tools and resources on a plug-and-play basis. The two language teams – Icelandic and Swedish – have tested the architecture design by implementing two ICALL applications which convincingly show how principles defined by Service-Oriented Architecture (SOA), with web services as implementation technology, can benefit re-use of existing NLP components in ICALL applications. This paper introduces the project, provides the theoretical and practical background, describes the different paths adopted within the two language teams, and presents the first results.

1 Introduction

The project described in this paper was prompted by the surprising fact that existing NLP tools and resources do not tend to find their way into the language learning classroom, despite their obvious potential uses in language learning. The reasons may be twofold. On the one hand, there is a lack of interested sponsors. On the other hand, there is a general lack of interest in the NLP community in CALL applications. Borin (2002), for example observed that "[...] while certainly not part of the core of NLP, CALL seems not to have a place even in its periphery", and "[...] most NLP work on Nordic languages has nothing to do with CALL". While this might have changed for English, and a small number of other languages in the past ten years,[1] it still holds true for the Nordic languages.

We are aware of only three ICALL[2] systems that are an integral part of a real-life foreign language program in universities today: TAGARELA for Portuguese (Amaral and Meurers, 2011; Amaral et al., 2011), E-tutor for German (Heift, 2003), and Robo-Sensei for Japanese (Nagata, 2009). It seems that the few systems that have been developed are either copyrighted and restricted by high licensing fees – and hence too expensive for universities and schools – or fall short of the required quality in linguistic or pedagogical functionality.

This situation calls for a change. Since ICALL is a truly interdisciplinary field, it is important that researchers from several areas, like linguistics, pedagogy, NLP, and human-computer interaction (HCI) cooperate for the purpose of making ICALL projects successful. In view of that, we have joined

[1]Major NLP conferences tend to organize workshops on the use of NLP technologies in language learning, e.g. NAACL and COLING. The same holds true for the main conferences within computer-assisted language learning where AI and NLP approaches are studied within the area of pedagogy, e.g. CALICO and ICCE.

[2]Intelligence in CALL systems can be understood differently by different researchers. In this paper, we define ICALL as NLP-based CALL, i.e. intelligence in CALL is ensured through the use of NLP tools and resources like parsers, taggers, corpora, lexicons, etc.

forces in order to design and develop open-source system architecture for supporting ICALL systems. The architecture is open-source in order to encourage participation from other researchers and developers, and to facilitate re-usability of existing NLP tools and resources in the area of CALL. This is an ongoing collaboration, and some preliminary results and earlier versions of the implementations described below have been presented (in much less detail and without evaluation) in other contexts (Volodina and Borin, 2012; Volodina et al., 2012).

Our main argument is that the use of NLP tools and annotated resources can ensure linguistic analysis of input data, thus adding generative power. This is accomplished by applying the same analysis model to different (authentic) language samples, e.g. for generating exercises or detecting errors in learner text production. This, in our view, will not only relieve teachers of monotonous tasks that can be performed by computers, but can also support autonomous learning by students. And last, but not least, we hope it will increase the applications of NLP tools among CALL end-users.

For this purpose, we need access to existing NLP tools (e.g. sentence segmenters, tokenisers, part-of-speech [PoS] taggers, lemmatisers, syntactic parsers, error parsers, spell-checkers, etc.), as well as to existing (available and reliable) annotated resources (e.g. corpora, lexicons, learner-oriented word lists, etc.). We intend to re-use existing NLP tools and resources as much as possible (as opposed to developing new ones).

However, one problem is that most available resources and tools are difficult to deploy in CALL applications since (1) they are monolithic and inflexible and need to be individually adapted to each new application; (2) they are not readily available as the rights to their use are held by individuals or institutions all over the world and they are physically located in different places; and (3) they are not interoperable via standardised interfaces.

In order to achieve more flexibility, we need to cooperate with the owners of tools and resources. We need a standardisation effort within the ICALL community. One of the goals of this project is to design an architecture for deploying NLP tools and resources that will have well-defined principles and requirements, as well as provide easy-to-follow guidelines. We hope it will generate an interest in ICALL standardisation, and at best, if we are fortunate – encourage owners to provide a wrapper layer to their tools and resources making them re-usable in ICALL (and other) applications via web services. One overarching goal of our project is to test web services as a possible approach to making tools and resources available for re-use.

To avoid being too abstract, we are also implementing two end-user applications that will help us (1) test and refine the architecture; (2) produce guidelines for making a service wrapper layer to the tools and resources; (3) define relevant input/output formats and documentation standards; as well as (4) demonstrate the architecture design in practice for potential end-users and web service providers.

The rest of this paper is structured as follows. In section 2, we present the technical framework which we have adopted for the development of our architecture. Sections 3 and 4 are descriptions of the two examples where web services are used in development of ICALL applications, one for Icelandic (section 3) and one for Swedish (section 4). Section 5 concludes the paper with some general considerations about the effectiveness of the adopted approach and its future.

2 Technical framework

2.1 Background

The idea of re-usability as a paradigm for software development is not original. It is well-known that programmers often make chunks of their code available to each other in order to save time on implementation of something similar. With the appearance of the Free Software Foundation[3] in 1984, developers could have access to each other's code, copy it, modify and built upon it, which speeded up development times and reduced costs. Initiatives like that are very popular, but they have some limitations: first, the code comes in various different programming languages and it is not certain that it will be available in the language you need; second, they often lack documentation with explanation of their design or how the program works; and third, they are often centered around one problem specific for the current project, which is most probably not the one that is relevant for your needs (Wood, 2008).

Standardisation is a key notion in such initiatives. In addition to work carried out on standardisation of e-learning (IMS Global Learning Consortium,[4] ADL,[5] etc.) and of text

[3]http://www.fsf.org/
[4]http://www.imsglobal.org/
[5]http://www.adlnet.gov

corpus and lexicon resource formats (TEI,[6] EAGLES,[7] etc.), some successful standardisation efforts have been initiated for NLP components as well, e.g. GATE,[8] NLTK,[9] UIMA,[10] which are frameworks for integrating NLP tools and resources. However, the NLP components are still bound to particular programming languages: Java (GATE and UIMA) and Python (NLTK).

2.2 NLP component re-use through web services

The original initiative of re-using different existing programming functionalities in applications without re-writing the code is known as *Service-Oriented Architecture* (SOA).[11] SOA is an architectural style based on a set of global principles and requirements defined first by Erl (2005) and later by the SOA Manifesto Working Group.[12] SOA emphasises implementation of components as modular services that can be re-used by other clients. The main idea is that, despite different programming languages or platforms, the existing functions have a common communication layer consisting of a well-defined interface, where the user can formulate a request and get a response which can be re-used in other applications. The data is passed in standardised formats between the service and a client or between several services through coordinated calls. The key requirements are interoperability, re-use, standards-compliance, and well-documented metadata. Services can be made accessible to a closed group, e.g. within a company's intranet, or be open to anybody concerned via internet, for a fee or for free. Services are loosely coupled, and can be combined and re-combined for different purposes in production of other applications.

If SOA is an architectural style, then web services[13] are an implementation technology (one of many) for SOA. Web services make programs accessible through Internet protocols independent of platforms or programming languages. They can represent new applications or wrap around existing tools, becoming a port of access to them. Each service in the SOA architecture has, in turn, its own architecture. It includes all the resources used by a service, e.g. databases, software components, other services, and the physical design of their communication.

The basic principles and ideas behind SOA, particularly with web service technology as its implementation form, seem to be the answer to the question of accessibility of existing NLP tools and resources over the internet, and not only for ICALL applications. The software can still be residing on the original server and in the original programming language. It is the wrapper layer (web service) that makes it available to the users world-wide.

2.3 A platform for supporting ICALL

From an end-user perspective, Learning Platforms (LP), virtual learning environments (VLE) and learning/content management systems (LMS/CMS) serve different pedagogical purposes. They are different types of online services facilitating communication between teachers and students, e.g. for delivery of course-related information, resources and tools; as well as for synchronous (e.g. web chats, video conferences) and asynchronous (e.g. forums) meetings between students and teachers where course-related questions can be discussed. Such systems model a real-life communication between all involved parties, and may be used either in e-learning/ distance learning, i.e. without any class meetings, or as enhancement of face-to-face courses. Examples of such platforms are Moodle (Martín-Blas & Serrano-Fernández, 2009) and Fronter.[14]

Viewed from a developer's perspective, LPs can be compared to operating systems since they share some common characteristics, e.g. they are composed of a number of web-based applications that can be run within some environment.

ICALL is a specific area of learning, and thus a platform aimed at language learning requires a more specific design. Further, a platform offering intelligent analysis of language input needs to be designed for re-use of the components that can perform such analysis.

We therefore define an ICALL platform in technical terms as a structured backend, i.e. a "machinery" for deploying different NLP tools

[6]http://www.tei-c.org

[7]http://www.ilc.cnr.it/EAGLES96/home.html

[8]http://gate.ac.uk

[9]http://nltk.org/

[10]http://uima.apache.org/

[11]Architecture is a description of a system, defining its purpose, functions, externally visible properties and interfaces; including the description of its internal components and interoperability along with the principles governing its design, operation and evolution. It is thus a design of a system, not its implementation (Srinivasan and Treadwell, 2005).

[12]www.soa-manifesto.org, 2009

[13]A web service is an implemented software component that can be accessed via a network to provide functionality to a service requester/client (Srinivasan and Treadwell, 2005).

[14]http://com.fronter.info/product/

and lexical resources for supporting language learning activities, as well as specifically tailored algorithms for various language learning tasks (e.g. exercise generators). We neglect most of the administrative and content management functions that pedagogical platforms described above usually imply.

In particular, we build two ICALL platforms on SOA principles where the collection of web-services are the basis of the platforms.[15] The user interface,[16] on the other hand, is a top layer that is used for delivering the results of existing web services and should not necessarily be viewed as an integral part of the platform. It is rather an environment for presenting the output of web services and may be developed by different users according to their tastes and needs.

The advantage of separating ICALL modules into a frontend (user interface) and a backend (web services) parts is that the algorithms for required language learning task can be made language independent, i.e. they will rely only on the availability of corresponding NLP tools and lexical resources for other languages with the same type of annotation.

Another advantage is that in case we optimise or change the backend algorithm, the user interface remains unaffected; it is just a container for collecting user input and for showing the results of the web service.

One more advantage is that the web services are made re-usable for any other applications/user interfaces.

That is our starting point, and we are currently testing this approach by building two ICALL applications based on NLP components accessed through web services. The two ICALL applications are aimed at different language learning tasks: error analysis and feedback on L2[17] learner written input for the Icelandic partner; and corpus-based exercise generation for the Swedish partner, as described in the sections that follow.

3 ICALL through web services – an Icelandic example

3.1 NLP and ICALL for Icelandic

A decade ago, Icelandic could have been categorised as a less-resourced language, i.e. a language for which only a few, if any, NLP resources exist. Ten years later, the situation has changed dramatically (Rögnvaldsson, 2008). A number of BLARK[18] (Krauwer, 2003) components have now been developed, e.g. the open-source *IceNLP* toolkit,[19] a collection of tools for processing and analysing the Icelandic language (Loftsson and Rögnvaldsson, 2007b).

Among other tools, IceNLP contains a tokeniser, the PoS tagger IceTagger (Loftsson, 2008), and the shallow parser IceParser (Loftsson and Rögnvaldsson, 2007a). IceTagger, which performs morphosyntactic disambiguation, is the current state-of-the-art tagger for Icelandic (Loftsson et al., 2009). IceParser, which receives disambiguated input from a PoS tagger and whose task is to label constituents and syntactic functions, is the only publicly available parser for the language.

Two lexical resources are important parts of the Icelandic BLARK. First, the Icelandic Frequency Dictionary (Pind et al., 1991), a PoS-tagged corpus, and, secondly, the morphological database BÍN[20] (Bjarnadóttir, 2005). Both resources are available for research purposes, while the data of the latter can be used for developing language technology applications.

Currently, no ICALL application exists for the Icelandic language. On the other hand, the development of the web course (CALL application) *Icelandic Online* (IOL)[21] began in 2000. The sequential course is pedagogically driven in that instructional goals were served by the available pre-web 2.0 technology (the opposite was true for most CALL courses at the time). The technology used by IOL was only limited by the Digital Divide. This meant that, at the time, students in countries other than the most technologically advanced did not have the bandwidth to download websites heavily based on videos and interactive learning objects with many images.

IOL I and II were launched in 2004 and 2005. The goal of those courses is to introduce the structure and lexicon of Icelandic in a meaningful context using 40 pre-programmed learning objects, the contents of which can be altered and geared to the particular pedagogical goals of the lesson. The first courses were also heavily dependent on

[15]The terms platform and backend are used interchangeably in the text.

[16]The terms GUI, user interface, and frontend are used interchangeably in this text.

[17]L2 covers both foreign and second language learning.

[18]BLARK – Basic LAnguage Resource Kit, a joint initiative for European countries which has been extended to many other than European languages, see http://www.blark.org/.

[19] http://icenlp.sourceforge.net

[20] http://bin.arnastofnun.is

[21]http://icelandiconline.is

individually programmed interactive Flash lessons that introduced new vocabulary and grammar. The limitations of the courses were that they taught perceptive language with limited activities for students to practice productive skills other than form focused discrete vocabulary and grammar exercises (Arnbjörnsdóttir, 2004).

In 2010, Icelandic Online 3 and 4 and IOL for Immigrants were launched. These courses use the 40 learning objects but also introduce lesson content through authentic videos, texts and interactive websites, chosen and sequenced to advance the lesson goals. This was post web 2.0 which made available different social networks and functionalities that allow learners to interact with each other and practice their target language and negotiate meaning in social situations. This has been made full use of in Icelandic Online 3 and 4 (Arnbjörnsdóttir, 2008).

Currently, Icelandic Online has almost 90,000 registered users and has received universally positive feedback. IOL has revolutionised accessibility to Icelandic language and culture for teachers and students at the University of Iceland and worldwide. IOL is free and open to all.

To date, technology has not been able to provide CALL projects, like IOL, with meaningful intelligent feedback on second language writing. Despite the availability of spelling and grammar checkers in some languages, these tend to correct, rather than instruct, which is not always optimal for language learning.

3.2 The Icelandic platform

In the Icelandic part of the project, the platform connects various pre-existing NLP tools. Internally, the platform uses a particular XML format, the Text Corpus Format (TCF), proposed in the WebLicht SOA project (Hinrichs, 2010), for communication of information between the various components. Each annotation (e.g., at the level of tokens, PoS tags, or constituents) is stored in a separate layer, but all annotations for a particular text is stored in a single XML file. In addition to using the layers proposed in the WebLicht project, we have added our own layer for information about grammatical errors.

Using a web service, a user asks the platform to carry out a given task. Thus, the platform does not need to be set up on the user's machine. Moreover, the server running the web service and the platform do not have to be located on the same machine.

3.3 Writing support for second-language learners

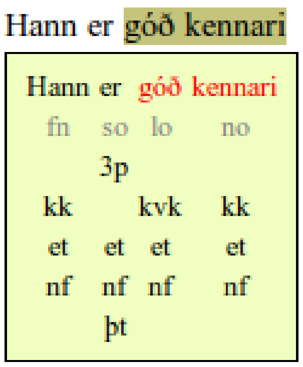
In IOL, second-language learners of Icelandic can receive feedback from a teacher on short written texts. Currently, teachers use special codes for hand-marking specific types of errors, i.e. spelling errors, feature agreement errors, case errors in objects of verbs, etc.

In order to automate part of the hand-marking, and to test our platform, we are currently in the process of developing a web service which allows students of IOL to send texts to the service for the purpose of detecting particular types of grammatical errors. This will allow the students to correct potential errors and re-submit the texts for error detection again, and so forth, before finally submitting the text to the teacher. The web service merely identifies error candidates, but does not attempt to correct errors. At this stage, the goal is to help students correct second language grammar issues, and free instructors to focus on content.

The web service uses the platform, which, in turn, uses tools from the IceNLP toolkit, to detect the following types of grammatical errors, chosen for this first version: (1) feature agreement errors in noun phrases, i.e. errors in gender, number and case; (2) feature agreement errors between subjects and verb complements; (3) feature agreement errors between subject and verbs, i.e. errors in person and number; and (4) incorrect case selection of verb objects.

In using the feedback feature, the student inputs Icelandic text through a web application. The application submits the text to a web service, requesting it to analyse the text and carry out error detection. In turn, the platform calls components from IceNLP for carrying out the given tasks. IceNLP outputs XML in TCF, which the platform forwards to the web service, which in turn sends it back to the client application. The TCF contains all information from the analysis, i.e. information about the individual tokens, their PoS tags, individual constituents and error candidates. The client application converts the TCF to HTML and displays the resulting page to the student, where the original text submitted is shown with error candidates highlighted. In addition, by clicking on a word in a given sentence, the student can see morphological information for each word of the sentence.

Figure 1 shows the feedback given to a student for the sentence *Hann er góð kennari* 'He is (a) good teacher', in which the adjective *góð* 'good (feminine)' does not agree in gender with the following noun *kennari* ‘teacher (masculine)’ in the noun phrase *góð kennari*. The phrase containing the disagreement is displayed, as well as morphological information for each word.


Hann er góð kennari

Hann	er	góð	kennari
fn	so	lo	no
	3p		
kk		kvk	kk
et	et	et	et
nf	nf	nf	nf
	þt		

Figure 1. Feedback given to a student for a sentence containing a disagreement in a noun phrase.

Preliminary tests of the application have been carried out with two groups of students – a group of 11 advanced and 12 intermediate students in a summer course in Icelandic as a foreign language. The purpose of the test was twofold: First to elicit feedback from students about their experiences using the application and, second, to test the functionality of the application itself – the accuracy of the error detection.

In general students found the system helpful for error detection and that it aided them in their writing. Most found the directions for use clear. Two respondents wanted clearer suggestions for corrections or even declension tables to be attached to the system. The latter could be accomplished using the morphological database BÍN (see above).

The accuracy of the error detection was evaluated using the first texts submitted by the second group (12 intermediate students). The results are shown in table 1. In total, the system pointed to 25 grammatical errors, out of which 19 were true positives. This is equivalent to 76% accuracy, which is too low for practical use. Note, however, that the third error type, feature agreement errors between subject and verbs, is mainly to blame. Out of seven error candidates signalled by the system for this error type, only three were true positives. All the four false positives are due to the same error made by the error detector when analysing a sentence like: *Konan og drengurinn voru að þvo* ... ‘The woman and the boy were washing … ’. For this sentence, the error detector signals a disagreement in number between the singular noun phrases ‘the boy’ and the verb form ‘were’, not taking into account that the two singular noun phrases ‘The woman’ and ‘the boy’ indeed constitute a plural subject! When we account for this, both the precision and the recall will presumably increase.

Error type	**Precision**	**Recall**
agreement errors in noun phrases	80%	100%
agreement errors between subjects and verb complements	100%	87.5%
agreement errors between subjects and verbs	42.9%	42.9%
incorrect case selection of verb objects.	100%	50%
All error types	76%	76%

Table 1. Accuracy of the error detection.

Overall, we feel that the system has shown its value as a first step in the development of a semi-automatic writing feedback feature for Icelandic as a second language.

4 ICALL through web services – a Swedish example

4.1 NLP and ICALL for Swedish

Language technology research has a long history in Sweden, going back to the 1960s, and is conducted in a number of groups at the main Swedish universities and in some groups in industry. Consequently, most of the basic BLARK components exist for Swedish in quite stable and mature forms. For example, there are several PoS taggers and parsers, annotated reference corpora, and large lexical databases with morphological analysers available for Swedish, many (but not all) under open-source licenses.

Swedish ICALL has a shorter history. In recent years, there have been four main, partly overlapping, strands of research (ignoring speech-based ICALL, which is also being pursued at the Royal Institute of Technology in Stockholm, but which is out of scope for this paper) (see also Borin, 2006):

(1) Supporting reading of authentic texts by automatic selection of texts containing vocabulary and linguistic constructions at a suitable level for a particular language learner proficiency level (Nilsson and Borin, 2002).

(2) Automatic generation of focus-on-form exercises from annotated corpora, for PoS and syntactic functions such as subject and object (the ITG project;[22] Saxena and Borin, 2002; Borin and Saxena, 2004), and for vocabulary (Volodina, 2010).

(3) Writing support for second-language learners using online (bilingual and monolingual) lexicon access, and spelling and grammar checkers (the Grim project; Knutsson, 2005).

(4) Research on the characteristics of learner language and text complexity with an explicit aim of informing the research described under the previous three points (Magnusson and Johansson Kokkinakis, 2008; Johansson Kokkinakis, 2009).

Both the ITG project (2) and the Grim project (3) have resulted in concrete ICALL applications. The ITG application is open-source and is maintained by University of Gothenburg. It has been used extensively in university-level linguistics courses at the universities in Uppsala and Stockholm, and also in a high school in Uppsala. Its point of departure is what Second Language Acquisition (SLA) researchers have dubbed "focus-on-form" (FoF; contrasted to more traditional form-based drills, referred to as "focus-on-formS" in the SLA literature):

> Whereas learners are able to acquire linguistic forms without any instructional intervention, they typically do not achieve very high levels of linguistic competence from entirely meaning-centered instruction. For example, students in immersion programs in Canada fail to acquire such features as verb tense markings even after many years of study. This had led second language acquisition researchers [...] to propose that learners need to do more than simply engage in communicative language use; they also need to attend to form. (Ellis et al., 2002: 401)

In the ITG application, annotated Swedish text corpora are the basis for guided form exercises as well as curiosity-driven corpus exploration of particular linguistic features (the application includes a general corpus search interface), in both cases using authentic language material directly from the corpora, rather than made-up exercises and examples.

The Grim writing support application is not open-source (although the language tools used in it are) and it can be accessed only via a web page. Both ITG and Grim use a technology for the user interaction with the tool *Java Web Start* which was state of the art at the time, but which practical experience shows is not the optimal solution today, when web technology has developed to a point where pure web solutions will provide equivalent or better functionality in a much more transparent way to the user. Important for our purpose is that the language tools used in both these applications are to a large degree open-source and independent of the technology for realising the user interaction part.

4.2 Lärka and its architecture

In designing the new architecture for the Swedish application, we first ported the existing Swedish FoF exercises developed for the ITG application and started adding the Swedish vocabulary exercises developed by Volodina (2010). Having the existing ITG exercises allows us to quickly assess the viability of the architecture for this kind of application. Together with the new modules to be developed in this project, they make up a broad and varied spectrum of ICALL applications which will allow us to test the flexibility of the architecture. The ITG exercises use manually annotated corpora and although the text material is authentic, it is also now slightly dated and becoming more so all the time. One goal of this project is thus to adapt the language tools at our disposal with the aim of achieving the same kind of functionality using arbitrary text, e.g. from the internet. Another goal is to extend the range of FoF exercises offered and to explore how these exercises should be connected to other language learning activity types.

The application developed as a part of this project is web-based and has been given the name *Lärka*[23] *(*LÄR språket via KorpusAnalys 'learn language by corpus analysis'), with the English equivalent *Lark* (Language Acquisition Re-using Korp). The two main guiding principles for the implementation of Lärka have been modularity and re-use. The main components of Lärka are, as shown in figure 2):

- *frontend* – the graphical user interface that handles user interaction, sends requests to the backend, prettifies its output and assigns behaviour to the buttons and fields;

- *backend* – a number of web services for creating language training exercises, selecting

[22]ITG stands for *IT-based collaborative learning in grammar*; see http://spraakbanken.gu.se/swe/itg

[23]The Swedish word *lärka* means 'lark' (the bird), hence the logo; see http://spraakbanken.gu.se/larka/

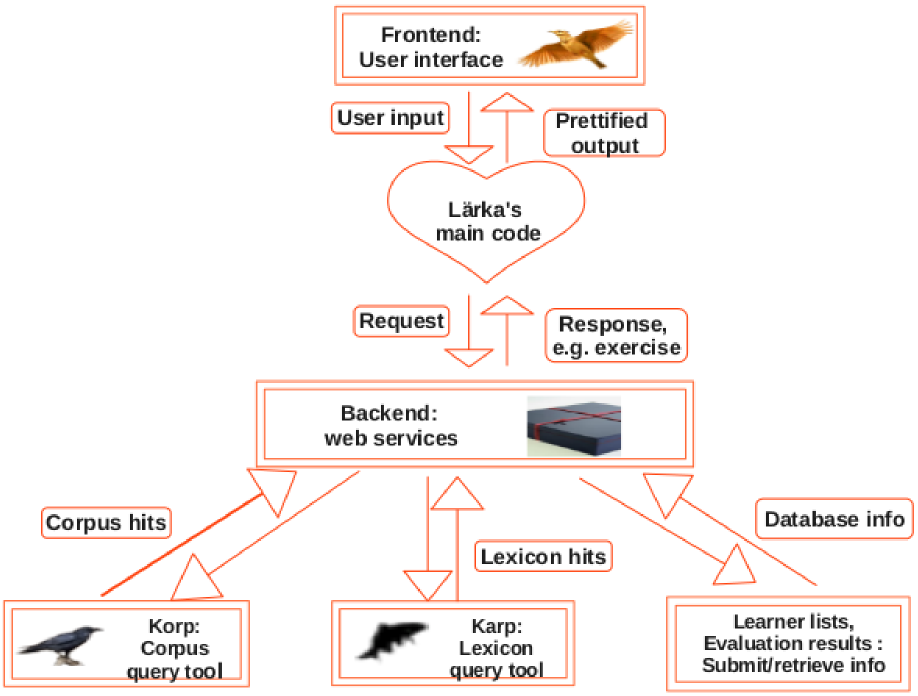


Figure 2. The architecture of Lärka

distractors, generating syntactic trees and rating corpus hits according to their appropriateness for particular exercise types;

- *Korp*[24] is Språkbanken's web-service based infrastructure for maintaining and searching a constantly growing corpus collection, at the moment amounting to over one billion words of Swedish text (Borin et al. 2012a). The corpora available through Korp contain multiple annotations, e.g. lemmatisation, compound analysis, PoS tagging, and syntactic dependency trees, which can form the basis for versatile exercises;

- *Karp*[25] is the corresponding web-service based infrastructure for maintaining and retrieving information from Språkbanken's collection of computational lexical resources (Borin et al. 2012b);

These four components together constitute Lärka's *architecture*. Below, we describe the backend and the frontend, discuss the functionality that Lärka can provide at the moment, and outline future work.

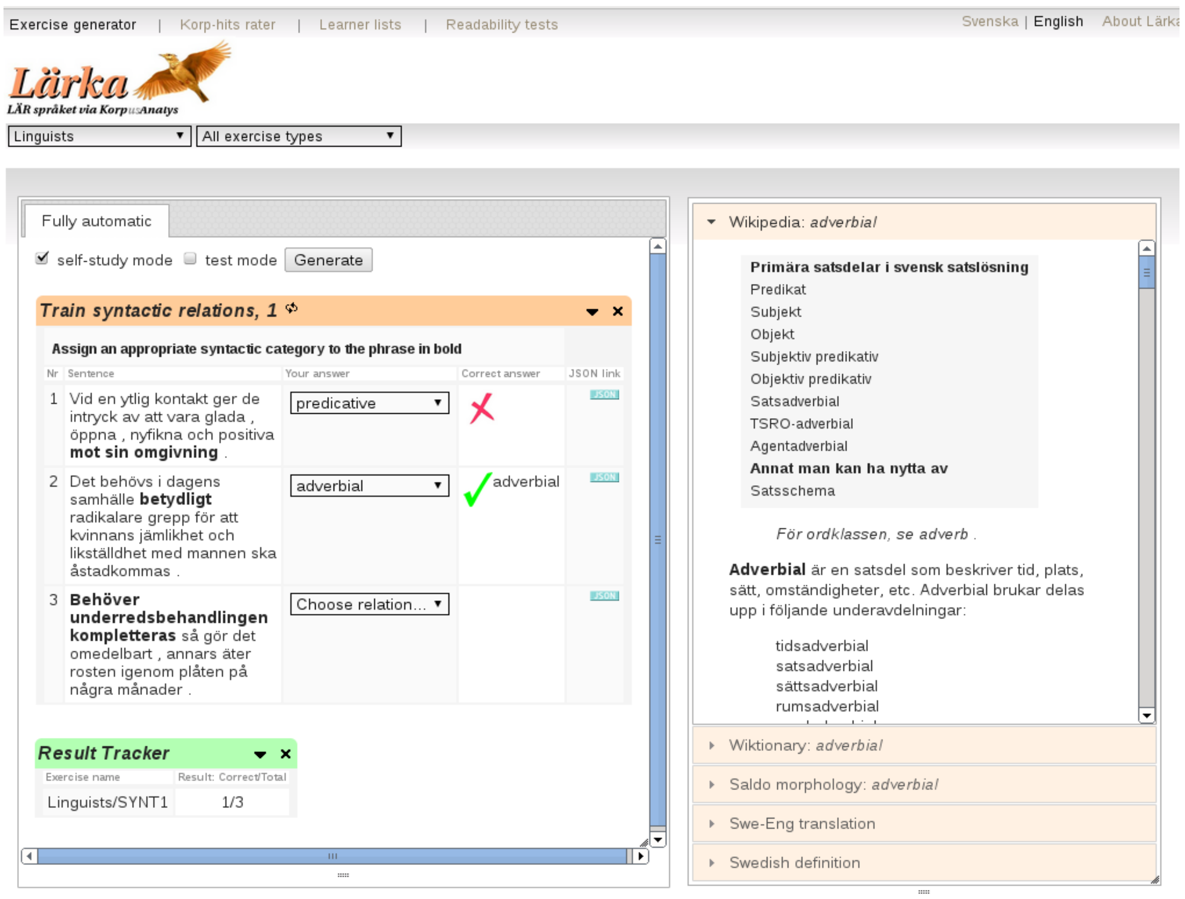
Lärka's frontend (figure 2, top) is the graphical user interface that collects user input and sends requests to the backend. The design has been inherited from the two other applications mentioned above – Korp and Karp. Similarly to these, Lärka will have the functionality to encode the exercise type in a URL (defining the exercise type, training mode, corpus, learner level, etc), so that exercise configurations can be referenced directly as URLs – i.e., bookmarked and passed around – saving users the extra effort of always going through the menus on the main webpage.

Each exercise (or any other future learner activity) is added as a separate module with minimal additions to the user interface code and as a web service. Exercises and other learning objects can thus be developed separately and get integrated with minimal efforts.

At the moment of writing, Lärka offers three exercise types: (1) training PoS; (2) training syntactic relations; and (3) multiple-choice vocabulary exercise items for language learners (re-implemented from Volodina 2010). The first two types are intended for linguistics students and ported from ITG. Each of the exercise types can be run in test mode or in self-study mode, see figure 3. As soon as one item is answered, the next one is generated. The result tracker shows the learner progress.

Lärka's backend is the heart of the architecture; see figure 2. Lärka depends heavily on the corpora and their annotation, and therefore uses Korp's web service for sentence selection. The rich

[24]http://spraakbanken.gu.se/korp/ (*korp* means 'raven').
[25]http://spraakbanken.gu.se/karp/ (*karp* means 'carp' [the fish]).


Exercise generator | Korp-hits rater | Learner lists | Readability tests

Svenska | English About Lärka

Linguists

All exercise types

Fully automatic

☑ self-study mode ☐ test mode Generate

Train syntactic relations, 1

Assign an appropriate syntactic category to the phrase in bold

Nr	Sentence	Your answer	Correct answer	JSON link
1	Vid en ytlig kontakt ger de intryck av att vara glada , öppna , nyfikna och positiva **mot sin omgivning** .	predicative	✗	JSON
2	Det behövs i dagens samhälle **betydligt** radikalare grepp för att kvinnans jämlikhet och likställdhet med mannen ska åstadkommas .	adverbial	✓ adverbial	JSON
3	**Behöver underredsbehandlingen kompletteras** så gör det omedelbart , annars äter rosten igenom plåten på några månader .	Choose relation...		JSON

Result Tracker

Exercise name	Result: Correct/Total
Linguists/SYNT1	1/3

Wikipedia: *adverbial*

Primära satsdelar i svensk satslösning
Predikat
Subjekt
Objekt
Subjektiv predikativ
Objektiv predikativ
Satsadverbial
TSRO-adverbial
Agentadverbial
Annat man kan ha nytta av
Satsschema

För ordklassen, se adverb .

Adverbial är en satsdel som beskriver tid, plats, sätt, omständigheter, etc. Adverbial brukar delas upp i följande underavdelningar:

tidsadverbial
satsadverbial
sättsadverbial
rumsadverbial

Wiktionary: *adverbial*

Saldo morphology: *adverbial*

Swe-Eng translation

Swedish definition

Figure 3. Lärka's frontend (GUI)

annotations available in Korp facilitate generation of exercise types other than the ones that have already been implemented; these are planned for future implementation.

```
{"corpus": "TALBANKEN",
 "distractors":["FV","IO","IV","OO","SP",
                "SS","VA"],
 "distractors_en_sv": {
  "FV":{"en":"finite verb","sv": "finit verb" },
  "IO": {"en":"indirect object","sv":
         "indirekt objekt"},
  "IV": {"en":"nonfinite verb","sv": "infinit
          verb"},
  "OO": {"en":"object","sv": "objekt"},
  "SP": {"en":"predicative","sv":
          "predikativ"},
  "SS": {"en":"subject","sv": "subjekt"},
  "VA": {"en":"adverbial","sv": "adverbial"} },
 "exetype": "synt1",
 "sent_index": 5648,
 "sentence_left": "De hanliga
          vinkarkrabborna bygger ",
 "sentence_right": "i stranden . ",
 "target": "hålor ",
 "target_deprel": "OO",
 "target_index": 4}
```

Figure 4. Example output (in JSON[26] format) from Lärka's backend.

The output from Lärka's backend (i.e. web services) can be used by any program, e.g. in mobile apps. An example of the output from the web service is shown in figure 4. Here you can see all the necessary information for the syntactic training exercise (in JSON, which currently is the common data communication format used by all Språkbanken's web services):

- sentence_left, target and sentence_right make up a complete sentence;
- the target is the part of the sentence that needs to be matched with a syntactic relation;
- the target's syntactic relation (correct answer) is provided as a tag in target_deprel;
- the list of distractors is provided together with the Swedish and English terms for each tag;
- the extra information, like corpus, sent_index (sentence index), target_index (position of the target item in the sentence), etc. are provided in case the user would want to replicate exactly the same item once again through a call to the backend.

In the user interface a JSON link is provided for every single exercise item for those who want to see the web service output.

The web service algorithms for exercise

[26]JSON is an acronym for JavaScript Object Notation.

generation are language independent since they rely on the annotation only. The exercise generation can therefore be made language independent provided there are resources (corpora and underlying word lists) for other languages using the same annotation.

At the moment, the web service output is provided in one format only – JSON. Eventually, other formats will be added, e.g. QTI (Question and Test Interoperability; IMS 2006) and TCF (Hinrichs 2010).

Next on our to-do list is to add syntactic tree visualisations, show relevant encyclopedia entries as an accompanying feature for exercises, design morphological and semantic exercise items based on Karp's web-services (Borin et al. 2012b), add gap cloze and wordbank items as well as diagnostic tests for vocabulary knowledge training. In the more distant future we are planning to:

- add an option of editing existing exercises by providing word lists, texts or selecting other distractors;
- extend the Lärka with Hit-ex – a web service and frontend for showing results from an algorithm for rating corpus searches according to different combinations of linguistic parameters. Tests with Hit-ex are ongoing;
- add the possibility to measure text readability using several readability indices;
- and of course add more exercise types, for grammar, word-building, etc.

5 Concluding remarks

The main idea of our project is to stimulate the re-use of existing accurate NLP tools and resources in language learning by designing and implementing a system architecture for ICALL, at the moment on a more abstract level – where our two subprojects share the general philosophy of making NLP components available via web services – and in the next phase of the project on the concrete level of having a common data exchange format (e.g. TCF). ICALL researchers and developers clearly stand to benefit from our project. In addition, language learners will also be affected because the system architecture and the two test applications will benefit language learners in the form of a more versatile and open-ended CALL experience, thanks to the NLP components.

Our experiences so far indicate that web services are a promising approach to re-use of existing NLP components: they are easy to develop and they preserve their independent stable form despite the changes introduced to the user interface. However, web service providers – including ourselves – should keep in mind, that (1) the services need to be stable and predictable over time, i.e. not undergo sudden changes in their output formats or any other unwelcome changes that can influence the performance of the application(s) based on them; (2) they should deliver as much information as possible to allow the end-user some variation in using their output, e.g. in the case of Lärka's syntactic exercises, the output from the web service could contain not only strings of left and right contexts, but also all associated annotation information for each token coming from the corpus web service.

Practical experience also shows that the web services as far as possible should be split into one separate component that reads information in the request and makes calls to separate request-specific components. In other words, the service-based architecture should be consistently applied all through the application. In the long run, this makes maintenance of the components easier.

It is at the moment undecided which formats we will adopt as standards in the final versions of our web-services. The two formats – TCF and JSON – adopted at the moment by the two language teams work well for us at this testing stage. We should, however, consider the end user interests; for example there is one format we know is used for exercises – QTI (see above) – that we consider important for inclusion as an output format for the exercise generator; there might be other relevant formats that need to be considered.

However, we believe that once our web-service based philosophy is adopted by other owners of NLP components, the two applications described in this paper may become a potential portal for delivering results gained by researchers in CALL, NLP and HCI to the general user and therefore fulfil a very important aim: to make NLP and ICALL research results available outside academia in the form of hands-on applications, thus making technology benefit language learning.

Acknowledgments

The work in this paper has been partially funded with support from NordPlus Sprog, grant LA-2011_1a-25339. The Swedish group has also been partially funded by the University of Gothenburg through its support of the Centre for Language Technology and of Språkbanken.

References

Luiz A. Amaral and Detmar Meurers. 2011. On using intelligent computer-assisted language learning in real-life foreign language teaching and learning. *ReCALL* 23(1): 4–24.

Luiz A. Amaral, Detmar Meurers, and Ramon Ziai. 2011. Analyzing learner language: towards a flexible natural language processing architecture for intelligent language tutors. *Computer Assisted Language Learning* 24(1): 1–16.

Birna Arnbjörnsdóttir. 2008. Kennsla tungumála á netinu: Hugmyndafræði og þróun Icelandic Online [The teaching of languages through the Net: The ideology and development of Icelandic Online]. *Hrafnaþing* 5: 7–31.

Birna Arnbjörnsdóttir. 2004. Teaching morphologically complex languages online: Theoretical questions and practical answers. *CALL for the Nordic languages*, ed. by Peter Juel Henrichsen. (Copenhagen Studies in Language 30.) Copenhagen: Samfundslitteratur.

Kristín Bjarnadóttir. 2005. Modern Icelandic inflections. In H. Holmboe, editor, *Nordisk Sprogteknologi* 2005, 49–50. Museum Tusculanums Forlag, Copenhagen.

Lars Borin. 2002. What have you done for me lately? The fickle alignment of NLP and CALL. In Proceedings of the EUROCALL 2002 pre-conference workshop “NLP in CALL”. Jyväskylä, Finland.

Lars Borin. 2006. Sparv i tranedansen eller fisken i vattnet? *Språkteknologi och språklärande. Från vision till praktik: Språkutbildning och informationsteknik*, ed. by Patrik Svensson. Rapport 1:2006, Nätuniversitetet. 25-49.

Lars Borin and Anju Saxena, A. 2004. Grammar, incorporated. *CALL for the Nordic languages*, ed. by Peter Juel Henrichsen. (Copenhagen Studies in Language 30.) Copenhagen: Samfundslitteratur. 125–145.

Lars Borin, Markus Forsberg, and Johan Roxendal. 2012a. Korp – the corpus infrastructure of Språkbanken. *Proceedings of LREC 2012*. Istanbul: ELRA. 474–478.

Lars Borin, Markus Forsberg, Leif-Jöran Olsson, and Jonatan Uppström. 2012b. The open lexical infrastructure of Språkbanken *Proceedings of LREC 2012*. Istanbul: ELRA. 3598–3602.

Rod Ellis, Helen Basturkmen, and Shawn Loewen. 2002. Doing focus-on-form. *System* 30: 419–432.

Thomas Erl. (2005) *Service-Oriented Architecture: Concepts, Technology, and Design*, Prentice-Hall, USA

Trude Heift. 2003. Multiple learner errors and meaningful feedback: A challenge for ICALL systems. *CALICO Journal*, 20(3), 533–548.

Marie Hinrichs, Thomas Zastrow, and Erhard Hinrichs. 2010. WebLicht: Web-based LRT services in a distributed eScience infrastructure. *Proceedings of LREC 2010*. Valletta, Malta: ELRA.

IMS (2006). IMS Question and Test Interoperability overview. Version 2.1 Public Draft (revision 2) Specification. IMS Global Learning Consortium. http://www.imsglobal.org/question/qtiv2p1pd2/ imsqti_oviewv2p1pd2.html (Retrieved on 29th Juny, 2012).

Sofie Johansson Kokkinakis,. 2009, Readability and multilingualism. *Multilingualism, Proceedings of the 23rd Scandinavian Conference of Linguistics,.* Studia Linguistica Upsaliensia 8 (Acta Universitatis Upsaliensis). 323–324

Ola Knutsson. 2005. Developing and evaluating language tools for writers and learners of Swedish. Doctoral thesis in human-computer interaction. KTH, Stockholm.

Steven Krauwer. 2003. The Basic Language Resource Kit (BLARK) as the first milestone for the language resources roadmap. *Proceedings of SPECOM 2003*. Moscow.

Hrafn Loftsson. 2008. Tagging Icelandic text: A linguistic rule-based approach. *Nordic Journal of Linguistics*, 31(1), 47–72.

Hrafn Loftsson and Eiríkur Rögnvaldsson. 2007a. IceParser: An incremental finite-state parser for Icelandic. In J. Nivre, H-J. Kaalep, K. Muischnek and M. Koit (eds.), *Proceedings of the 16th Nordic Conference of Computational Linguistics (NODALIDA-2007)*. Tartu, Estonia.

Hrafn Loftsson and Eiríkur Rögnvaldsson. 2007b. IceNLP: A natural language processing toolkit for Icelandic. In *Proceedings of InterSpeech 2007, Special session: "Speech and language technology for less-resourced languages*". Antwerp, Belgium.

Ulrika Magnusson and Sofie Johansson Kokkinakis. 2008. Quantitative measures on student texts. *Papers from the ASLA Symposium in Stockholm, 7-8 November, 2008*, Association suédoise de linguistique appliquée (ASLA), Language and Learning (ASLA:s skriftserie nr 22). 43–56.

Teresa Martín-Blas and Ana Serrano-Fernández. 2009. The role of new technologies in the learning process: Moodle as a teaching tool in Physics. *Computers & Education 52* (2009), p.35–44

Noriko Nagata. 2009. Robo-Sensei‘s NLP-based error detection and feed-back generation. *CALICO Journal*, 26(3), 562–579.

Kristina Nilsson and Lars Borin. 2002. Living off the land: The Web as a source of practice texts for *learners* of less prevalent languages. *Proceedings of LREC 2002*. Las Palmas: ELRA. 2002. 411–418.

Jörgen Pind, Friðrik Magnússon, and Stefán Briem. 1991. Íslensk orðtíðnibók [The Icelandic Frequency Dictionary]. The Institute of Lexicography, University of Iceland, Reykjavik.

Eiríkur Rögnvaldsson. 2008. Icelandic language technology ten years later. In *Proceedings of “Collaboration: Interoperability between People in the Creation of Language Resources for Less-resourced Languages”, SALTMIL workshop, LREC 2008*. Marrakech: ELRA.

Anju Saxena and Lars Borin. 2002. Locating and reusing sundry NLP flotsam in an e-learning *application*. *LREC 2002. Workshop Proceedings. Customizing knowledge in NLP applications: strategies, issues, and evaluation.* Las Palmas: ELRA. 45-51.

Latha Srinivasan and Jem Treadwell. 2005. An overview of service-oriented architecture, web services and grid computing. Hewlett-Packard Development Company, V02, 11/2005.

Elena Volodina. 2010. *Corpora in Language Classroom: Reusing Stockholm Umeå Corpus in a vocabulary exercise generator.* Saarbrücken: Lambert Academic Publishing.

Elena Volodina and Lars Borin. 2012. Developing a freely available web-based exercise generator for Swedish. *EuroCALL 2012 Proceedings*, Gothenburg.

Elena Volodina, Hrafn Loftsson, Birna Arnbjörnsdóttir, Lars Borin, and Guðmundur Örn Leifsson. 2012. Towards a system architecture for ICALL. In G. Biswas et al. (eds), *Proceedings of the 20th International Conference on Computers in Education*. Singapore: Asia-Pacific Society for Computers in Education.

Peter Wood. 2008. Developing ICALL tools using GATE. *Computer-Assisted Language Learning*, 21:4, 383-392.