Crafting Interesting Dialogues in an Interactive Spoken CALL System

BAUR, Claudia, RAYNER, Emmanuel, TSOURAKIS, Nikolaos

Abstract
We describe work carried out in the context of a project whose goal is to develop a web-enabled serious game, designed to help German-speaking beginner students of English improve their generative and auditory competence. The game, which is currently undergoing a first round of formal user testing, is intended as a self-study adjunct to normal classroom instruction. It was developed in collaboration with a secondary school teacher, with the content taken from a textbook commonly used in German-speaking Switzerland, and offers a short course of 8 interactive lessons using a combined vocabulary of about 450 words. A lesson is structured as a short prompted dialogue between the student and the machine, where the student is encouraged to use simple language in practical contexts like booking a hotel room, buying clothes, or ordering a meal in a restaurant. We present an overview of the architecture, which is designed to balance three main considerations: it should support coherent grammatical responses and reasonably interesting dialogues, but also allow course content to be constructed by personnel who do not necessarily [...]
CRAFTING INTERESTING DIALOGUES IN AN INTERACTIVE SPOKEN CALL GAME

Claudia Baur¹, Manny Rayner¹, Nikos Tsourakis¹
¹Université de Genève (SWITZERLAND)

Abstract
We describe work carried out in the context of a project whose goal is to develop a web-enabled serious game, designed to help German-speaking beginner students of English improve their generative and auditory competence. The game, which is currently undergoing a first round of formal user testing, is intended as a self-study adjunct to normal classroom instruction. It was developed in collaboration with a secondary school teacher, with the content taken from a textbook commonly used in German-speaking Switzerland, and offers a short course of 8 interactive lessons using a combined vocabulary of about 450 words.

A lesson is structured as a short prompted dialogue between the student and the machine, where the student is encouraged to use simple language in practical contexts like booking a hotel room, buying clothes, or ordering a meal in a restaurant. We present an overview of the architecture, which is designed to balance three main considerations: it should support coherent grammatical responses and reasonably interesting dialogues, but also allow course content to be constructed by personnel who do not necessarily have a computer science background. We use as an example a new strategy we have recently developed, where the central idea is to add further dialogue paths to the scripts, graded by their degree of “cooperativeness”. The system starts with a simple cooperative interaction, and opens up increasingly “uncooperative” dialogue paths as the student gains familiarity with the game.

Keywords: CALL, SLA, translation game.

1 INTRODUCTION
Beginning in 2009, we have been involved in building an online platform, CALL-SLT [1], designed to help students practice speaking in a foreign language. The basic idea is to prompt the student in a way which constrains the meaning of what they are supposed to say, but leaves open the way they are going to say it. In contrast to mainstream tools like RosettaStone or Duolingo, this allows the student to practice receptive and productive language skills, and not merely pronunciation.

When creating a tool of this kind, there are several challenges to address. First, speech recognition must work in a consistent way, so that, if the student uses a particular phrase or grammatical response in response to one prompt, they can also feel confident that they will be able to use it in response to a different prompt. Second, examples should be presented in a thematic way; for example, they can be chosen to illustrate a particular grammatical structure, or form part of a coherent dialogue. Third, it should be easy for people with a non-technical background to create new content. These goals conflict to some extent, and the challenge is to find a good trade-off point.

In this paper, we present a case study based on the version of CALL-SLT we have developed for use by Swiss-German beginner students of English. For this L1-L2 pair, the main goal students face is to build up conversational competence. Content is thus organized as short interactive dialogues, each based on a functional theme like “reserving a hotel room” or “buying a train ticket” [4]. We start by presenting an overview of the architecture, and then describe how we were able to use it to craft moderately complex dialogue flows based on the idea of “uncooperative dialogues”. The intuition is that people naturally find cooperation easier than conflict; it is easier to say yes than no, easier to agree than disagree, and easier to praise than complain. Dialogues are thus organized so that students start with a scenario which is entirely positive. When they have successfully completed the dialogue a few times, new paths are opened up which involve uncooperative interactions; a waiter brings the wrong food at a restaurant, a shop assistant tries to sell the student an article they don’t want, and so on.

By integrating uncooperative dialogue flows, the students have the opportunity to become more self-confident even in challenging situations, where they have to disagree or negate with their conversation partner, when reaching higher levels in the game. Another reason why we decided to integrate
The rest of the paper is organised as follows. Section 2 describes the CALL-SLT language game, Section 3 gives a brief summary of the technical aspects and the dialogue framework of the system, and Section 4 describes the newly added dialogue content covering uncooperative sub-dialogues. Section 5 summarizes and concludes.

2 BACKGROUND

The idea behind CALL-SLT is a very natural one: a speech translation system can be adapted for use as a CALL tool. To the best of our knowledge, this was first realised in concrete form by Wang and Seneff (2007). The basic scenario is the following. We have a speech translation system which translates from the L2 (the language being learned) to the L1 (the student’s native language). We collect a set of L2 examples, which constitute the material on which the student will practice. At each turn, the system chooses an example, processes it to effect translation from L2 to L1, and shows the student the L1 result. The student then tries to say something in the L2 which will result in the same L1 sentence as the one they have just seen. The system processes their response in the same way, matches the results, and tells the student whether or not they have been successful.

There are evidently some problems to solve: the most pressing one is that the system should not mark the student’s response as incorrect if they have uttered a paraphrase of the original utterance. For example, if the L1 is German and the L2 is English, the L2 example “I’d like a beer” might result in the L1 prompt *Ich möchte ein Bier*. If the student now responds with “Could I have a beer?”, it will probably be regarded as unhelpful for the system to respond by marking it as an error. These issues can, however, be addressed by using a translation scheme which maps semantically equivalent L2 phrases into the same L1 phrase. An architecture of this basic type was used in the original Wang and Seneff CALL application, and was also the basis for the first version of CALL-SLT [1].

User testing with the first version soon convinced us that students found an unstructured set of prompts less than optimally motivating: they wanted examples linked by some kind of unifying theme, so that they could practice specific language skills. Ordinary language textbooks suggest two obvious ways to perform this grouping, both of which we have explored. One is linguistic: a unit consists of a group of sentences linked by a common phoneme, word or grammatical construction that the student is intended to practice. For example, in [2], Chinese students of French were given groups of examples which allowed them to practice question and nominal constructions; in [3], the groups were arranged around sounds which French students of English find hard to pronounce.

Some students respond well to this type of organization; they practiced diligently, and tests administered before and after the experiment suggested that they had made concrete progress by studying with the CALL tool. Post-experiment interviews, however, suggested that other students were less enthusiastic. When we began working with German 12 and 13 year olds who were learning English, we decided that a linguistically oriented approach was inappropriate for them. Chinese speakers who are learning French definitely benefit from practicing the complex and entirely alien French structures (Chinese lacks number, gender, tense and WH-movement, all of which are central to French). In contrast, Germanophone students do not find English grammar challenging, since it is close to being a simplified form of German. What they said they needed was conversation practice. Several also suggested that they would prefer a more interactive framework which contained a multimedia component.

These ideas formed the basis for a new version of the system, organized around functional competences rather than grammar. Each lesson in this version is a short interactive dialogue, arranged around a theme like “checking into a hotel” or “visiting a restaurant”. The initial content was developed together with an English language teacher working in German-speaking Switzerland, and is based on the textbook most often used there by secondary school students. The total vocabulary supported is around 450 words.

The new version of CALL-SLT is described in [4]. Summarizing briefly, the system is deployed over the web and can be accessed through a normal browser; Fig. 1 illustrates the interface. Each step in
the conversation starts with the system playing a short recorded English-language video, which in the figure comes from an early step in the Hotel dialogue.

Here, the cartoon desk clerk has asked how many nights the student wants to stay. The German text immediately beneath, *Frag: Zimmer für 6 Nächte* ("Request: room for 6 nights") tells the student how they are supposed to respond. In this example, correct replies include "I would like to stay for six nights", "I want a room for six nights", "A room for six nights, please", and other variants. If the student does not know how to reply, they have the option of pressing the Help button (the question-mark icon, lower right), which gives them a valid response in both written and spoken form. Spoken help responses are automatically recorded from successful interactions with users registered as native L2 speakers.

When the student has decided what they are going to say, they press the microphone icon (middle right) and speak while holding it down. The system uses speech recognition and machine translation to process their response, and either accepts or rejects. The green border around the prompt in the figure shows that the student's response has just been accepted. After the student has replied, the dialogue transitions to a new state. In this example, an accept will move to a state where the desk clerk continues by asking what kind of room the student wants, and a reject to a state where the desk clerk apologizes for not understanding what the student has said and repeats his question. If the student is rejected twice, the dialogue moves to a state where the clerk says he will assume that the student only wants a room for one night. In general, each state contains multiple possible prompts, and many states have multiple possible continuations. When several possibilities are available, the system chooses one at random.

The system is gamified [5], using a standard score-and-badge framework. Students collect four levels of badges (plain, bronze, silver, gold) for each lesson by completing it under increasingly strict conditions. At the lower levels, all they need to do is reach the end; as they progress to the higher ones, they must also achieve a minimum score. The student loses points for responses which are rejected and optionally gains bonuses for phrases which the course designer wants to encourage.

In the next section, we discuss the central issues involved in creating the dialogue framework.

### 3 THE DIALOGUE FRAMEWORK

When designing the dialogue framework, we have had two main design goals in mind. The first is to give the student a varied, interesting dialogue experience that will allow them to practice English conversation in a natural way. The second is to offer a simple process for creating new course content, so that designers without a software engineering background can do as much as possible of the work by themselves. These two goals are in tension with each other. In this section, we describe the solution we have evolved to address them.

An interactive dialogue of the kind considered here requires specification of several different kinds of information. Some of these are unchallenging; for example, the video segments used as multimedia prompts can be constructed using easily available freeware tools. There are, however, two aspects which pose nontrivial problems. Looking first at individual dialogue steps, there is the question, already mentioned above, of supporting flexible responses to the prompt. If the range of permitted responses is too narrow, the student is essentially reduced to querying the online help to find out what they are supposed to say and then repeating it. This is unlikely to give the experience of participating in a real dialogue; instead, speech recognition needs to be able to capture as many as possible of the plausible
ways to respond and reject responses which are too far from intuitive acceptability. A good solution to this first problem allows the student to move beyond simple repetition and begin practicing true generative language skills. The second problem is how to connect the individual steps into an interesting dialogue. If this dialogue is too heavily scripted, the student will become bored and lose interest; at the other extreme, a genuinely free-form dialogue is obviously very difficult to design and maintain. In both cases, some compromise is needed.

We examine these two problems in turn.

3.1 Flexible responses

It is overambitious to try to craft a speech understanding module which will accept any intuitively reasonable response to a given prompt, and simultaneously reject all unreasonable ones. A realistic strategy, given that the system is based on a school textbook, is to aim to accept the common constructions which the student has been taught in class. For second year students like the ones we are using in our study, this appears quite feasible. The critical thing is that coverage should be consistent. For example, if the student has learned that an acceptable response to one request-type prompt is a sentence of the form “Could I have X?”, then responses of the same form should also be accepted in other steps which give request-type prompts.

Given that our coverage goals are formulated in terms of grammatical constructions, we have implemented a solution which uses a grammar-based approach to recognition. The Regulus platform [6] supports example-based construction of efficient grammar-based recognizers, using a method in which the training examples create specialized grammar rules by composing existing rules taken from a resource grammar. What this means in practice is that a construction which is used in an example response to one prompt becomes available as a response to all similar prompts. Thus, for example, if the training corpus gives the response “I would like a room for six nights” to the prompt Frag : Zimmer für 6 Nächte and the response “Could I have a single room?” to the prompt Frag : Einzelzimmer, then rule generalization means that the system will also accept “Could I have a room for six nights?” as a response to the first prompt and “I would like a double room” as a response to the second.

Unfortunately, this power does not come free. The downside is that the course designer is obliged to add lexical entries for all the new vocabulary items they intend to use, and also define translation rules which map the semantic representations of responses into semantic representations of prompts; both of these tasks require linguistic expertise, and there is a non-trivial learning curve involved. We have consequently begun experimenting with a simpler scheme, which does not have such attractive formal properties but can be used much more easily by non-experts. We return to this point in the final section.

3.2 Interesting dialogues

The central material the student is learning – the grammar and vocabulary – is in the individual steps. The dialogue linking the steps together serves two purposes. It sets up plausible contexts for the exchanges in the steps, and, by connecting them into a single conversation, tries to make them more interesting by adding some narrative structure. Some initial experimentation soon convinced us that we needed to be extremely careful in limiting the expressive power of the dialogue framework; it easily turns into a general programming language, in which dialogues become pieces of software that can only be debugged by people with appropriate skills. This is inconsistent with our goal of making the system accessible to designers from a pedagogical background.

The current dialogue framework is thus designed to be as simple as possible, and imposes the uniform structure illustrated by the example in section 2. At each step, the system plays a multimedia prompt and chooses one of a set of text prompts. The prompt is selected randomly, except that the course designer has the option of associating prompts with key/value pairs, which must be kept consistent over the dialogue; this will be explained shortly. The student responds, and the system decides whether to accept or reject. If the student’s response is rejected, the system either moves to a new step immediately or repeats the prompt up to a specified maximum number of times, after which, if the student has still not been accepted, it moves to a new step. (The intention is to make sure that the student does not get stuck on a difficult step). If the student’s response is accepted, the system moves to one of a specified number of new steps. Conditions determine which new step is chosen; the critical design decision is what form these conditions can take.

Concretely, each step is an XML structure which specifies the following pieces of information:
1. The step ID
2. The group of prompts associated with the step.
3. The multimedia files associated with the step.
4. The step to move to if the response is rejected.
5. The step to move to if the maximum number of rejections has been reached.
6. One or more steps to move to if the response is accepted, possibly together with associated conditions. Conditions may be any of the following: a random probability, a restriction on the current badge level, a key/value pair, or a Boolean combination of these.

The following example from the Hotel script illustrates:

```
<!-- Ask for number of nights -->
<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>
```

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. 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.

Despite the extreme simplicity of the framework, we have found that it is in fact possible to craft reasonably interesting dialogue flows. The least obvious aspect is the use of key/value pairs. It will often be desirable for one step to make reference to an object or condition that entered the dialogue in a previous step. For example, in the Shopping lesson, the student is first asked to order a given item of clothing (shirt, pair of trousers, etc), and in a later step may be asked to change their mind and say they do not want it; if they have requested a shirt in the first step, the prompt in the second step also needs to be one that refers to a shirt. This is enforced by marking both prompts as garment=shirt.

In the next section, we will describe more elaborate examples, which in particular illustrate the principle of “uncooperative subdialogues” mentioned in Section 1.

### 4 Dialogue Content

As mentioned in section 2, the dialogue content has been developed in close collaboration with an English teacher at a secondary school. The lessons are based on the commonly used English textbook “Ready For English 1”; in order to guarantee that they are relevant and easy to be used in the Swiss curriculum. In a first phase eight basic lessons have been developed to imitate a virtual trip to London. The lessons included the following scenarios and contents: (1) Train station (course content = name, nationality, numbers, locations, time expressions), (2) Contact – Getting to know someone (course content = name, nationality, siblings, capitals), (3) Tube station (course content = numbers, locations, prices), (4) Hotel (course content = numbers, room types, prices, payment types, where-questions), (5) Tourist Information Office (course content = numbers, cultural knowledge of London, time expressions, ordinal numbers), (6) Restaurant (course content = food and beverages, payment types), (7) Asking and giving directions (course content = where-questions, directions, distances, cultural knowledge of London), (8) Shopping (course content = clothing, colours, numbers, like/dislike

---

1 The notation used for the XML fragments presented here has been slightly simplified for presentational purposes.
expressions). When conducting a first pilot study with more advanced students of English, one of the most important weaknesses of the tool proved to be the rather linear composition of the lessons. The third year English students used for that study were often found to answer in a bored and unmotivated way after playing through the lessons a few times, demonstrating that the consistent level of difficulty was not challenging enough.

The central idea of the current improved version of CALL-SLT was therefore to include a series of uncooperative dialogue steps. As mentioned above one of the main reasons for this implementation was to increase the lesson’s difficulty with the advancement in the game and to further challenge students at higher badge levels. However, another very important factor is to allow the language learners to become familiar with more challenging situations that one might encounter when communicating in a foreign language and to build up some self-confidence when it comes to scenarios in which they need to complain, insist or rectify something.

There are naturally various scenarios in which a non-native speaker feels insecure to react with limited language skills. We integrated some commonly occurring situations as follows:

1. When buying a ticket at the train station, the ticket vendor might offer you a more expensive ticket for a faster or more comfortable train. In this situation it's certainly good to know how to insist on buying the cheaper ticket.

2. When meeting someone and getting into small talk with a foreigner, there is often a misunderstanding as to where one comes from. In our example the conversation partner thinks that the country of origin (only European countries in the current coverage) is in Africa. In this case the student needs to correct the other and explain that the country is in Europe.

3. Another common problem when buying a ticket e.g. at the tube station, is that the ticket vendor won’t accept large bills. Then the student needs to try and find another payment method or ask if there is an ATM where he can get change.

4. When taking a room at a hotel it is very important that one can complain if the room is dirty or if there is anything missing in the room, such as a hairdryer. In this lesson the students learn to complain, to accept/reject the hotel clerk’s offer of re-cleaning the room and to ask where a certain article can be bought if it isn’t available at the hotel.

5. At the tourist information office one might be confronted by a ticket seller who advises you to go see another musical because the one you chose got bad reviews. Here again the student needs to be able to accept the alternative musical or stick with his first choice.

6. The probably most common scenario is a sales assistant who tries to talk you into buying an ill-fitting piece of clothing. In this sub-dialogue the student learns to explain that the clothing doesn’t fit and that he doesn’t want to buy it.

7. When asking for directions one might risk being drawn into an endless conversation. In this case it’s handy to know how to politely exit the conversation.

8. Similar to the shopping scenario, the waiter might bring you the wrong food at a restaurant. In our sub-dialogue the student needs to complain and ask to speak to the manager. With the manager he then needs to negotiate a deal.

Table 1 gives an overview of the eight lessons with their standard cooperative content and the newly added uncooperative sub-dialogues for higher levels of expertise.

Table 1: Lesson Content

<table>
<thead>
<tr>
<th>Lesson</th>
<th>Cooperative content</th>
<th>Uncooperative content</th>
<th>Student’s response</th>
</tr>
</thead>
<tbody>
<tr>
<td>Train station</td>
<td>Buying a ticket to London.</td>
<td>Ticket agent tries to sell a more expensive ticket.</td>
<td>- Insist on buying the cheap ticket&lt;br&gt;- Agree to buy expensive ticket</td>
</tr>
<tr>
<td>Contact</td>
<td>Small talk on the train.</td>
<td>Conversation partner misplaces country of origin.</td>
<td>- Correct conversation partner</td>
</tr>
<tr>
<td>Tube station</td>
<td>Buying a tube ticket.</td>
<td>Ticket agent won’t accept large bill.</td>
<td>- Explain that he doesn’t have smaller notes&lt;br&gt;- Find alternative payment method</td>
</tr>
<tr>
<td>Location</td>
<td>Activity</td>
<td>Problem</td>
<td>Possible Actions</td>
</tr>
<tr>
<td>-------------------</td>
<td>---------------------------------------</td>
<td>---------------------------------------------------</td>
<td>-------------------------------------------------------</td>
</tr>
<tr>
<td>Hotel</td>
<td>Getting a room at a hotel.</td>
<td>Something is wrong with the room.</td>
<td>- Ask for ATM</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- Complain about dirty room</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- Complain about missing article in room</td>
</tr>
<tr>
<td>Tourist information</td>
<td>Buying musical tickets.</td>
<td>Ticket agent warns that the chosen musical is terrible.</td>
<td>- Insist on first choice of musical</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- Ask for other recommendation</td>
</tr>
<tr>
<td>Shopping</td>
<td>Buying clothes.</td>
<td>Agent tries to sell ill-fitting item.</td>
<td>- Disagree and explain that item doesn't fit</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- Reject to buy item</td>
</tr>
<tr>
<td>Directions</td>
<td>Asking and giving directions.</td>
<td>Tourist keeps asking questions.</td>
<td>- Politely get rid of tourist</td>
</tr>
<tr>
<td>Restaurant</td>
<td>Ordering food and beverages.</td>
<td>Waiter brings wrong food.</td>
<td>- Complain about wrong food</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- Ask to speak to manager</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- React to discount offering</td>
</tr>
</tbody>
</table>

Conditions on the success tags (see section 3) are used to make sure that uncooperative dialogue paths are only accessible to students who have already proved that they can master the lower badge levels with the more basic content. The students may enter the additional and more demanding steps only after reaching the higher badge level.

As described in detail in section 3, the dialogue steps are written in an XML-based script. Besides the ID, group and multimedia information, there are several indications on which step to choose next, depending on various conditions. As mentioned above, the repeat tag indicates that the same step is to be repeated if the student fails to get recognised, and limit indicates the step to go to if the student wasn't recognised twice in a row. In this case the student moves on to an easier yes/no question to keep the dialogue flow going. The success tag indicates multiple alternative steps to go to if the recognition result was successful. The example shown below is an extract from the "Tourist Information Office" lesson and illustrates how the dialogue can be developed in various directions depending on the badge level. The choice of step is handled by inserting a condition that needs to be fulfilled in order for the step to be chosen. In the example shown below, the student might move on to a step in which he's being asked where he would like to sit (which_seat); if the student is playing on a low badge level there is also a chance that the next step will inform them that the musical is not available and he'll be asked if another musical would be okay (not_available_other_musical_okay). On a higher badge level, there is then also a 50% chance that the student will be redirected towards an uncooperative step in which the agent tells him that the chosen musical got bad reviews and that he’d suggest an alternative musical (bad_reviews). The level condition is defined as badge-level=bronze.

<!-- Which musical would you like to see? -->

```xml
<step>
  <id>which_musical</id>
  <multimedia>which_musical</multimedia>
  <group>musical_name</group>
  <limit>dont_understand_cats_okay</limit>
  <repeat>which_musical</repeat>
  <success probability="50">which_seat</success>
  <success badge-level="bronze">bad_reviews</success>
  <success>not_available_other_musical_okay</success>
</step>
```

5 SUMMARY AND CONCLUSIONS

We have described an approach to crafting interactive script-based spoken dialogue games in a web-enabled CALL platform. The games are intended for first-year Germanophone students of English,
and focus on practicing conversational functionalities. The key goals are consistent grammar coverage, engaging and varied dialogue, and the possibility for non-experts to be able to develop course content autonomously.

We think it is reasonable to claim that the framework we have developed responds well to the first two goals. The grammar-based approach to speech recognition creates efficient, tightly focused specialized recognizers, but automatically ensures that grammatical constructions licensed when responding to one prompt are also simultaneously licensed for all other prompts. The scripting mechanism, although simple, makes it possible to craft dialogue flows that allow a fair range of variation. We have in particular illustrated using the idea of “uncooperative subdialogues” which are only activated after the student has gained some experience using the system.

The place where we feel least satisfied with our progress is the third goal. Although the scripting language makes few demands on the course designer and can readily be mastered by anyone who is familiar with XML notation, the grammar-based language modelling tool is hard to learn and assumes considerable knowledge of linguistics and software engineering. We have consequently begun experimenting with a less sophisticated alternative, in which response grammars are specified in an enumerative way which involves only very basic use of grammatical formalism. The goal of uniform coverage is sacrificed to some extent, with the course designer being given most of the responsibility for enforcing uniformity through coherent use of a template mechanism. The upside is that the simple nature of the framework makes it far easier to adjust coverage inconsistencies, so it is in practice feasible to hope that grammar coverage can quickly be iterated into an acceptable form. We are still in the early stages of testing this idea, and will describe it in detail elsewhere.

REFERENCES


