CALL-SLT Lite: A Minimal Framework for Building Interactive Speech-Enabled CALL Applications

Manny Rayner, Claudia Baur, Nikos Tsourakis
University of Geneva, FTI/TIM, 40 bvd du Pont-d’Arve, CH-1211 Geneva 4, Switzerland
{Emmanuel.Rayner,Claudia.Baur,Nikolaos.Tsourakis}@unige.ch

Abstract

We present a framework, CALL-SLT Lite, which can be used by people with only very basic software skills to create interactive multimodal speech-enabled CALL games suitable for beginner/low intermediate child language learners. The games are deployed over the web and can be accessed through a normal browser, and the framework is completely language-independent. As the name suggests, the framework grew out of an earlier platform, CALL-SLT, which enables construction of similar games but uses a more sophisticated architecture. We review the history of the project, describing the type of game we are aiming to build and our reasons for believing that they are useful, and then present CALL-SLT Lite and an initial evaluation comparing the performance of the two versions of the framework. The results suggest that the Lite framework, although much simpler, offers performance at least as good as that of the original system.

Index Terms: CALL, speech recognition, web

1. Introduction

We present a framework, CALL-SLT Lite, which can be used by people with only very basic software skills to create interactive multimodal speech-enabled CALL games suitable for beginner/low intermediate child language learners. The games are deployed over the web and can be accessed through a normal browser, and the framework is completely language-independent. As the name suggests, the framework grew out of an earlier one, CALL-SLT, which enables construction of similar games but uses a more sophisticated architecture.

We start by reviewing the history of the project, in particular describing the type of game we are aiming to build and our reasons for believing that such games are useful; we then present CALL-SLT Lite and the results of an initial evaluation comparing the performance of the two versions of the framework. The final section summarises and suggests further directions.

2. Background: dialogue games and the CALL-SLT system

When speech recognition has been used in CALL, it has most often been in the context of pronunciation practice. The system gives the student a sentence to speak, normally by playing a recorded file, and the student is asked to imitate it to the best of their ability. They are then given feedback about the differences between their pronunciation and the desired one. A typical and well-documented example is EduSpeak[©][1]; there are many others.

Although systems like these are clearly useful, they only address one part of the more general problem of acquiring skills in spoken language. There has been increasing interest in the question of designing CALL systems which use speech recognition to exercise the student's generative competence and listening comprehension. Ideally, a system designed around these goals would be able to conduct a free conversation with the student in a rendered virtual environment; it is, however, evident that this is an extremely challenging goal. A few applications along these general lines have been built, the most successful probably being TLTCS [2], but it is hardly an accident that this was constructed for the US military.

Most potential users cannot even consider investing the resources required to create a system like TLTCS. A more modest architecture builds on the idea of “prompted interaction”; at each turn, the system indicates to the student in some indirect manner how they are supposed to reply, and rates their response. An influential early example of this way of doing this was Wang and Seneff’s “spoken translation game” [3]: the student is prompted with a sentence in the L1 and is supposed to respond by speaking a corresponding sentence in the L2. CALL-SLT [4], which will be our starting point in the current paper, uses a version of this scheme.

As Wang and Seneff point out in their original paper, a spoken CALL system can easily be constructed by reconfiguring components taken from an L2-to-L1 speech translation system, and this plan was also adopted when we built the CALL-SLT system. The central idea is straightforward. A corpus of L2 sentences forms the basic material on which the student will practice, and is first translated offline into the L1. At each turn, the student is
prompted with one of these L1 translations, and asked to speak an L2 sentence which will result in the same L1 when translated. When the student responds, the system uses speech recognition and machine translation to convert what they say into L1 form. The result is compared with the prompt, and the student is given feedback on how close they are.

There are two immediate problems that need to be solved in order to realise the scheme successfully in practice. The first has to do with speech recognition. The students will be non-native speakers who, more or less by definition, can be expected to speak incorrectly. Although the point is to help students improve their command of language by showing them the difference between utterances which are recognised and ones which are rejected, experience shows that many subjects become discouraged if too much of what they say is rejected. This implies that recognition needs to be rather more robust than usual; in general, it suggests the use of limited-domain recognisers, which can be tuned to give high performance on in-coverage material. The second problem concerns translation. If the student responds to the prompt with a close paraphrase of the original L2 utterance, it will be unhelpful for the system to reject them. For example, if the L2 is English and the L1 is German, the original L2 corpus utterance is “I would like a coke”, and the resulting L1 prompt is *Ich möchte eine Cola*, the system should probably accept responses like “Could I have a coke?” or “A coke please”. This suggests that translation should be organised “many-to-one”, so that L2 utterances are as far as possible translated into a canonical form.

In the original CALL-SLT system, these core problems have been addressed entirely within a grammar-based framework. Speech recognition is performed using the Regulus toolkit [5], in which efficient domain-specific grammars are compiled out of general resource grammars using an explanation-based learning method driven by small example corpora. Roughly speaking, the training examples are used to extract pieces of the resource grammar and glue them together into more coarse-grained rules which cover specific constructions. If the technique is used to derive a single language model for the whole application, this has the desirable consequence that grammar coverage is uniform: a construction available in one context is also available in another, even if the contexts are linked to different training examples. So, for example, if an example response to the prompt *Ich möchte eine Cola* is “I would like a coke” and an example response to the prompt *Ich möchte ein Einzelzimmer* is “Could I have a single room?”, then the response “Could I have a coke?” will automatically be valid for the first prompt, and the response “I would like a single room” for the second. The grammar used to build the language model also assigns semantic interpretations to responses: these are then translated into corresponding semantic interpretations for the prompting language, which is used to generate the L1-based surface forms for the prompts. The fact that the prompts are defined through a single grammar means that they are also guaranteed to have a uniform appearance. In practice, the prompting grammar was constructed to produce simplified, “telegraphic” utterances, which for example ignored issues like subject/verb agreement. This meant that it could be kept simple and efficient.

The first version of the CALL-SLT system followed the model of the original Wang and Seneff translation game, and drew practice examples from a single corpus. It soon became clear, however, that students would prefer to have the material organised in a manner similar to that of normal language textbooks, with examples illustrating a common theme grouped together. Once again, the grammar-based framework permits an elegant solution: rules can be written to pick out examples containing given syntactic and semantic structures, for instance requests involving singular noun phrases or WH-questions of a certain type [6].

The system is deployed over the web using a thin client architecture which makes it accessible from both desktop and mobile platforms [7]. This has made it possible to perform substantial evaluations: the largest of these to date is the one described in [8], carried out through the Amazon Mechanical Turk crowdsourcing site and involving 130 subjects. Contrasting a group of subjects which used the normal CALL-SLT system against a control group that used a version with the speech recognition disabled, it was possible to establish that availability of speech recognition significantly improved learning outcomes ($p < 0.02$).

2.1. Problems with the CALL-SLT framework

Initial versions of CALL-SLT were trialled on university-level students; for example, the studies described in [9, 10] involved adult learners of French whose L1s were Arabic and Mandarin respectively. More recently, we have begun to experiment with younger learners, focussing in particular on beginner students in French- and German-speaking Switzerland. These subjects, typically around 12 to 14 years old, exhibit a rather different pattern of behaviour. While older learners were generally happy with the application and said that interacting with it had helped them, the young students expressed impatience and dissatisfaction in post-experiment debriefing [11].

Three points in particular stood out. First, the top-level organisation of the course seemed wrong. The experience of using the original CALL-SLT framework is roughly that of working through exercises of the kind one might find at the end of a chapter in a language textbook. The young students said instead that they would prefer a framework which organized the examples into a narra-
tive structure, so that the underlying metaphor was closer to that of playing a videogame. Second, false negatives were perceived as much worse than false positives: if the student felt they had been incorrectly rejected too many times, they tended to give up. Third, the students were uncomfortable with the form of the prompts. Although the telegraphic L1-based prompting language has the advantages of simplicity and uniformity, they often found it unnatural, and said they would prefer prompts in normal L1.

In order to respond to these demands, we needed to make substantial modifications to the architecture. We changed the overall structure by introducing a scripting language that allowed us to present lessons as interactive dialogues [12]. (Similar schemes are presented, for example, in [13, 14]). With regard to recognition, we constructed multiple smaller language models which only covered a single lesson each (the smaller the coverage of the model, the more forgiving recognition becomes). To make prompts more natural, we added a post-processing step so that they could be arbitrarily modified by surface rewriting rules.

Our intuitive impression is that the reworked system is much more user-friendly. This is reinforced by the ongoing data-collection we are performing in three schools situated in the Basel area [15, 16]. In previous school-based data collection exercises, we struggled to obtain hundreds of interactions; the current one has already logged several thousand, with individual students contributing as many as 800 interactions. It seems clear that at least some of the young users now enjoy working with the system.

The downside of these changes, however, is that the elegance of the original architecture has been lost. Recognition grammar coverage is no longer guaranteed to be uniform, since the model for each lesson is trained on a different set of examples; similarly, the post-processing stage used for the prompt grammar means that prompts are not guaranteed to be uniform either. In general, the tendency was more and more strongly towards an enumerative approach. We consequently decided to create a third version of the framework. This is described in the next section.

3. CALL-SLT Lite

CALL-SLT Lite is a complete redesign of the CALL-SLT framework, where one of the main goals has been to make it possible for people without specialist skills to create interactive speech-enabled courses on their own. We rapidly concluded that only an enumerative approach would be practically compatible with this goal; but, as already pointed out, this is the direction in which the project was moving already. A CALL-SLT Lite course is defined by two files. One of these specifies the narrative flow of the dialogue in terms of abstract classes of prompts and associated responses. The second defines the concrete prompt/response pairs. In both cases, we have resisted the temptation to create a overly powerful and general notation. One source of inspiration has been VoiceXML [17], a successful framework which allows implementers to create moderately complex form-filling spoken dialogue applications with minimal means: from this point of view, we are building a VoiceXML-like platform for creating spoken CALL courses.

As with VoiceXML, the application’s dialogue flow is specified in a simple XML-based notation. The dialogue description consists of a set of named dialogue states (steps). Each step specifies a multimedia file, which constitutes the system’s part of the turn, a pointer to a group of prompt/response pairs (defined in the prompt/response pair file), and a list of links which specify the step to transition to next. In general, these links will be conditional on whether or not the student’s response to the prompt was accepted by the system; there may be other conditions as well.

Figure 1 shows an example of a typical step, taken

```xml
<step>
  <id>
    ask_for_number_nights
  </id>
  <multimedia>
    how_many_nights
  </multimedia>
  <group>
    room_for_number_of_nights
  </group>
  <repeat>
    ask_for_number_nights
  </repeat>
  <limit>
    is_one_night_okay
  </limit>
  <success probability="25"> not_available </success>
  <success>
    ask_type_of_room
  </success>
</step>
```

Figure 1: Example of a dialogue step.

1It is reasonable to ask why we don’t just use VoiceXML. The answer is straightforward: prompted multimedia CALL dialogues have a very different structure from form-filling information-seeking dialogues. Even if it were possible to construct them in VoiceXML by making heavy use of embedded Javascript elements (the usual solution when plain VoiceXML won’t do the job), this would negate the point of the exercise, which is to provide a simple framework for specifying dialogue flow in CALL applications.

2The notation used has been simplified for expositional purposes.
Figure 2: Screenshot of CALL-SLT user interface. The application plays the multimedia file and presents the user with an L1 text prompt indicating how they are supposed to respond. The student can get a correct example (recorded during previous successful interactions with native speakers) by pressing the Help button. Completing the lesson with a sufficiently high score gives credit towards the next badge.

Figure 3: Example of a basic Prompt.

from a lesson in which the student is playing the role of a customer in a hotel. The id, multimedia and group tags respectively specify the step ID, the multimedia prompt file and the prompt group; the multimedia file shows the desk clerk asking how many nights the student wishes to stay, and the prompt groups consists of requests to stay for varying numbers of nights. (Figure 2 shows a screenshot of the interface the student sees). The repeat tag says to repeat the step if the student’s response is not accepted. If it is not accepted three times, the limit tag says to move to the step is_one_night_okay, where the student is asked a simple yes-no question. Conversely, if the response is accepted, the two success tags say to move either to the step not_available (25% probability) or otherwise to the step ask_type_of_room. Simple as the scripting language is, it is possible to use it to craft reasonably interesting dialogues; this is described in more detail in [18].

The prompt/response pair file is also specified using a minimal notation, in which the basic unit is the Prompt;

Figure 4: A more elaborate version of the Prompt from Figure 3, illustrating use of the alternation and optionality constructs.

this could equally well have been written in XML, but for compactness uses a plain text format instead, since no attribute fields are required. Figure 3 shows a basic example of a Prompt, taken from a step in a restaurant dialogue where the student asks for the check. The first three fields give the Domain, Lesson and Group: the last of these is the group used in the XML step units. The Text line is the text of the prompt (this example is taken from an English-L2 course intended to be used by German-L1 speakers).

The Response lines specify the possible responses accepted from the student: these are used to create the language models which drive the course-specific speech recogniser. At the moment, these consists of a set of flat grammars, one per lesson, each of which lists the union
of the possible responses available in the prompt units for the lesson in question. It would be equally easy to create a more fine-grained language model which used one grammar per prompt, or a more coarse-grained one with a single grammar for the course.

A completely enumerative framework is arguably appropriate for the most naive course designers, but, after some experimentation, we decided that it was reasonable to introduce some basic notions of grammar in order to be able to list responses more compactly. We have limited ourselves to four constructs: alternation, optionality, a template/macro mechanism, and explicit definition of incorrect responses. Figure 4 shows a typical example of the first two constructs, where as usual constituents separated by a vertical bar mark alternation and optional constituents are marked with a question mark, and Figure 5 an example of the template/macro mechanism. Finally, Figure 6 (this time, taken from a French-L2/Italian-L1 course) illustrates the notation for defining incorrect responses. In all cases, the set of alternatives is expanded out into a flat grammar at compile time.

4. A comparison of the two frameworks

As can be seen, CALL-SLT Lite is almost embarrassingly simple. The obvious question is whether such a trivial framework can actually do the job; in this section, we present an initial comparison with the more complex grammar-based framework.

We took our two most substantial CALL-SLT courses and converted them into Lite format. The one on which we have focussed, due to easy availability of recorded data for offline tests, is the English-for-German-speakers course from [18] already referred to above; we also converted the French-for-Italian-speakers course from [19], with results that were anecdotally similar. The English-for-German-speakers course consists of eight lessons, containing a total of 145 dialogue steps; the annotated corpus file used to train the original grammar-based language models contained 1349 examples, giving a combined vocabulary of about 450 words. We wrote a simple script which converted this corpus, together with associated information about prompts, into a Lite prompt/response file in the format shown in the preceding section. The resulting file contained one response for each example in the original corpus. One of us then manually edited the file to add alternations, optional elements and templates, as illustrated in Figures 4 and 5, a process which took about 10 hours. The final version of the file contains 55 templates and 493 template applications, creating a total of 11102 responses.

Both the original and the Lite versions of the grammar were compiled into language models and then into recognition packages using the commercial Nuance Toolkit. A separate package is compiled for each lesson.

We performed the evaluation on the corpus described in [15]; this contains 862 recorded utterances taken from young Germanophone students, annotated to specify adequacy or otherwise of vocabulary, grammar, pronunciation and recording quality. As usual in a spoken CALL application, we are interested in the contrast between recognition performance on correct and incorrect responses, where ideally we want correct utterances to be accepted and incorrect ones rejected. Table 1 presents the figures for the two versions of the course. It is reasonable that a fair number of “incorrect” utterances are accepted; grammar mistakes are often no more than an incorrect article, and pronunciation errors may only affect one vowel in the whole sentence.
Table 1: Proportion of spoken utterances rejected by standard and Lite versions of the CALL-SLT English-for-German-speakers course, for various conditions. “Language OK” = grammar and vocabulary marked as correct; “Pronunciation OK” = pronunciation marked as correct; “Sound OK” = sound quality marked as correct (no loud background noise, no cutting, normal recording volume).

<table>
<thead>
<tr>
<th>Condition</th>
<th>#Utts</th>
<th>Rejected</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Standard</td>
</tr>
<tr>
<td>Language OK</td>
<td>712</td>
<td>23.5%</td>
</tr>
<tr>
<td>Language not OK</td>
<td>150</td>
<td>60.0%</td>
</tr>
<tr>
<td>Pronunciation OK</td>
<td>561</td>
<td>22.6%</td>
</tr>
<tr>
<td>Pronunciation not OK</td>
<td>301</td>
<td>43.2%</td>
</tr>
<tr>
<td>Sound OK</td>
<td>679</td>
<td>21.1%</td>
</tr>
<tr>
<td>Sound not OK</td>
<td>183</td>
<td>62.3%</td>
</tr>
<tr>
<td>Language OK + sound OK</td>
<td>593</td>
<td>21.1%</td>
</tr>
<tr>
<td>Language not OK + sound OK</td>
<td>86</td>
<td>44.2%</td>
</tr>
<tr>
<td>Pronunciation OK + sound OK</td>
<td>463</td>
<td>16.2%</td>
</tr>
<tr>
<td>Pronunciation not OK + sound OK</td>
<td>216</td>
<td>31.5%</td>
</tr>
<tr>
<td>Language OK + pronunciation OK + sound OK</td>
<td>440</td>
<td>14.8%</td>
</tr>
</tbody>
</table>

There is no intuitively obvious criterion, based on these figures, that can be used to decide which version of the system is better, and the most reasonable argument we can offer is the following. Both versions reject about half of the utterances which contain grammar or vocabulary errors (60% for the standard version; 43% for the Lite version), and are about three times less likely to reject if the linguistic content of the utterance is correct (24% for standard, 13% for Lite). Students thus get reasonably informative feedback on correctness of language. For incorrectly pronounced utterances, the standard version rejects 43% and the Lite version 28%, but the difference compared to correctly pronounced is only about a factor of two (23% standard, 13% Lite). The feedback is thus considerably less effective with regard to pronunciation.

Our overall feeling, based on earlier studies, is that the system’s main strength is in helping students build up conversational competence; as previously noted, students are also more distressed by false negatives than false positives. With thoughts like these in mind, the Lite version would appear to be at least as good as the standard one. Initial anecdotal testing reinforces this impression.

5. Summary and further directions

We have described CALL-SLT Lite, a VoiceXML-like framework which supports rapid construction of interactive speech-enabled dialogue games. Our initial study suggests that recognition performance of CALL-SLT Lite applications is not obviously inferior to that of the more sophisticated grammar-based architecture. The important difference between the two version, it seems to us, is that CALL-SLT Lite courses can be built with little or no input from technical experts, which contrasts extremely favorably with the grammar-based architecture. It is in particular straightforward to port courses to a new L1, since this only involves translation of fixed prompts. We tested this by porting the English-for-German-speakers course to make it suitable for French students, a process which only took a few hours.

Another advantage of the minimal CALL-SLT Lite framework is that recognition grammars are not only easy to write, but also easy to modify. When debriefing students who have used the system, we have often received complaints that some correct responses had been rejected by the system due to missing grammar coverage. We plan in the near future to enhance the interface so that students, in this type of situation, can enter their own suggestions for additional responses to be covered. These suggestions can be stored and reviewed later by the course designer, who can if she wishes include them in the grammar by doing no more than ticking a dialogue box. The idea can be thought of as crowdsourcing the task of debugging and extending the response grammar. It also seems feasible, though slightly more complex, to allow corresponding crowdsourcing of the dialogue flow.

In general, opening up the toolbox to allow external users to become involved in the course construction process creates many interesting possibilities, and points naturally towards the development of a social network which links together course constructors and course users. The current phase of the CALL-SLT project has just started, and will run until mid-2017; we expect the ideas sketched above to form some of the major themes of research during this period.

---

3This is normally expressed as “The system doesn’t know the following word/expression”.
6. References


