Development and evaluation of a software for second language learning: CALL-SLT

BAUR, Claudia

Abstract

Computer Assisted Language Learning (CALL) is becoming increasingly in today's world, where time is a rare good and the command of foreign languages is indispensable. CALL applications allow of efficient and timesaving second language learning. This thesis takes a closer look at the history of CALL during the last fifty years, including Wang and Seneff's Translation Game. The focus of this paper lies on the CALL-SLT Project, which was developed at the University of Geneva. This project is based on two main technologies, namely machine translation and speech recognition, using the Regulus platform and aims at second language acquisition in the restaurant domain. In a more practical section the development of a version of the CALL-SLt software is described that allows learning German as an L2. A concluding evaluation of the developed software gives an idea of its strengths and weaknesses.

Reference


Available at:
http://archive-ouverte.unige.ch/unige:15822

Disclaimer: layout of this document may differ from the published version.
MASTER’S THESIS

DEVELOPMENT AND EVALUATION OF A SOFTWARE FOR SECOND LANGUAGE LEARNING: CALL-SLT

ETI, Université de Genève
Thesis Director: Prof. Pierrette Bouillon
Thesis Jury: Johanna Gerlach

by
Claudia Baur

September 2010
Acknowledgements

I would like to thank Prof. Pierrette Bouillon who helped and supported me throughout the process of writing this thesis. A big thanks also goes to Johanna Gerlach, the jury of this thesis, who was a very important source of advice and help. I would also like to thank Maria Georgescul who helped me answer my questions and Robert Kirk who corrected and proofread my thesis.
Contents

1 Introduction .................................................. 1

2 CALL .......................................................... 3
   2.1 History of CALL ........................................... 3
      2.1.1 1960/1970 ............................................. 4
      2.1.2 1980 .................................................. 7
      2.1.3 1990 .................................................. 12
   2.2 State of the Art .......................................... 13
   2.3 Translation Game ......................................... 16
   2.4 Conclusion ............................................... 21

3 CALL-SLT ..................................................... 22
   3.1 CALL-SLT Project ......................................... 22
   3.2 Speech Recognition ....................................... 25
      3.2.1 Data-driven vs. rule-based language modelling .... 27
         3.2.1.1 Data-driven language model .................... 28
            3.2.1.1.1 Hidden Markov Model .................... 28
         3.2.1.2 Rule-based language model .................... 29
            3.2.1.2.1 Context Free Grammar .................. 29
      3.2.2 CALL-SLT Approach: Regulus ....................... 32
         3.2.2.1 Feature Grammars / Unification Grammars .... 34
         3.2.2.2 Nuance Grammar Specification Language (GSL) 37
   3.3 Translation ............................................... 39
      3.3.1 Translation systems and Interlingua ............... 39
         3.3.1.1 Statistical Approach to Translation .......... 39
         3.3.1.2 Linguistic Approach to Translation .......... 40
         3.3.1.3 Interlingual Approach to Translation ....... 42
      3.3.2 Interlingua in CALL-SLT .......................... 44
   3.4 Help ..................................................... 46
   3.5 Interface ................................................ 47
## List of Figures

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.1</td>
<td>Behaviourist approach to language learning</td>
<td>4</td>
</tr>
<tr>
<td>2.2</td>
<td>Constructivist approach to language learning</td>
<td>8</td>
</tr>
<tr>
<td>2.3</td>
<td><em>A la rencontre de Philippe</em> (video)</td>
<td>11</td>
</tr>
<tr>
<td>2.4</td>
<td><em>A la rencontre de Philippe</em> (key word transcription)</td>
<td>11</td>
</tr>
<tr>
<td>2.5</td>
<td><em>Who is Oscar Lake?</em></td>
<td>13</td>
</tr>
<tr>
<td>2.6</td>
<td>Different kinds of exercises in <em>Tell Me More</em></td>
<td>15</td>
</tr>
<tr>
<td>2.7</td>
<td>Example of a video-question exercise in <em>Tell Me More</em></td>
<td>16</td>
</tr>
<tr>
<td>2.8</td>
<td>Phoneme pronunciation training</td>
<td>17</td>
</tr>
<tr>
<td>2.9</td>
<td>Translation Game system-user-interaction</td>
<td>17</td>
</tr>
<tr>
<td>2.10</td>
<td>Interface of the Wang and Seneff <em>Translation Game</em> prototype</td>
<td>18</td>
</tr>
<tr>
<td>2.11</td>
<td>Paraphrasing of input utterances</td>
<td>19</td>
</tr>
<tr>
<td>2.12</td>
<td>Flow chart of the Wang/Seneff <em>Translation Game</em></td>
<td>20</td>
</tr>
<tr>
<td>3.1</td>
<td>CALL-SLT with clip-art</td>
<td>23</td>
</tr>
<tr>
<td>3.2</td>
<td>Schema of the different steps in CALL-SLT</td>
<td>25</td>
</tr>
<tr>
<td>3.3</td>
<td>The three main parts of the speech recognition process</td>
<td>27</td>
</tr>
<tr>
<td>3.4</td>
<td>Parse tree for the noun phrase &quot;a beer&quot;</td>
<td>31</td>
</tr>
<tr>
<td>3.5</td>
<td>Parse tree for the sentence &quot;I prefer a sundown beer&quot;</td>
<td>32</td>
</tr>
<tr>
<td>3.6</td>
<td>The two levels of abstraction in Regulus</td>
<td>33</td>
</tr>
<tr>
<td>3.7</td>
<td>The nuance-compile utility</td>
<td>34</td>
</tr>
<tr>
<td>3.8</td>
<td>Statistical approach to translation</td>
<td>40</td>
</tr>
<tr>
<td>3.9</td>
<td>Linguistic approach to translation</td>
<td>41</td>
</tr>
<tr>
<td>3.10</td>
<td>Interlingual approach to translation</td>
<td>42</td>
</tr>
<tr>
<td>3.11</td>
<td>Interlingua example</td>
<td>43</td>
</tr>
<tr>
<td>3.12</td>
<td>Interface of the CALL-SLT prototype</td>
<td>48</td>
</tr>
<tr>
<td>4.1</td>
<td>Transfer rules (Source language to Interlingua)</td>
<td>65</td>
</tr>
<tr>
<td>4.2</td>
<td>Reverse transfer rules (Interlingua to source language)</td>
<td>65</td>
</tr>
</tbody>
</table>
List of Tables

2.1 Behaviourist and constructivist influences on CALL .......................... 13
4.1 Macros for singular nouns ............................................. 51
4.2 Macros for plural nouns ............................................. 52
5.1 Results of beginners’ questionnaire ................................. 71
5.2 Results of intermediates’ questionnaire ............................ 72
5.3 Results of advanced questionnaire ................................. 73
5.4 Results of the German CALL-SLT questionnaire ................ 74
5.5 Results of the German CALL-SLT questionnaire by a native-speaker 74
5.6 Results of the German CALL-SLT speech recognition performance 76
1 Introduction

This Master’s Thesis is dedicated to CALL (Computer Assisted Language Learning) in general and to the CALL-SLT Project in particular, with a main focus on developing and evaluating a version of CALL-SLT for German.

The thesis will give an introduction to CALL and its development and progress during the last fifty years. This will help to better understand the state of the art and the meaning of CALL in language learning from not only a technical, but also a pedagogical point of view. A section of the introduction to CALL is dedicated to Wang and Seneff’s Translation Game which is important for this thesis because it serves as a model for the CALL-SLT Project.

The CALL-SLT Project of the University of Geneva, which is conducted at the TIM department of the ETI and is supported by the Swiss National Science Foundation sets the context of this thesis. The project and the technologies used will be described in detail in order to give the framework for the practical section of the paper.

As mentioned above, the aim of this thesis is to develop and evaluate a version of CALL-SLT with which French and English native speakers can learn German as a second language. This practical section consists in writing a German grammar and corpus, as well as transfer rules, which allow to verify the semantic correctness of a learner’s utterance.

To finish, the CALL application developed in the practical section, as well as a prototype of CALL-SLT for English are evaluated. This evaluation will give an idea of the strengths and weaknesses of the program and will help improve the prototype in future versions.

The paper is divided into six major parts:

**Part one** consists of a short introduction to the goal and context of this thesis, as well as an overview of the thesis’ structure.

**Part two** will give an overview about the history of CALL, beginning in the 1960s until the 21st century. The state of the art of CALL will also be described in that second part, as well as the notion of the Translation Game, introduced by Wang and
Seneff, which constitutes the basis of this thesis.

**Part three** will be dedicated to the CALL-SLT Project and the technologies used. The two major technologies being speech recognition and translation. Both areas will be commented, with an emphasis on the Regulus approach used for the project.

**Part four** will demonstrate the work of the practical section written within the CALL-SLT project. The goal of this fourth part is to show the different steps which are necessary to create a CALL-SLT version for German. These steps include writing a German grammar, a German corpus, as well as transfer rules.

**Part five** will contain an evaluation of the developed CALL-SLT software, for both the English and German prototypes.

And to round off the thesis, **Part six** will give a conclusion, as well as an overview of future prospects.
2 CALL

CALL is the acronym for Computer Assisted Language Learning. It is mostly used in second language learning, which refers to the acquisition of a language other than one's mother tongue. The concept was first introduced in the 1950s and has survived through an important development until today. There are several advantages, as well as a number of disadvantages, in the use of CALL. The most obvious one is the lack of human intelligence: even though CALL programs have become more and more intelligent in recent years, they will never be able to substitute a human being, which is why CALL applications should be seen as a supplement to human language teachers. However, CALL applications are easier to access and to handle thanks to their electronic format, which are amongst the several advantages. Other advantages include the possibility for each student to advance at his/her own pace and the individualisation of the exercises to suit the needs of each individual.

This chapter will be divided into three different sections: section 2.1 will describe the developmental history of CALL from the 1960s until the 1990s; section 2.2 will then discuss the state of the art (21st century) of CALL in which recent activities in CALL technologies will be introduced. And finally, section 2.3 will be dedicated to the "Translation Game", a concept introduced by Wang and Seneff which lay the foundations for the CALL-SLT project (c.f. Chapter 3).

2.1 History of CALL

The history of CALL can generally be divided into three epochs, namely the 1960s and 70s, the 1980s and the 1990s. In the 1960s, everything began with Skinner's introduction of programmed instruction, which will be discussed in section 2.1.1. In the 1980s the approach to language learning was revolutionised by the constructivist approach which was perfectly supported by the microcomputer that was introduced in the same epoch (section 2.1.2). But it was only in the 1990s, with the arrival of the internet, that CALL technologies were used by a wide audience (section 2.1.3).
2.1.1 1960/1970

The 1960s and 70s were mainly influenced by the behaviourist school of thought, which was at its peak during the 1960s and claimed that most human behaviour is conditioned or learned rather than genetic\(^1\) The behaviourist language learning model is derived from Skinner’s stimulus and response theory. The aim of this theory was that the student is confronted with new knowledge, memorises these new things and concepts and then responds to the questions with his/her new knowledge. B.F. Skinner argued that all teaching methods should be concentrated on ‘cause-and-effect relationships’, since these can be established by observation and can therefore be proven with scientific procedures [Hung 2001]. This school of thought had an important effect on language teaching, whilst placing an emphasis on spoken exercises in the target language. An important part of the behaviourist approach to language learning is positive reinforcement, which means that the student receives positive feedback if s/he successfully finishes a task. Figure 2.1 summarises the most important aspects of the behaviourist design for (language) learning [Beatty 2003:17].

![A behaviourist design:](attachment:image.png)

**Figure 2.1:** Behaviourist approach to language learning

With the behaviourist school also came the concept of *programmed instruction*, which was introduced by Skinner in 1954 [Beatty 2003:87] and which lay the foundations for CALL applications. The main idea of this programmed instruction was to introduce

\(^1\) [http://thebookman.wordpress.com/2008/03/01/postmodern-terms-absence-to-curtain-wall/](http://thebookman.wordpress.com/2008/03/01/postmodern-terms-absence-to-curtain-wall/)
self-teaching with the help of either conventional textbooks or so-called teaching machines. Such teaching machines could automatically correct simple multiple choice exercises using a linear approach, which was typical for behaviourism. These machines allowed the student to actively take part in the learning process. The learning matter was divided into small steps and positive feedback was implemented to motivate the language students [Antoniadis 2006]. If the students did not advance, the learning matter was divided into even smaller steps in order to facilitate the students’ task; however, the student was never blamed if s/he did not succeed. The main goal of this approach was to individualise language learning, since each student could decide by him/herself how fast to proceed. In practice, the teaching machines gave the students instructional steps, telling them exactly what to do, then the students had to give active responses to the questions asked and in response they were rewarded with immediate feedback from the machine [Levy 1997:15].

Another approach to language learning which is also part of the behaviourist school of thought is the audio-lingual approach which encouraged drill and practice exercises. This approach focused principally on grammar exercises and less on vocabulary exercises. The positive aspect of these drill exercises was that they were easily programmable thanks to their systematic and routine character and lack of open-endedness [Kenning and Kenning 1990].

Two of the most important ‘first generation’ CALL projects, which both have their roots in the behaviourist approach are PLATO and TICCIT.

The PLATO Project (Programmed Logic for Automatic Teaching Operations), which was initiated at the University of Illinois in 1960 [Beatty 2003:18] was one of the first CALL projects. It was part of the CAI (Computer Assisted Instruction) movement, which was a further development of Skinner’s teaching machines and therefore directly emanated from behaviourism. This movement was designed to teach languages as well as other subjects, even though in the beginning it was mainly designed to learn Russian (at a time when the Cold War was in full swing). After an initial version, three more versions were to follow. The goal of the PLATO Project was
to provide a platform which could give interactive instructions for large numbers of students [Smith and Sherwood 1976:344], but which could also be adapted to the individual pace of each student. Some of the most important features were the so-called note files and the talk option which encouraged communication between users. A first spelling and grammar checker were also included to help correct the writing skills of the learners. At the same time the user’s records were registered for the student’s and teacher’s information, as well as for research purposes. As far as language learning exercises were concerned, PLATO mostly covered the fields of reading and vocabulary, writing and listening. Like most early language learning programs, PLATO followed the linear system of the behaviourist school, which means that learners always follow the same steps in the same way and receive points as a reward and advance to a next level if they answer correctly [Beatty 2003:19]. The goal of PLATO was to cover the more mechanical types of exercises in order to have class time dedicated to more expressive activities [Levy 1997:16]. An evaluation of the PLATO Project conducted by Murphy and Appel (1978) showed that both students and teachers found the system appealing; however, neither positive nor negative effects on student achievement could be proven. PLATO, nevertheless, advanced through the years, including new technologies such as a touch-screen in recent years and is today still used in Japan, where the electronic conglomerate TDK is still further developing the most recent version [Beatty 2003:22].

Another important project of the 1960s is the TICCIT Project (Time-shared, Interactive, Computer-Controlled, Information Television). The project was launched at the Brigham Young University within the MITRE Corporation in 1968 [Merrill 1980]. It is a computer-based instructional system, which differs from previous CAI programs in three major ways. The first difference lies in its design to teach primarily concept-classification and rule-using objectives rather than memory drill or problem-solving objectives; the second difference is that it is based on a courseware theory and as a third difference can be mentioned that it allows the user to choose which content to study and which strategy to use [Merrill 1980:4]. An evaluation of the TICCIT Project by Alderman (1978) stated that the performance of students using the plat-
form was significantly better compared to those in conventional classes; however, student’s and teacher’s attitudes were not as positive as they were for the PLATO Project.

The 1970s were, in summary, marked by a period of moderate growth in CALL technologies as well as some disillusion. Even though the National Science Foundation and the Office of Education made large investments in the USA, projects often disintegrated once federal funds expired [Levy 1997:21].

2.1.2 1980

After the 1960s and 70s during which Skinners behaviourist approach predominated the CALL domain, the 1980s were mostly influenced by the constructivist approach to language learning – the philosophical school that followed behaviourism.

The constructivist school is based on the language learning theory that human beings generate knowledge from their experiences or in other words "learners construct their own reality or at least interpret it based upon their perceptions of experiences, so an individual’s knowledge is a function of one’s prior experiences, mental structures, and beliefs that are used to interpret objects and events" [Jonassen 1991:60]. This approach mainly focused on processes and interactions, whether individually or socially (social constructivism), since learning is seen as an active process of constructing rather than acquiring knowledge [Hung 2001]. Furthermore, the interpretation of knowledge is dependent on the prior knowledge and beliefs of an individual, as well as of the cultural and social context through which the knowledge was constructed [Hung 2001]. The most important aspects of the constructivist design for (language) learning can be seen in Figure 2.2.

With the constructivist school of thought came the Communicative Language Teaching (CLT). In contrast to the behaviourist school of the 1970s, this new approach saw the goal of language teaching in training communicative competencies rather than
pure written multiple-choice exercises. The different practical aspects of this new
development will be shown in the multiple CALL applications, which will be intro-
duced later in this chapter.

Not only the learning theory changed in the 1980s, but also technology. The 1980s
became an important epoch for the CALL domain, due to the many technological
changes and inventions of these years.

The most important technological invention was the microcomputer. When the
microcomputer was invented in 1973, the interest in CALL grew dramatically com-
pared to earlier years and first introductory books on CALL, as well as specialised
journals dedicated to CALL, appeared in the early 1980s as a consequence of this
sudden interest [Levy 1997:20]. The introduction of the microcomputer made it
possible for language teachers to write their own CALL applications: all that was
required was a high-level programming language like BASIC. The most common
exercise types covered by these CALL programs were text reconstruction, gap-fill
exercises, speed-reading, simulation and vocabulary games [Levy 1997:23]. However,
even though microcomputers revolutionised the development of CALL applications,
high-end mainframe computers were still used in academic CALL research.
Another important change in technology of the 80s was the introduction of the videodisc, which allowed for some important changes in language learning. Bush and Crotty (1989) summarise them as follows:

- priority of listening over speaking
- exclusive use of the target language
- implicit rather than explicit grammar
- correction and corroboration through modelling
- special efforts to create a low-anxiety atmosphere

The three most successful videodisc programs were Macario, Montevidisco and Dígame. Whereas Macario followed the behaviourist linear approach of the 1970s, Montevidisco and Dígame featured constructivist non-linear approaches to language learning, thus giving the students the possibility to choose themselves what to learn and how to do so. This new non-linear approach added much to the individualisation of the learning process in CALL software, as it allows for greater learner autonomy and encourages critical thinking [Beatty 2003:26].

Yet another technological invention of the 1980s is the so-called authoring system. Authoring systems are defined as computer software which allow tutorials, courseware or other interactive programs to be created [Murray 1999]. One of the best-known authoring programs is Storyboard. HyperCard is another authoring system which was released in 1987 for Mac computers. It provided a constructivist non-linear approach to text production and consumption, with which computer-based learning units could be created. It was wide spread, since it was installed on all Mac computers by default. HyperCard is a database system based on the concept of a stack of virtual index cards which hold data. Data could contain various things, such as text, images, audio files, animations or videos. Thanks to specially added buttons, which link the different index cards, the users are lead to a new card with further questions, information or answers once they have read the information or answered the question of one card [Beatty 2003:34].
The introduction of authoring systems was specifically interesting for language teachers interested in writing their own CALL programs, since they no longer had to learn a high-level programming language to do so.

With these technological changes in the 1980s also came an important CALL project, namely the ALLP (Athena Language Learning Project). It was initiated as part of the Athena Project at the Massachusetts Institute of Technology (MIT) in 1983, which was an 8-year research program to explore innovative use of computers in education [Levy 1997:27]. The aim of the ALLP was to create a communication-based prototype to conduct beginners and intermediate courses in French, German, Spanish, Russian and English as a second language. Instead of working with mainframe computers, the project used UNIX\textsuperscript{2} workstations. Within the project two important research initiatives were promoted: one being the MUSE multimedia authoring environment, which allowed for cross-referencing of video, audio and graphical material via basic structures of hypertext and hypermedia [Levy 1997:27] and the other being MIT-based artificial intelligence techniques, which aimed to develop a natural language processing system that could intelligently guess meanings intended from minimal clues and check its understanding with the user [Murray 1989]. Just like the TICCIT Project, ALLP was also designed to supplement classroom activities and did not intend to substitute the learner-teacher relationship.

A la rencontre de Philippe and Dans le quartier Saint-Gervais are two CALL programs which were introduced in the 1980s as a result of the technological changes discussed above. Both programs feature exercises to learn the French language via a constructivist approach to language acquisition with interactive real life scenarios in which the learner can make his/her own documentaries and multimedia documents from the resource material provided by the program [Beatty 2003:32]. The main idea of A la rencontre de Philippe is to improve students’ language understanding skills by helping the young freelance journalist Philippe to get around in Paris and help him with his several problems (e.g. money problems, job problems or problems with his

\textsuperscript{2} UNiversal Interactiv eXecutive or UNiversal Inter-eXchange
girlfriend). In Figure 2.3 we can see an example of a video exercise in which Philippe explains all the possible resources that can be used in the process of searching for an apartment (e.g. the newspaper, the telephone, etc.). The student can stop the video and repeat certain segments which have not been readily understood. Another help function can be seen in Figure 2.4. It shows how a student can access keywords which are used in the dialogue via a transcription of the relevant segment with additional information, which helps understand the meaning of the vocabulary. In Figure 2.4 the student learns what the expression "donner un coup de main" means in the context of the video watched beforehand. The real life scenarios help to encourage the learning process and the authentic dialogues which can be listened to help improve the understanding of spoken Parisian French by native speakers in real life contexts.

![Figure 2.3: A la rencontre de Philippe (video)](image)

![Figure 2.4: A la rencontre de Philippe (key word transcription)](image)

The experiences from large-scale projects in the 1970s and 80s were then also used in smaller projects for microcomputers; however, an evaluation by Dunkel (1991) stated that CAI and CALL were ‘limited and somewhat equivocal’ at that epoch. An important change during the 1980s was the movement towards computer workstations (such as UNIX) and away from larger mainframe computers, which meant
limited computer power, but it allowed greater access by a broader public.

2.1.3 1990

In the 1990s there was a number of CALL projects which led to further developments in the various aspects involved in CALL research, such as interactivity, communication and the implementation of real life contexts.

One such project was the International Email Tandem Network which was initiated at the University of Trier [Levy 1997]. It encouraged language learning in a tandem system, based on written e-mail communication. Forum discussions were also promoted for all participants, as well as different subnets for the various language combinations which were continually monitored by system administrators [Levy 1997:33].

Another project was the CAMILLE\textsuperscript{3} France InterActive Project, which was conducted by a consortium of partners from the UK, France, Spain and the Netherlands. Its goal was to provide beginners courses in Dutch and Spanish, as well as advanced courses in French and English [Levy 1997:34]. It is a learning resource rather than a teaching resource and it combines a communicative competence approach to language acquisition with an interactive multimedia environment.

A popular CALL program of the 1990s was Who is Oscar Lake (cf. Figure 2.5) which was an interactive program available in English, Spanish, French, German and Italian and offered a live action video environment with a 1200 word vocabulary and 33 different language learning activities [Beatty 2003:35]. It also offered answers, comments and advice from native speakers. Thanks to the interactive environment, the user could work through various possible endings of the various scenarios.

As a summary of the development of CALL throughout history, Table 2.1 (based on [Hung 2001]) gives an overview of the philosophical schools and their impact on

\textsuperscript{3} Computer-Aided Multimedia Interactive Language Learning
language learning, as well as the different CALL applications which came with them:

![Figure 2.5: Who is Oscar Lake?](image)

<table>
<thead>
<tr>
<th>Learning</th>
<th>Behaviourism</th>
<th>Constructivism</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type of Learning</td>
<td>Memorisng and responding</td>
<td>Problem solving in realistic and investigative situations</td>
</tr>
<tr>
<td>Instructional strategies</td>
<td>Present for practice and feedback</td>
<td>Provide for active and self-regulated learner</td>
</tr>
<tr>
<td>Key concepts</td>
<td>(Positive) reinforcement</td>
<td>Personal discovery from first principles</td>
</tr>
<tr>
<td>Computer-mediated tools</td>
<td>Variety of drill and practice computer-based learning software</td>
<td>Individual generic purpose tools</td>
</tr>
<tr>
<td>CALL applications</td>
<td>- PLATO - TICCIT</td>
<td>- A la rencontre de Philippe</td>
</tr>
</tbody>
</table>

Table 2.1: Behaviourist and constructivist influences on CALL

### 2.2 State of the Art

As the world became more global and languages became increasingly more important in both professional and personal aspects of people’s lives, the need to learn foreign languages grew from year to year.

One of the most effective ways of learning a foreign language is to live in the respective country in order to practice understanding and hone communication skills; however, this method demands a considerable amount of time, as well as funds, which most people are unable to afford. Another good method is the tandem method, where
oral communications are held with a study partner who is a native-speaker of the desired language, but this method also demands great effort and most importantly the time of two independent study partners. The goal of CALL in the 21st century therefore became to develop an actual communication partner for language students.

Natural Language Processing (NLP) has played an important role in the development of CALL in recent years. The goal of NLP is to build computational models of natural languages in order to being able to analyse and generate them [Bharati 1994].

There are some major NLP technologies which are particularly interesting for CALL applications, such as lexical thesauri, parsers, generators, semantic interpreters and generators, as well as speech synthesisers [Tufis 1996]. Lexical thesauri are very useful in CALL applications in order to provide lists of synonyms, antonyms, hyper- and hyponyms, as well as to give the contexts in which the words are used. Parsers are important to treat the student’s input; however, they must be error tolerant and they should, at the same time, have a broad enough coverage to be useful for both beginners and advanced students. In this context, corpora shall also be mentioned, as they play a significant role in CALL, since they can help detect errors and include native tongue speaker examples which are important aspects in second language acquisition.

Generators are an essential piece of technology when it comes to the communication between the student and the computer. In order to create an environment in which the student can actually communicate with the machine, the computer must be able to produce natural language output which is done by generators. And finally speech synthesisers are the key technology to train the student’s pronunciation skills.

Even though the technological means are at hand, as has been seen, programs which offer real spontaneous dialogue options between the computer and the language learner are very expensive to realise. This is why most CALL systems of the 21st century so far use a so-called ‘closed response design’ in which the responses are limited (either by a very tight question design or by different answer proposals, from which the learner can choose) [Nerbonne 2003]. An example of a closed response
question is "Would you like a drink?" for which the response can be either "yes" or "no"; other closed response questions which have more scope than yes/no-questions are questions of the type "What would you like to drink?" for which the answers are of a limited range from which the learner will choose one option.

One of the most popular commercial CALL programs of the 21st century is Auralog’s Tell Me More. It is a piece of software which offers various types of exercises, including speech recognition technology to train pronunciation.

Before each training session the user can choose between a "Free-To-Roam Mode" in which the learner can choose the topics him/herself, as well as the nature of the exercises s/he wants to complete; a "Guided Mode" which is a programmed mode adapted to the user’s goals and time constraints; and finally, a "Dynamic Mode" in which the learner’s progress is evaluated automatically by the system so that the exercises can be adapted to the learner’s needs as s/he progresses. The user can also choose between different dialects. When learning English s/he can therefore choose between British and American English, which specifically adapts to the learner’s needs and aims.

The exercises are roughly divided into pronunciation, activities and video exercises (cf. Figure 2.6).

![Figure 2.6: Different kinds of exercises in Tell Me More](image)

Pronunciation exercises include dialogue, sentence pronunciation, word pronunciation and phonetic exercises and activity exercises include picture/word association,

---

Tested version: English 1 for Beginners on DVD-ROM, Montigny-le-Bretonneux, Auralog, 2005.
word searches, word association, find the right word, fill-in-the-blanks, words and topics, words and functions, grammar practice, mystery phrase, crossword puzzles, word order, sentence practice, dictation and written expression exercises. The video exercises include different video scenarios, for which the language learner has to answer questions. As can be seen in Figure 2.7, the student first watches a video with the transcribed text to the right of the video and then s/he moves on to the question section in which s/he has to answer multiple choice questions about the video that has just been watched. The challenge is that more than just one answers may be correct.

![Figure 2.7: Example of a video-question exercise in Tell Me More](image)

For all exercise types *Tell Me More* focuses on real-world scenarios in order to motivate the learner to train the foreign language in useful scenarios (which can be chosen by the user on demand). Such scenarios include for example a situation (in a video interaction game), in which the learner is confronted with a ticket officer in a train, whose questions and comments have to be answered correctly. Thanks to the speech recognition technology, the system continuously corrects the learner while s/he completes a pronunciation exercise. The user also has the possibility to train certain phonemes specifically, as can be seen in Figure 2.8. This option is especially useful if the language learner experiences difficulties with a particular phoneme, as for example the "th" in English.

### 2.3 Translation Game

Since good programs that offer real dialogue interaction between the computer and the language learner are still very difficult to realise from a technical point of view, Wang and Seneff opted for a web-based *Translation Game* in 2006 [Wang and Seneff
The idea of the translation game is to give the language learner a sentence in his/her mother tongue which s/he then orally translates into the language s/he wishes to learn. Since the game is offered online, all the users need is a computer with an internet connection, the Java run time environment, a web browser, a microphone and headphones.

The translation game prototype by Wang and Seneff is based on a flight information domain and is aimed at learners of the Chinese language. An example of the interaction between the system and the user is displayed in Figure 2.9 [Wang and Seneff 2007].

<table>
<thead>
<tr>
<th>System:</th>
<th>(greet and prompt user) Welcome! You are playing at level four. Okay, here’s your first sentence to translate: The return date is Tuesday October 25.</th>
</tr>
</thead>
<tbody>
<tr>
<td>User:</td>
<td>(translate with error) fan3 hui2 ri4 qi1 shi4 yue4 er4 shi2 wu3 hao4 xing1 qi1 er4.</td>
</tr>
<tr>
<td>System:</td>
<td>(paraphrase in Chinese) fan3 hui2 ri4 qi1 si4 yue4 er4 shi2 wu3 hao4 xing1 qi1 er4.</td>
</tr>
<tr>
<td>System:</td>
<td>(paraphrase in English) The return date is Tuesday April 25.</td>
</tr>
<tr>
<td>User:</td>
<td>(repeat date) shi2 yue4 er4 shi2 wu3 hao4.</td>
</tr>
<tr>
<td>System:</td>
<td>(paraphrase in Chinese) shi2 yue4 er4 shi2 wu3 hao4.</td>
</tr>
<tr>
<td>System:</td>
<td>(paraphrase in English) October 25.</td>
</tr>
<tr>
<td>System:</td>
<td>(respond to user) Great! You successfully translated the sentence in separate fragments. Let’s move on to the next sentence: I need to fly to Hong Kong the day after tomorrow.</td>
</tr>
<tr>
<td>User:</td>
<td>(click on the help button)</td>
</tr>
<tr>
<td>System:</td>
<td>(repeat English prompt) I need to fly to Hong Kong the day after tomorrow.</td>
</tr>
<tr>
<td>System:</td>
<td>(offer example translation) wo3 xu1 yao4 hou4 fai1 xiang1 gang3.</td>
</tr>
<tr>
<td>User:</td>
<td>(initiate the translation) wo3 xu1 yao4 hou4 fai1 xiang1 gang3.</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>System:</td>
<td>(respond to user) You translated nine out of ten sentences. You took on average 1.6 turns per sentence. You have advanced to level five. Would you like to play another round?</td>
</tr>
</tbody>
</table>
The translation game is easy to use and its structure is intuitional. Once the learner accesses the start page of the Translation Game s/he can enter a username as well as a difficulty level (optional) and the number of sentences s/he wants to translate in each session (optional). The main page is then displayed (Figure 2.10) and provides details on the meta commands as well as the random selection of sentences to be translated in the current game session (e.g. "Is there a flight around one o’clock?" or "I do not want to leave from Hong Kong"). In total, there are over 1000 templates (in English) which are categorised by their length and linguistic challenge into different difficulty levels. The templates were assembled semi-automatically from a corpus of real user utterances.

Once the game has been started the learner will be shown a first sentence stimulus in English which he has to translate into Chinese. The system then paraphrases the learner’s utterance (cf. Figure 2.9 and 2.11 [Wang and Seneff 2006]) and if the translation is judged as being correct by the machine translation system, the student will be congratulated and moved on to the next sentence by the system.
If the translation is wrong (as can be seen in Figure 2.9, where October has been falsely translated by April) the learner has three options of how to proceed: s/he can either make use of the "help me" button which will give him/her a translation of the stimulus sentence; s/he can stop the game session via the meta command button "give up"; or s/he can simply retry to translate the prompt. When the game session is finished, meaning that the learner has translated all the sentences from the initial list, a summary of the performance is shown and a new session can be played.

The difficulty level of the new game session is set via the following formula, which is based on the average number of turns it took the learner to translate each sentence stimulus correctly:

\[
\text{new\_index} = \text{old\_index} + (3.0 - \text{score}) \times \text{step\_size}
\]

The new difficulty level (new_index) is calculated depending on the previous difficulty level (old_index), the learner’s score on each translation and the step size, which Wang and Seneff decided to automatically set at 5% of the total number of templates. The difficulty level stays the same – however with different stimulus sentences (with varying cities, dates and times) – if the learner takes on average 3 turns per exercise; s/he moves to a lower level if s/he uses more than 3 turns per sentence and s/he moves up to a higher level if the average is below 3 turns per sentence. It must be noted that the consulting of the meta command "help" counts as one turn [Wang and Seneff 2007].
From a technical point of view, the system uses two main technologies, which are speech recognition and a machine translation system. The learner’s utterance is first processed by a speech recogniser (based on the SUMMIT landmark-based speech recognition system), once the utterance is recognised it is then determined whether or not it corresponds to the meaning of the English stimulus sentence [Wang and Seneff 2007].

To evaluate the semantic correctness, the English stimulus provided by the student is paraphrased into Chinese with an interlingua based machine translation system, and then this translation is compared to the learner’s translation via an encoding of the meaning in terms of [key:value] [Wang and Seneff 2007] (cf. flow chart in Figure 2.12). If a partial match occurs, the learner can simply re-translate the faulty part of the sentence (e.g. errors in dates or times), such an example can be seen in Figure 2.9, where just the month has been translated incorrectly ("The return date is Tuesday October 25" has incorrectly been translated as "The return date is Tuesday April 25"). Instead of repeating the whole sentence, it is enough if the student correctly repeats the month in order to get a correct answer (e.g. "October 25").

![Figure 2.12: Flow chart of the Wang/Seneff Translation Game](image)

The main problem is that a faulty recognition of the learner’s utterance can lead to the rejection of a perfectly correct translation – which is highly frustrating for the language learner – or to the acceptance of an incorrect translation. In order to minimise the user frustration, the system automatically moves on to the next stimulus after a certain number of faulty translations and the learner also has the option of aborting the game session via the meta command "give up".
2.4 Conclusion

In this chapter, we have seen that CALL became an interesting tool for second language learning from the 1960s onwards; however, while during the early years the main focus was on outsourcing easy and repetitive exercises to CALL applications (e.g. multiple choice questions or fill-in-the-blanks exercises), recently far more complex technologies in the CALL domain have been exploited. It therefore became possible to use such applications not only to train written language skills, but – thanks to speech recognition – also spoken language skills. The most recent development discussed in this chapter – Wang and Seneff’s Translation Game – is a very promising new approach to language learning using CALL technologies and will be further developed in the CALL-SLT Project, which is the main focus of this Master’s Thesis and will be the topic of the following chapter.
3 CALL-SLT

In this chapter I will introduce the CALL-SLT Project which constitutes the main focus of this Master’s Thesis and which is based on the Translation Game concept we have seen in the previous chapter (2.3).

The chapter is divided into five major sections: first, the CALL-SLT Project will be introduced and then the three main components – speech recognition, machine translation and the help function – will be discussed in detail. Finally the fifth section is dedicated to the interface of the CALL-SLT prototype.

The practical section which I have carried out within the CALL-SLT project will be the topic of chapter 4 in which I will describe the different steps in the development of a CALL-SLT version for German.

3.1 CALL-SLT Project

The CALL-SLT Project is a project supported by the Swiss National Science Foundation (SNSF) and is conducted by the ETI/TIM/ISSCO research group which is part of the School of Translation and Interpretation (ETI)/Multilingual Information Processing Unit (TIM) of the University of Geneva.

The goal of this project is to use Wang and Seneff’s Translation Game (as described in 2.3) as a base, but the ideas introduced by Wang and Seneff will be developed in two main ways. The first point focuses on the number of languages involved: whereas Wang and Seneff concentrate their translation game on one language pair (namely English-Chinese), the CALL-SLT project aims to address a multilingual language learning community. The CALL-SLT application will also allow the language learners to practice speaking as well as listening thanks to a bi-directional approach to the system, which supports translation both to and from the source language.

The second main point where this project differs from Wang and Seneff’s translation game is that the stimuli does no longer consist of prompts in L1 (= mother tongue), since this has the effect of tying L2 (= foreign language) too closely to L1 in the student’s mind. The goal is thus to create a ‘spontaneous’ translation game. Instead...
of providing sentences to translate, CALL-SLT aims at presenting the student with scenarios (e.g. order a series of dishes or asking for a table), so that no sentence structure is given to start with. Another idea is to depict the scenarios with clip-art pictures. An example of such a clip-art approach can be seen in Figure 3.1. [Rayner et al. 2010].

Figure 3.1: CALL-SLT with clip-art

The three main technologies used in this project are the Regulus platform to construct grammar-based speech understanding systems, the Regulus-based speech translation and the framework for providing intelligent help in speech-enabled systems:

1. The Regulus platform used in the CALL-SLT project is an Open Source platform for building grammar-based speech systems. It has been developed through a collaboration between the University of Geneva and the NASA Ames Research Center which began in 2002. This piece of technology is used to recognise a sentence and for its syntactic and semantic analysis.

2. The Regulus-based speech translation is a rule-based interlingual translation just like the one used by Wang and Seneff. The main aim of the speech translation within CALL-SLT is to create a semantic prompt, which is based on the interlingua. It verifies if a sentence correctly corresponds to a given prompt.

3. As far as the help option is concerned, the CALL-SLT project will include an intelligent help module, which will provide students with examples covered by the Regulus grammar. These examples designed to support the learner will be chosen using criteria to be as close to the user’s initial utterance as possible and will be collected from a corpus of example sentences. There will be both written, as well as spoken help available.
Before we take a detailed look at the technologies used in this application, the schema in Figure 3.2 will give a first idea of the different steps of CALL-SLT. As can be seen, in the beginning a sentence is randomly chosen from the corpus (1). This sentence is then first transformed into its source representation with the source language grammar (2) and then into the interlingua with so-called transfer rules (3). As a next step the interlingua grammar transforms the interlingua representation of the sentence into a human-readable prompt which is displayed in the user’s mother tongue (4). The language learner sees the prompt and speaks a translation of the interlingua-surface-form prompt into the microphone (5). This spoken input is then processed by a speech recogniser and results in the source representation of the user’s utterance (6). This source representation is then transformed into the interlingua with the transfer rules (7). Once the interlingua of the user’s utterance has been obtained, it is compared to the interlingua of the initial prompt (8). If the two interlingua correspond, the translation was correct and the learner will receive positive feedback in form of a happy emoticon (9). S/he can then move onto the next prompt. If the user’s translation was faulty, however, the interlingua of his/her utterance will not correspond with the one of the initial prompt. In this case a doubting emoticon will be displayed (9) and the student will have the option of getting help from a number of example phrases with the correct underlying interlingua structure collected from the corpus (10).

In the following sections, the different technologies will now be introduced in more detail. In section 3.2 Speech Recognition will be explained with a main focus on the difference between data-driven and rule-based language modelling. In section 3.2.2 we will then take a look at the rule-based approach used in the CALL-SLT project. We will see how the Nuance Grammar (3.2.2.2) is derived from the Feature Grammars (3.2.2.1) and the interlingua approach to translation used in CALL-SLT (3.3.2) will be discussed in section 3.3. The help function will be introduced in section 3.4 and to conclude this chapter the interface of the CALL-SLT application will be introduced in section 3.5.
3.2 Speech Recognition

Even though the first speech recognisers were developed in the 1950s [Waibel 1990:9], they have only become powerful enough for actual use in recent years. Nowadays, speech recognition is used in many different domains, e.g. for telephone applications, such as those used in call-centres. This technology has rapidly become highly interesting to CALL, since it allows language learners to train their speaking and pronunciation skills effectively. In CALL, the quality of speech recognition technology is particularly important since a real spoken interaction with the computer is required to be obtained. In order for the language learner to being able to train his/her language skills effectively – in particular the pronunciation skills – it is im-

---

**Figure 3.2:** Schema of the different steps in CALL-SLT
important that s/he can actually speak with the computer and that s/he is readily understood by the system.

The aim of speech recognition is to transform spoken input into written text. This means that an acoustic signal which is registered via a microphone is transformed into a chain of characters, as can be seen in the following graph:

![Speech Recognition Diagram](image)

In order to transform the acoustic signal into a written chain of characters, the speech recognition system needs three main pieces of knowledge, namely signal processing, an acoustic model and a language model. The signal processing converts the waveforms of the acoustic signal into digital representations and the acoustic model specifies how utterances in the language are realised as digital representations of waveforms [Rayner et al. 2006:5]. The acoustic model also contains a pronunciation lexicon which describes each word with a list of phonemes. While the signal processing and the acoustic model relate the sounds to words, the language model on the other hand specifies the words which the application should be able to recognise and the order in which they are allowed to occur. This helps the system to find a word faster, since the search space is restricted to those lists of words specified by the language model, instead of searching all possible lists of words in the language. Figure 3.3 [Rayner et al. 2006:6] displays these three processing steps and in which order they occur during the speech recognition.

There are, however, a number of difficulties to overcome in the use of speech recognisers: since there are often different possibilities to pronounce a word (e.g. different dialects and accents), the recognition system must be able to take all possibilities into consideration; another problem could be caused by sounds which are uttered, but which are not actual words which must be recognised (e.g. ‘hmm’ or ‘ah’). It can also be difficult to distinguish the single words from each other when there are
elisions, which means that one or several sounds are omitted either between two words or within a word in order to facilitate pronunciation for the speaker.

The most important factors in order to achieve satisfactory speech recognition are that it is conducted in a quiet environment and that the speaker has a clean speech (i.e. no heavy accent or mumbling), as well as the equipment being of a satisfactory quality (e.g. a high-quality microphone).

In a CALL application which is used by non-native speakers, the most important problem is accents. If the system is not able to recognise utterances with a heavy accent, it will be highly frustrating for the language learner if none of his actually correct utterances are recognised.

3.2.1 Data-driven vs. rule-based language modelling

As mentioned above, the language model specifies which words should be recognised by the system and in which order they are allowed to appear. In fact, there are two main language model approaches to distinguish acceptable from unacceptable sequences (e.g. ‘fifty nine’ = acceptable; ‘fifteen nine’ = unacceptable). The first is the data-driven approach, which will be described in section 3.2.1.1 and the second

Figure 3.3: The three main parts of the speech recognition process
is the rule-based approach, which will be used in the CALL-SLT project and will be
described in more detail in section 3.2.1.2.

### 3.2.1.1 Data-driven language model

Data-driven language models are also often referred to as "corpus-based" or "sta-
tistical" models, since they are based on the statistical probability that a word is
followed by a certain other. In order to calculate this probability so-called N-grams
(in most cases either bigrams (2-word-sequences) or trigrams (3-word-sequences)) are
used. If we take the example cited above, we will find out that the probability of
"fifteen" being followed by "nine" is largely inferior to the probability of "fifty" be-
ing followed by "nine". Therefore a speech recogniser would recognise "fifty nine" as
being a correct utterance, due to its high statistical probability, and "fifteen nine"
as being an incorrect utterance. But in order to have a representative probability
chart, a large quantity of data in form of a corpus is needed, which is not always
easy to have access to and which is subsequently the most important disadvantage
of data-driven language models.

Another main disadvantage of data-driven language models for CALL applications is
that they do not have an underlying grammar, which is why grammatical errors can
easily be left undetected. Since a CALL application aims at teaching grammatical
correctness, this model is not very favourable for the CALL domain.

#### 3.2.1.1.1 Hidden Markov Model

Most statistical speech recognisers (e.g. early editions of the Dragon System) are
based on Hidden Markov Models (HMM). These models are of interest for speech
recognition owing to two main factors: firstly, the mathematical framework of HMM
and secondly, its implementational structure [Juang 1991:257]. As far as the mathe-
matical framework is concerned, the HMM technology is based on statistics and the
probability theory, which allow for the model to be studied analytically, since the
mathematical structure can easily be traced. HMM models combine the modelling of
stationary stochastic processes for the short-term spectrum of speech and the tem-
poral relationship between the processes in a probability space. The processes can also be decomposed into the joint probability of observation sequence and the state sequence, which is of advantage since this flexibility with consistency can easily convert time-varying signals, such as speech in the case of speech recognition. Another advantage of HMM is that they are relatively easy to train from a set of training data (= sequences of observations). As far as implementation is concerned, numerical difficulties, as well as the computational complexity which come along with the implementation of algorithms can fairly easily be dealt with [Juang 1991].

3.2.1.2 Rule-based language model

Rule-based language models (also called "grammar-based language models" or "linguistic language models") differ from data-driven models as they do not need extensive training data, instead a so-called Context Free Grammar (CFG) is used which describes all acceptable structures. The speech recognition system will only try to recognise strings which are covered by the underlying grammar [Rayner et al. 2006]; however, most recognition systems represent CFG with slightly different notations. The Nuance platform [Nuance 2001], which has been used in this project, for example, uses the Grammar Specification Language (GSL) formalism.

The main advantage of rule-based language models is that there are no grammatical errors thanks to the underlying grammar and that no training data is needed; however, a disadvantage of a CFG approach is that a large number of rules must be written in order to cover all possible structures which are grammatically correct. Another disadvantage of rule-based language models is that it is very difficult to write a good CFG grammar for a reasonably complex domain [Rayner et al. 2006:12].

In the following sections we will now take a closer look at CFGs and how Regulus deals with the problems mentioned above.

3.2.1.2.1 Context Free Grammar

The idea of phrasal constituency and therefore also Context-Free Grammars (CFG) which constitutes the basis of the rule-based language model, can be lead back to
Wundt’s description of a sentence in the very beginning of the 20th century [Wundt 1900:240]:

“Den sprachlichen Ausdruck für die willkürliche Gliederung einer Gesamtvorstellung in ihre in logische Beziehung zueinander gesetzten Bestandteile”

A Context Free Grammar (CFG) – also called Phrase-Structure Grammar – is a device for generating and parsing sentences [Jurafsky 2006]. It is probably the most common language for grammars, which is handled by computer parsers and was formalized in 1956 by Chomsky [Jurafsky 2006].

CFG grammars are based on the notion that a group of words can behave as a single unit (e.g. a noun phrase), called a constituent [Jurafsky 2006:2]. The role of the CFG is then to model these constituency facts. CFGs consist of a set of rules (or productions). Each line of the grammar is a so-called rule, which dictates how symbols of natural languages can be grouped together. The second part of a CFG – next to the rules – is a lexicon of words and symbols.

A noun phrase can, for example, be described as follows in a CFG [Jurafsky 2006]:

\[
\begin{align*}
\text{NP} & \rightarrow \text{Det Nominal} \\
\text{NP} & \rightarrow \text{ProperNoun} \\
\text{NP} & \rightarrow \text{Noun} \mid \text{Nominal Noun}
\end{align*}
\]

These rules indicate that a noun phrase (NP) can consist of a determiner and a nominal, a proper noun or simply of one or several nouns. Furthermore, determiners and nouns will also be specified in the lexicon:

\[
\begin{align*}
\text{Det} & \rightarrow \text{a} \\
\text{Det} & \rightarrow \text{the} \\
\text{Noun} & \rightarrow \text{beer}
\end{align*}
\]

With the rules described above, the parse tree in Figure 3.4 can be created.

---

5 “The linguistic expression for the arbitrary division of a total idea into its constituent parts placed in logical relations to one another”
6 Chomsky type 2
7 The arrow must be read as "rewrite the left of the arrow with the string on the right"
8 nominal = a noun or a word group that functions as a noun phrase
Figure 3.4: Parse tree for the noun phrase "a beer"

There are two classes of symbols used in the rules: the first class are terminal symbols and the second are non-terminal symbols. While the terminal symbols are introduced by a set of rules in the lexicon, the non-terminal symbols express clusters or generalisations of the terminal symbols of the lexicon [Jurafsky 2006]. To the left of the arrow is always a single non-terminal symbol – unlike in Context-Sensitive Grammars, where there can be a number of non-terminal symbols on the left hand side of the arrow – and to the right of the arrow there are one or several terminals or non-terminals.

As can be seen in the parse tree in Figure 3.4, the noun phrase NP is the starting point from which a set of strings is derived; however, usually CFG grammars have a starting point S, which can be interpreted as the ‘sentence’ node. A sentence can then be described as follows:

\[ S \rightarrow NP \ VP \]

This means that a sentence can be made out of a noun phrase followed by a verb phrase. A complete parse tree of the sentence "I prefer a sundown beer" would look as represented in Figure 3.5 (adapted from [Jurafsky 2006]).

Since parse trees are rather space consuming they can also be written in a so-called bracketed notation format, which would result in something like the following for the sentence "I prefer a sundown beer":

\[
[S \ [NP \ [Pro \ I]] \ [VP \ [V \ prefer] \ [NP \ [Det \ a] \ [Nom \ [N \ sundown] \ [Nom \ [N \ beer]]]]]]
\]
A major disadvantage of CFGs is that a separate rule must be written for each possible form, therefore a very large number of rules must be written in order to produce a comprehensive grammar.

In the following chapter, we will now take a look at the Regulus approach to grammar writing, which has been used in the CALL-SLT project. The main advantage of Regulus, as we are about to see, is that it allows for the grammars to be written in a more compact way and that they can then be compiled into CFG in order for Nuance to be able to treat them.

3.2.2 CALL-SLT Approach: Regulus

The functionality of Regulus in terms of speech recognition is that typed unification grammars (or feature grammars\footnote{Feature Grammars (FG) and Unification Grammars (UG) are treated as quasi-synonyms in this thesis} are compiled down into annotated context-free grammar language models which are then expressed in the Nuance Grammar Specification Language (GSL), which is also a kind of CFG [Rayner et al. 2004]. By using the Nuance Toolkit compiler utility, GSL language models can then be converted into runnable speech recognisers, which means a unification grammar can be compiled
into a speech recogniser. The main advantage of this approach is that the grammars are much more compact thanks to the unification grammar (cf. 3.2.2.1).

The Regulus approach uses two levels of abstraction which can be seen in Figure 3.6. The first level of abstraction is the specialisation of the general Feature Grammar (FG). Since the extensive general FG is too complex to be compiled into CFG form, an appropriate subset of this grammar is extracted via the EBL\(^{10}\) algorithm. This method takes into consideration the general FG, the added domain lexicon, a corpus of training examples and a set of declarations, the so-called operationality criteria. The goal of this grammar specialisation is to reduce the size and the ambiguity of the initial FG and to create a manageable domain FG [Rayner et al. 2006:14]. The second level of abstraction is the compilation of the abstract and complex specialised Feature Grammar into a CFG grammar.

![Figure 3.6: The two levels of abstraction in Regulus](image)

Once the CFG grammar has been transformed into a GSL format, it must then be compiled into a Nuance recognition package with the `nuance-compile` utility, as can be seen in Figure 3.7 [Nuance 2001].

In the following section, we will take a look at the grammar formalism that is used in Regulus and has been mentioned above. We will see that Feature Grammars are an interesting alternative to CFGs, because they allow rules to be written in a more

\(^{10}\) Explanation Based Learning
3.2.2.1 Feature Grammars / Unification Grammars

Feature Grammars are similar to CFGs in the way they work. The same structures that can be described with a CFG can also be represented in a Feature Grammar; however FGs are much more compact, meaning that less rules are necessary to describe the same structures. This is very helpful when writing a grammar, since it takes less time to write the rules and at the same time the grammar appears more organised.

The basic idea of Feature Grammars is that of feature-value pairs, meaning that values are assigned to features which are then attached to a category and finally the categories can be combined into rules. Such unification grammars also allow to verify the compatibility of the values, which is important in order to ensure that, for example, a determiner and a noun have the same singular/plural value.

The main advantage of FGs is that the number of rules can be highly reduced because they allow for multiple rules to be collapsed into one, as well as the fact that the values of features can be variables (starting with upper-case), as well as constants (starting with lower-case).

A simple example is given below:
singplur = sing 
sem = 1 
-----> 
spec : [sem = 1, singplur = sing] 
-----> 
spec : [sem = 1, singplur = sing] --> one.

In the example cited above, we can see in the first step that the value sing is assigned to the feature singplur and the value 1 is assigned to the feature sem. In the second step, the feature-value pair [sem = 1, singplur = sing] is attached to the category spec. And finally, in the last step the category is combined into a rule in which the left-hand side of the arrow corresponds to the category and the left-hand side of the arrow shows the surface word one.

Now if we want to write a rule for a noun phrase, we can write a single rule including singular noun phrases, as well as plural noun phrases if they have the same structure, since the feature singplur can either take the value sing or plur. At the same time the unification will ensure that the singplur-values of the two categories spec and noun are the same and therefore compatible.

np : [sem = [[spec, S], [noun, N]], singplur = Singplur] --> 
   spec : [sem = S, singplur = Singplur], 
   noun : [sem = N, singplur = Singplur].

However, all the used features and values also have to be declared. In Regulus there are four different kinds of declarations, namely feature_value_space, feature, category and top_level_category declarations [Rayner et al. 2006].

- feature_value_space declarations define the possible values contained in a certain value space (in our example above, the value space singplur_value contains the values sing and plur). Such a declaration would then look as follows:

feature_value_space (singplur_value, [[sing, plur]]).
• **feature** declarations define the features and associate a feature with its value space, which looks as follows:

```plaintext
feature (singplur, singplur_value).
```

• **category** declarations, define the category symbols and associate a category with the set of features it can take (the order in which the possible features are listed does not matter.) Such declarations are represented as follows:

```plaintext
category (np, [sem, singplur]).
```

• **top_level_category** declaration define the GSL "top-level grammar" which is used (Nuance Grammar Specification Language, which will be introduced in section 3.2.4). If the single top-level category is `.MAIN`, the declaration looks as follows:

```plaintext
top_level_category (`.MAIN`).
```

In order to give a better idea of what such a grammar might look like, we should take a look at the following example of a simple FG grammar\[^1\]

```plaintext
% Declarations
feature_value_space (singplur_value, [[sing, plur]]).
feature (singplur, singplur_value).

category (`.MAIN`, [gsem]).
category (np, [sem, singplur]).
category (spec, [sem, singplur]).
category (noun, [sem, singplur]).

top_level_category (`.MAIN`).
```

\[^1\] adapted from [Rayner et al. 2006]
% Grammar rules

'.MAIN' : [gsem = [value = NP]] -->
np : [sem = NP].

np : [sem = [[spec, S], [noun, N]], singplur = Singplur] -->
spec : [sem = S, singplur = Singplur],
noun : [sem = N, singplur = Singplur].

% Lexicon rules

spec : [sem = 1, singplur = sing] --> one.
spec : [sem = 2, singplur = plur] --> two.
noun : [sem = beer, singplur = sing] --> beer.
noun : [sem = coffee, singplur = sing] --> coffee.
noun : [sem = beer, singplur = plur] --> beers.
noun : [sem = coffee, singplur = plur] --> coffees.

This simple grammar would allow noun phrases such as "one beer", "two beers", "one coffee" and "two coffees" to be recognised. However, ungrammatical noun phrases, such as "one beers", "two beer", "one coffees" or "two coffee" would be rejected.

3.2.2.2 Nuance Grammar Specification Language (GSL)

We have now already seen the advantages and functions of Feature Grammars in contrast to Context Free Grammars. But in order to use the Nuance speech recognition package, which is used in Regulus, one final step is needed. After the abstract feature grammar is compiled into a context free grammar, it now has to be transformed into the Nuance Grammar Specification Language (GSL). The GSL is used to specify the grammar formally for a Nuance System application [Nuance 2001]. The GSL can be described as a version of CFG extended to allow construction of semantic values [Rayner et al. 2006].

We will now again take the example we have already seen for the feature grammars. If it were written in GSL, it would look as follows [Rayner et al. 2006:29]:

37
The semantic representation of utterances which can be recognised with the grammar above is of the following form:

\[
( ( \text{spec} \ < \text{number} > ) ( \text{noun} \ < \text{drink} > ) )
\]

Therefore the utterance "two beers" would result in the following semantic representation:

\[
( ( \text{spec} \ 2 ) ( \text{noun} \ \text{beer} ) )
\]

Since the CALL-SLT application is not written in GSL (this step is conducted automatically by Regulus), its functionality will not be explained further in this thesis; however, in Appendix A, a poem explaining the most important rules of GSL writing can be found.
3.3 Translation

Next to the speech recognition, the translation process is the other big issue in a CALL application, because it checks the semantic correctness of the user’s utterances. In the following sections we will first introduce the different approaches to translation and then introduce the approach used in the CALL-SLT project.

3.3.1 Translation systems and Interlingua

There are mainly two different approaches to translation systems in machine translation, which are on one hand a statistical and on the other hand a linguistic approach, which includes the interlingua-based approach as it has been used in the CALL-SLT project. These two approaches, as well as the interlingual approach will be introduced in the following sections.

3.3.1.1 Statistical Approach to Translation

The statistical approach to translation tries to work without formulating linguistic knowledge, by applying statistically or probabilistically based techniques to the machine translation. This approach works with two main notions which are the language model on the one side and the translation model on the other. The goal of the language model is to provide the probabilities for strings of words (most often described in either bigrams or trigrams, taking the probability of two, respectively three words being followed by each other) and the translation model gives information about the probability of a target sentence T to appear in a translation of a text, in which source sentence S appears [Arnold 2002:202].

The statistical approach therefore requires a method for computing language model probabilities, a method for computing translation probabilities, as well as a method for searching among possible source sentences for the one sentence that gives the greatest probability of correspondence [Brown et al. 1990]. [Brown et al. 1990] describes how these different components work: The source language model and the
translation model furnish a probability distribution over source-target sentence pairs \((S,T)\); the joint probability \(Pr(S, T)\) of the sentence pair is the product of the probability \(Pr(S)\) which is computed by the language model and the conditional probability \(Pr(T|S)\) which is computed by the translation model. The decoder is responsible for the actual translation: it finds a viable translation of a source sentence by choosing a target sentence from a large database whilst looking for the maximum probability \(Pr(S|T)\). In Figure 3.8 this process can be seen in a more abstract way.

Figure 3.8: Statistical approach to translation

However, in order to calculate the source language probabilities, a large amount of monolingual data is required. The downside of this approach is for this reason (as we have already seen for statistical speech recognition) that an extensive corpus is required. As far as the translation model is concerned, a large bilingual aligned corpus is needed, which is hard to obtain and very time-consuming to create [Arnold 2002:202]. The performance of statistical machine translation approaches are therefore very much dependent on the quality and size of the corpora used to calculate the probabilities.

3.3.1.2 Linguistic Approach to Translation

The linguistic approach to translation differs considerably from the statistical approach, because it works on the basis of linguistic knowledge. The foundation of this approach has been described by [Arnold 1994:71]:

40
High quality MT requires linguistic knowledge of both the source and the target languages as well as the differences between them.

The most important components of a linguistic MT are on the one hand an extensive grammar of both the source and the target language and on the other hand a comparative grammar which allows to relate a source sentence representation to a corresponding target language representation [Arnold 1994].

Figure 3.9 [Arnold 1994:72] shows the three major steps of a linguistic machine translation: in a first step the parser and the source language grammar analyse the source language input, in a second step the underlying representation of the source language sentence is transferred into an underlying representation of the corresponding target language sentence and in a third step the underlying representation of the target language sentence is transformed into a target language sentence with a generator and the target language grammar.

Figure 3.9: Linguistic approach to translation

The main advantage of this approach to translation is that a proper grammar of the target language is used which ensures the grammatical correctness of the system’s
3.3.1.3 Interlingual Approach to Translation

A particular kind of linguistic machine translation is the one based on interlingua. The goal of the interlingual approach to machine translation is to capture the commonness between a source and a target sentence which is the abstract meaning of the sentence. It therefore works with a ‘representation of meaning’ which, however, must be entirely language-independent, in order not to be linked either to the source or the target language. Such a language-independent representation of meaning can be obtained via a so-called Interlingua [Arnold 2002:80].

In Figure 3.10 [Arnold 2002:83] we can see how such an interlingual approach works. The source text is first analysed with the source language grammar and an interlingual representation is produced. The target language grammar then generates a target text from this interlingual representation.

![Interlingual approach to translation diagram]

**Figure 3.10:** Interlingual approach to translation
One interlingual representation can therefore have several realizations in natural language, which all have the same semantic value. It does not matter that the different sentences use different syntactic structures to express the meaning. This phenomenon can be seen in the example in Figure 3.11.

**Interlingua Surface Form:**

ASK-FOR POLITELY BEER

**Interlingua:**

\{
  \text{arg2}=[\text{drink, beer}],
  \text{null}=[\text{politeness, polite}],
  \text{null}=[\text{utterance type, request}]
\}

**Natural Language:**

I want a beer, please.
I would like a beer, please.
Could I have a beer, please?
Could you please bring me a beer?
etc.

*Figure 3.11: Interlingua example*

When working with an interlingual approach to translation, it is important to take account all the subtle differences between different languages. Such differences might include problems concerning non-isomorphisms (e.g. if one language has several ways to express one reality, whereas another language only has one), lexical holes (e.g. if a language does not know a certain reality and therefore has to paraphrase it) or non-compositionality (e.g. if the components of a source language sentence cannot be translated into the target language as they are).

But there are also purely grammatical issues which must be covered by the interlingua. These are of special interest in the CALL-SLT project, since learners should be able to concretely practice a grammatical phenomenon (e.g. the conditional tense). Hence why the temporal aspects of verbs must be included in the interlingua, in order for correct translations to be possible. The main problem with temporal aspects is that every language has its own model of expressing certain phenomena, as for
example continuity. The probably most prominent example is the present continuous in English. The German sentence "Ich trinke" could be expressed in two different ways in English, namely: "I drink" or "I’m drinking".

However, the use of an interlingua only makes sense in a limited domain because general language would demand for an infinite number of concepts and representations, which is practically unfeasable. But since the domain of our CALL application is well restricted to the restaurant domain, this should not pose any major problems. Some of the main problems when using an interlingua approach are the decision of the grain size of meaning to represent and the facets of meaning to include, the complexity of the interlingua (especially in projects taking place at multiple research sites, in which all researchers must be able to understand and work on the interlingua), as well as the distinction between general and domain specific components (so that the interlingua can be expanded to different domains) [Levin 2002].

3.3.2 Interlingua in CALL-SLT

In the CALL-SLT project an interlingual approach to the translation process is used, as it has been introduced above.

In this approach the source sentence (= the learner’s utterance) is first analysed with a specialised source language grammar for the speech recognition process, which results in the source representation of the sentence. The resolved source representation is then converted into its interlingual representation with so-called translation or transfer rules (cf. 4.5). It is this part of the process which is crucial to a successful translation game, because by means of this verification, the semantic correctness is checked. If the interlingua of the source representation and the interlingua of the initial prompt correspond, the translation is successful and the user’s utterance is judged as being correct.

This procedure is displayed in the following example of the sentence "I would like a beer":

44
As can be seen in the example above, the structural format of the interlingua in this project is a list of simple role-marked semantics, which are attribute-value pairs. This means that each semantic concept (e.g. ‘drink’ or ‘food’) has a value which corresponds to the concept (e.g. ‘beer’ or ‘pizza’).

Once a player starts a game session in CALL-SLT, the interlingua of a randomly chosen sentence from the corpus is transformed into a surface form with the interlingua grammar (cf. 4.6). This surface form has the aim of being abstract enough not to be linked too closely to natural language, but to be humanly readable at the same time. Such a surface form of the interlingua might look as follows: ASK-FOR POLITELY 1 BEER.

Another approach to the surface form of the interlingua is to depict the surface form with clip art pictures. This approach has the advantage that it completely detaches the initial prompt from natural language and is therefore a language-neutral realisation. An example of a picture prompt for the sentence "I would like a soup, please"
can be seen in Figure 3.1. For verbs which are difficult to be depicted with static icons, minimal animation can be introduced, in order to distinguish for example verbs like "to leave" and "to arrive". However, since there are going to be some icons which will be hard to understand at first sight, a backup will be retained which will give the student access to the interlingua prompt if needed. A first evaluation of this clip art approach to the interlingua has shown that female users prefer the linguistic interlingua surface form and find this easier to understand and male users prefer pictorial prompts [Tsourakis 2010].

3.4 Help

The third component of the CALL-SLT project (next to speech recognition and translation) is the help function. The goal of the help function is to provide example sentences, if a student does not know the solution to a prompt or if s/he simply is not sure how to pronounce certain words.

The help function consists of a written and a spoken part: the student can either simply read the help examples or has the option of listening to them.

The written help examples are collected from the initial corpus which contains multiple syntactic structures for one semantic representation. An example of possible examples aiding the student for the prompt ASK-FÖR POLITELY 1 BEER could look as follows in German:

- Ich möchte ein Bier.
- Ich hätte gerne ein Bier.
- Ich nehme ein Bier.
- Dürfte ich bitte ein Bier haben?
- Könnte ich bitte ein Bier haben?

The corresponding spoken help examples are wavfiles which were recorded by native-speakers beforehand. The user can choose between different voices (gender and age) in order to get as close to his/her own voice as possible.
However, the feedback system of the help function also has a drawback in that the system may not always correctly recognise what the learner uttered. It can therefore have a detrimental effect on the student’s learning by giving negative feedback or too much help, especially when the actual utterance was correct. In several studies, it has been demonstrated that it can be very frustrating for learners if they are corrected on accurate utterances. Our idea is thus to model the correct answer with an example from a native speaker which will, in turn, help nonetheless in terms of pronunciation even if the phrase was grammatically correct in the first place.

As far as the help function is concerned, there will be an important difference between the application prototype (which is described in this thesis) and the final version that will eventually be available online. In the prototype version, the help function works as described above. In the final version, however, the help examples will be recorded by other language learners during their ‘listening exercises’. This will have the advantage that many different dialects will be gathered and there will be a ‘giving and taking’ during the CALL-SLT translation game.

### 3.5 Interface

To top off this chapter on the CALL-SLT project, the functioning of the actual interface of the CALL-SLT prototype, which can be seen in Figure 3.12, will be described:

1. the student receives a prompt from the system

2. the student clicks on the "Speak" button and orally translates the prompt into the L2

3. the system displays the sentence as recognised by the program, as well as the interlingual representation of that sentence

4. if the interlingua does not correspond to the one of the prompt and the unhappy smiley emoticon is shown, the student can get help by either pressing the button with the lifebelt for written help or the speaker symbol to receive spoken help
5. once the translation is correct and the happy sun emoticon is shown, the student can progress to the next prompt by clicking on the button depicted as a book.

3.6 Conclusion

In this chapter we have seen how the CALL-SLT project functions in theory. More specifically, we have investigated the three main components of the project, namely, speech recognition with a linguistic approach, translation with an interlingual approach and the help function. Since this chapter has been dedicated to the theoretical principles of the project with a small number of examples to explain the various concepts, we will now analyse how the system works in reality. Chapter 4 will thus describe the practical part of the project which consists in developing and evaluating a CALL-SLT version for German.
4 Practical Section

The purpose of this fourth chapter is to give an insight into the rule development within the CALL-SLT project. My goal in the practical part of the project was to create and evaluate a version of the CALL-SLT application for the restaurant domain, which will enable English and French speakers to learn German. There are four major steps in the development: (1) to create a German corpus (section 4.1), (2) to write the grammar rules and lexicon entries (section 4.2 - 4.4) for which I concentrated on introducing German time structures, (3) to write the transfer rules (section 4.5) and (4) to write the German interlingua prompts (section 4.6).

4.1 Corpus

The corpus is a very important part of the CALL-SLT application for three reasons: first of all, the corpus defines the sentences of the translation game (cf. Figure 3.2), secondly it specialises the grammar for a specific domain (cf. 3.2.2) and finally it provides the examples for the help function (as described in chapter 3.2.6.) For all of these reasons, it is important that the corpus contains as many sentence structures as possible, as well as an extensive range of vocabulary.

The initial corpus was written in English, containing all kinds of sentences from the restaurant domain (e.g. "Could I have a beer please?", "I would like to book a table", "Where are the toilets?", etc.) At this point of time, the corpus is not extensive; however, new structures and vocabulary can be added on demand.

In order to create a corpus for German, the sentences were translated and all possible structures were added in German, which are interesting for the language learners. Since the main focus was on time structures, the following examples demonstrate how the corpus entries appear in relation to German temporal structures:

sent('ich möchte einen tisch um sieben uhr').

sent('ich möchte einen tisch um fünf nach sieben').
sent('ich möchte einen tisch um sieben uhr fünf').
sent('ich möchte einen tisch um zehn nach sieben').
As can be seen in the above examples the different possible sentence structures are represented in the format `sent('<Utterance>')`.

The entire training corpus, which is made up of 511 sentences, with a significant number of restaurant vocabulary items and a variety of different sentence structures can be seen in appendix B.1.1.

### 4.2 Lexicon

The lexicon file contains all words which have to be recognised by the speech recognition mechanism. In order to see how this vocabulary is written, we will now take a look at a noun entry and a verb entry.

The entry for the noun *Tagesgericht* (English: dish of the day) looks as follows in the German lexicon:
The @ sign signifies that the entry is linked to a macro, a Regulus function which allows to write rules in a more compact way [Rayner et al. 2006] that is saved in another file and is applied here. In tables 4.1 and 4.2 we can see the different macros that this lexicon entry invokes, as well as their respective forms for the noun *Tagesgericht*.

<table>
<thead>
<tr>
<th>Macro</th>
<th>Form</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Macro 1</td>
<td>Nominative</td>
<td><em>Tagesgericht</em></td>
</tr>
<tr>
<td>Macro 2</td>
<td>Genitive</td>
<td><em>Tagesgerichts</em></td>
</tr>
<tr>
<td>Macro 3</td>
<td>Dative</td>
<td><em>Tagesgerichts</em></td>
</tr>
<tr>
<td>Macro 4</td>
<td>Accusative</td>
<td><em>Tagesgericht</em></td>
</tr>
<tr>
<td>Macro 5</td>
<td>Plural Nominative</td>
<td><em>Tagesgerichte</em></td>
</tr>
<tr>
<td>Macro 6</td>
<td>Plural Genitive</td>
<td><em>Tagesgerichts</em></td>
</tr>
<tr>
<td>Macro 7</td>
<td>Plural Dative</td>
<td><em>Tagesgerichts</em></td>
</tr>
<tr>
<td>Macro 8</td>
<td>Plural Accusative</td>
<td><em>Tagesgerichte</em></td>
</tr>
</tbody>
</table>

Table 4.1: Macros for singular nouns

In this macro, we first have the noun in its eight different cases in German (singular nominative, singular genitive, singular dative, singular accusative, plural nominative, plural genitive, plural dative, plural accusative), followed by the gen-

51
Table 4.2: Macros for plural nouns

der of the noun which is defined as follows: ntr = neutral, fem = feminine, masc = masculine. In the square brackets we define the semantic values of the noun ([gericht, tagesgericht]). The next thing to be declared is the sortal type of the noun (sub-
stance) and then the kind of determiner the noun can take is declared which is either a definite article (def), an indefinite article (indef) or a numeric value (numeric).

The last declared feature, the OfMod feature, concerns the genitive complement which has the value none in our example since the noun Tagesgericht does not have such a complement.
We will now also take a look at a sample verb entry from the German lexicon which appears as follows:

@verb([möchte, möchtest, möchtet, möchten, möchten, möchten, möchten, gemocht], [verb,möchten], agent, (nx0vnx1\modal), haben, [objSemNType=(loc\substance), ppSemPPType=(loc\time\hour)]).

This verbal entry again invokes a number of macros which define the features with which the verb is associated. The macro for the verb in its present singular form for the first person ("möchte") is represented as can be seen below and all other macros behave the same way as has been shown in tables 4.1 and 4.2 for the noun Tagesgericht.

macro(verb([Pres1Sg, Pres2Sg, Pres3Sg, Pres1Pl, Pres2Pl, Pres3Pl_2Pol, Inf, PerfPart], Sem, SubjSem, Subcat, AuxV, Feats), (v:[sem=concat([Sem], [[tense, present]]), agr=(1\sing), subjSemNType=SubjSem, subcat=Subcat, vform=indikativ, auxV=AuxV, voice=active, prefix=(y\n) | Feats] --> Pres1Sg)).

In the entry we can see that we first define the verb in its crucial grammatical tenses in square brackets (PresentSingular 1st_person, PresentSingular 2nd_person, PresentSingular 3rd_person, PresentPlural 1st_person, PresentPlural 2nd_person, PresentPlural 3rd_person, infinitive, perfect_participle). We then define the semantic representation of the verb in another pair of square brackets [verb, möchten]. The feature subjSemNType declares the semantic value of the subject, which is in this case the agent and the feature subcat defines the kind of verb which can in the case of möchten either be a modal verb (modal) as in "John must sleep" or a simple transitive verb (nx0vnx1) as in "John loves Mary". The entry haben defines the auxiliary verb with which the verb is used ("haben" or "sein"). Then the sortal types objSemNType which defines the sortal type of the direct object the verb can take and which in this case is either a location (loc) or a substance and ppSemPPType which defines the sortal type of the prepositional complement the verb can take and
which in this case is either of the type location, time or hour are declared.

4.3 Time Structures

The main part of this practical section of my thesis was writing the grammar rules for time structures in German. Since it is very time-consuming to write a new grammar from scratch, I made use of the German grammar which has been written for the MedSLT Project by Johanna Gerlach. The goal of the MedSLT Project was to create an application which helps facilitate the communication between patient and doctor, if the two interlocutors do not speak the same language (for a more detailed description of the project see [Rayner et al. 2008]). Since the MedSLT German grammar was a good basis for CALL-SLT, I used it as the framework and adjusted it accordingly to suit the requirements of CALL-SLT which means that some entries were added that are specific to the restaurant domain. One important change to the grammar was the addition of time structures, which will be described in this section. Another crucial point was to add some new sentence structures, which will be introduced in the following section (4.4).

There are four rules which were written to cover all the possible German time structures which can be seen in the corpus in section 4.1. These rules will now be described in detail in the following four sections.

4.3.1 At 1 to 24 o’clock

The first of these rules covers the full hour, this means that it covers structures, such as "at 1-12 o’clock" (German: "um 1-24 Uhr"). The grammar rule for this structure can be seen below:

```plaintext
pp: [sem=@pp_p_np_sem_number(P, Num, Np), sem_pp_type=hour, sc=n, gapsin=null, gapsout=null] -->
prep: [sem=P, sem_pp_type=time, sem_n_type=uhr, np_has_spec=n],
number: [sem=Num, agr=Agr, num_type=stunde_amstunde_pm],
np: [sem=Np, agr=(3/\ sing), kasus=nom, sem_n_type=uhr, gender=fem, gapsin=null, gapsout=null].
```
This rule describes a prepositional phrase (pp) with the following components:

- **prep**: a preposition –> [um]
- **number**: a number –> [1-24]
- **np**: a noun phrase (which is always singular) –> [Uhr]

The preposition **prep** is defined by a number of different features: The **sem**-feature determines the semantic value of the preposition and the **sem_pp_type**-feature characterises the sortal type of the prepositional complement. In this case, the prepositional complement **sem_pp_type** has the value **time**. The **sem_n_type**-feature expounds the sortal type of the noun that is used with the preposition which has the value **uhr** in this case and the last feature of the preposition is the **np_has_spec**-feature which declares whether the noun phrase that is used with the preposition must have an article (as in **bei**) or not (as in **beim**).

The number (**number**) is also defined by a list of features. The first one being the **sem**-feature which again determines the semantic value of the number. The **agr**-feature, which gives information about the agreement, is not specified here because **uhr** is always singular. The **num_type**-feature defines the type of number, which in this case is **stunde_am**, as well as **stunde_pm**. This means that in German we can say "at seven o'clock" ("um sieben Uhr"), as well as "at 19:00" ("um neunzehn Uhr"), owing to the fact that both of these time structures are used in everyday life in German. **stunde_am** is therefore defined in such a way that it includes numbers one to twelve and **stunde_pm** includes numbers thirteen to twenty-four.

The noun phrase **np** is defined by the following features: the semantic value **sem** (as has already been explained for the preposition and the number) and the **agr**-feature which defines that the noun is in the third person singular. The **kasus**-feature gives information about the case of the noun, which is nominative (**nom**) in this case. The **sem_n_type**-feature defines the sortal type of the noun, which is **uhr** for this rule. The feature **gender** gives information about the gender of the noun which has the value feminine (**fem**).
We can also see how a macro is used inside this rule, namely \texttt{@pp\_p\_np\_sem\_number}. This macro is written in a separate file (\texttt{role\_marked\_sem.regulus}) and is then invoked inside this rule. The macro looks as follows:

\begin{verbatim}
macro(pp_p_np_sem_number(P, Num, Np),
  [[role, P, concat(Np, [[plus, [Num, 0]]]]])).
\end{verbatim}

This macro has been adopted from the French grammar in order to facilitate the transfer rules. It serves to derive the semantics of prepositional phrases \textit{pp}. The prepositional phrase "\textit{um fünf Uhr}" (English: "at five o'clock") would then be represented as follows: [(\text{um}=[\text{zeit}, \text{uhr}]), (\text{um}=[\text{plus}, [5,0]])].

4.3.2 At 1 to 59 minutes past 1 to 24 o'clock

In the second rule, minutes were added to the first rule so that constructions such as "\textit{um sieben Uhr zehn}" (English: "at seven ten") are covered. The rule is actually the same as the first one except that the semantic is different, because it includes an additional element for minutes and that after the noun phrase we have another number constituent which is for minutes. Below we can see the second rule:

\begin{verbatim}
pp: [sem= @pp\_p\_np\_sem\_number\_et(P, N1,Np,N2), sem_pp_type=hour,
  gapsin=null, gapsout=null] -->
prep: [sem=P, sem_pp_type=time, sem_n_type=uhr, np_has_spec=n],
  number: [sem=N1, agr=Agr, num_type=stunde_am\slash stunde_pm],
  np: [sem=Np, agr=(3\slash sing), kasus=nom, sem_n_type=uhr, gender=fem,
  gapsin=null, gapsout=null],
  number: [sem=N2, num_type=minute].
\end{verbatim}

This rule again describes a prepositional phrase \textit{pp} which is comprised of the following components:

- \textit{prep}: a preposition $\rightarrow$ [\textit{um}]
- \textit{number}: a first number for hours $\rightarrow$ [1-24]
- \textit{np}: a noun phrase $\rightarrow$ [\textit{Uhr}]
- \textit{number}: a second number for minutes $\rightarrow$ [1-59]
The first thing that differs from the former rule is the second number which has two features: the sem-feature and the num_type-feature. In contrast to the first number which can only take numbers with the value of full hours (stunde_am and stunde_pm), the second number is limited to minute values (minute). In order to distinguish between the numbers, all numbers from one to fifty-nine have the value minute.

The second difference, in contrast to the first rule, is the macro of the semantic value of the prepositional phrase which has again been adopted from the French grammar and appears as follows:

\[
\text{macro(pp_p_np_sem_number_et(P, N1, Np,N2),}
\]

\[
\quad [[\text{role, P, concat(Np,[[plus, [N1, N2]]])}}]).
\]

This macro serves to derive the semantics of prepositional phrases pp. The prepositional phrase "um sieben Uhr zehn" (English: "at 7:10") would then be represented as follows: \([(\text{um}=[\text{zeit,uhr}]), (\text{um}=[\text{plus,}[7,10]])]\).

4.3.3 At 5–25 past 1–12 o'clock, at a quarter past 1–12 o'clock

The third rule is a little bit more complicated than rules number one and two. There are different structures which are covered by this rule, namely the following:

- ✓ um fünf nach 1-12 ([AT 12:05])
- ✓ um zehn nach 1-12 ([AT 12:10])
- ✓ um fünfzehn nach 1-12 ([AT 12:15])
- ✓ um viertel nach 1-12 ([AT 12:15])
- ✓ um zwanzig nach 1-12 ([AT 12:20])
- ✓ um fünfundzwanzig nach 1-12 ([AT 12:25])

The pp-rule consists of the following components:
- **prep**: a preposition -> \[um\]
- **mod_time_et**: a constituent for the structure \(x \text{ past } y\) -> \([5, 10, 15, 20, 15, \text{ viertel nach}]\)
- **number**: a number for hours -> \([1-12]\)

The following example depicts what the third rule looks like:

```plaintext
pp: [sem= @pp_p_np_sem_number_et(P, N, MT), sem_pp_type=hour, gapsin=null, gapsout=null] -->
  prep: [sem=P, sem_pp_type=time, sem_n_type=uhr, np_has_spec=n],
  mod_time_et: [sem=MT],
  number: [sem=N, agr=Agr, num_type=stunde_am].
```

This rule differs from the previous two rules as it includes the constituent **mod_time_et** which allows to have a time specification that includes "x nach", as in "um fünf nach sieben" (English: "at five past seven"). This construction has again been adopted from the French grammar which has similar constituents for the structure "et quart".
included in the rule. The answer is simply that in German, it is not usual to use more
detailed time structures than 5-minute-steps in the restaurant domain. However, if
we wanted to adjust this rule for a transport domain, for example, in which very
precise time structures are imperative, the rule can be adjusted to look something
as follows:

```
pp: [sem= @pp_p_np_sem_number_et(P, N1, N2), sem_pp_type=hour,
gapsin=null, gapsout=null] -->
  prep: [sem=P, sem_pp_type=time, sem_n_type=uhr, np_has_spec=n],
  number: [sem=N1, agr=Agr, num_type=minute],
  nach,
  number: [sem=N2, agr=Agr, num_type=stunde_am].
```

4.3.4 At 5–25 to 1–12 o’clock, at a quarter to 1–12 o’clock

The fourth and final rule is similar to the third rule; however, it includes a structure
to say "x minutes to (instead of past) y hours" and covers the following structures:

✓ um fünf vor 1-12 ([AT 11:55])
✓ um zehn vor 1-12 ([AT 11:50])
✓ um fünfzehn vor 1-12 ([AT 11:45])
✓ um viertel vor 1-12 ([AT 11:45])
✓ um zwanzig vor 1-12 ([AT 11:40])
✓ um fünfundzwanzig vor 1-12 ([AT 11:35])
✓ um fünf nach 1-12 ([AT 11:35])
✓ um fünf vor halb 1-12 ([AT 11:25])

The fourth rule is displayed as follows,

```
pp: [sem= @pp_p_np_sem_number_moins(P, N,MT), sem_pp_type=hour,
gapsin=null, gapsout=null] -->
  prep: [sem=P, sem_pp_type=time, sem_n_type=uhr, np_has_spec=n],
  mod_time_moins: [sem=MT],
  number: [sem=N, agr=Agr, num_type=stunde_am].
```

and it consists of the following components:
- prep: a preposition $\Rightarrow$ [um]
- mod_time_moins: a constituent for the structure $x$ to $y$ $\Rightarrow$ [5, 10, 15, 20, 25, viertel vor, fünf vor halb, fünf nach halb]
- number: a number for hours $\Rightarrow$ [1-12]

The constituent $x$ to $y$ is defined in the lexicon by the macro mod_time_moins, which contains the following information:

macro(mod_time_moins(Surface, N), ( mod_time_moins:[sem=N] $\Rightarrow$ Surface )).

As we have already seen for the third rule, this macro is called in the lexicon as can be seen here:

@mod_time_moins((fünf,vor), 5).
@mod_time_moins((zehn,vor), 10).
@mod_time_moins((fünfzehn,vor), 15).
@mod_time_moins((viertel,vor), 15).
@mod_time_moins((zwanzig,vor), 20).
@mod_time_moins((fünfundzwanzig,vor), 25).
@mod_time_moins((halb), 30).
@mod_time_moins((fünf,vor,halb), 35).
@mod_time_moins((fünf,nach,halb), 25).

Besides the differing macro, we can again see the same phenomena as in rule three, namely that the hours are restricted to 1-12 (stunde_am) in number and that the noun phrase np is again not included in the rule, both for the same reasons explained above for rule three (4.3.3).

There are two special cases in German time structures, which are "um fünf vor halb" (= "at five minutes to half past") and "um fünf nach halb" (= "at five minutes past half past"). An easy way to cover both these two structures in the grammar without having to write an extra rule is to include them in the fourth rule. I decided to treat these cases as lexical units the same way "um viertel vor" (English: "at a quarter to") was handled. They are described with the mod_time_moins macro and are linked to their corresponding equivalents, namely "um fünf vor halb acht" = "um fünfunddreissig vor acht" and "um fünf nach halb acht" = "um fünfundzwanzig vor acht". 
acht". In the interlingua representation, this would then be 19:25 or 19:35 respectively.

4.4 New Sentence Structures

In order to cover a number of possible utterances for the same interlingua prompt, it was necessary to write some new sentence structures. The most important sentence structures which had to be added to the MedSLT grammar were declarative sentences, prepositional phrases for persons and sentences with two prepositional phrases. In order to highlight how this has been done, I will introduce a concrete example, the sentence "Ich möchte einen Tisch für vier" (English: "I would like a table for four").

To get a first idea of how this sentence is constructed, we will analyse the following parse tree:

```
.MAIN [GENERAL_GERMAN:212-219]
  med_utterance [GENERAL_GERMAN:227-232]
    / optional_interj null [GENERAL_GERMAN:941-943]
    | s [GENERAL_GERMAN:263-274]
    | / np [GENERAL_GERMAN:741-749]
    | | pronoun lex(ich) [GENERAL_GERMAN_LEX:207-209]
    | | / vp [GENERAL_GERMAN:358-371]
    | | | / v lex(möchte) [CALLSLT_LEX:1-4]
    | | | optional_adverb null [GENERAL_GERMAN:932-934]
    | | | np [GENERAL_GERMAN:706-710]
    | | | / d [GENERAL_GERMAN:715-719]
    | | | | number lex(einen) [GENERAL_GERMAN_LEX:282-283]
    | | | | nbar [GENERAL_GERMAN:785-788]
    | | | | n lex(tisch) [CALLSLT_LEX:153-157]
    | | | / prep lex(für) [GENERAL_GERMAN_LEX:542-543]
    | | | | np [GENERAL_GERMAN:711-714]
    | | | | nbar [GENERAL_GERMAN:789-792]
    | | | | number lex(vier) [GENERAL_GERMAN_LEX:293-293]
    | | optional_pp null [GENERAL_GERMAN:906-908]
  optional_interj null [GENERAL_GERMAN:941-943]
```

In the parse tree above, the lexical entries are highlighted in red to make it easier to see the rule of which they are part.
Declarative Sentence

The chosen sentence shows the constituents of a declarative sentence structure. We can easily recognise that the two main components in this kind of sentence are a noun phrase and a verb phrase. The optional interjection, which is not used in this sentence (value = null), would allow us to insert a polite bitte to the phrase, as in "Ich möchte einen Tisch für vier, bitte." (English: "I would like a table for four, please.")

The noun phrase (np) rule defines the pronoun *ich*. As can be seen in the np rule which is displayed below, it only consists of a pronoun, as it is required for the sentence.

np: [sem=Pro, agr=Agr, gender=Gender, sem_n_type=Type, takes_post_mods=n, n_of_mod_type=none, syn_type=pronoun, kasus=k, gapsin=Gin, gapsout=GIn] --> pronoun: [sem=Pro, agr=Agr, gender=Gender, sem_n_type=Type, kasus=k].

The verb phrase (vp) rule consists of a vbar, a np (the object), as well as two optional_pp rules. This vp rule can be seen below and its components will be looked at in more details afterwards.

vp: [sem=@vp_v_pp_np_sem(V, Pp, Np, OptPp), agr=Agr, vform=VF, subj_sem_n_type=SubjSem, inv=I, part_type=Part, aux_v=AuxV, is_aux_v=none, voice=Voice, v_final=n, gapsin=GIn, gapsout=QOut, präfix=n] -->
  vbar: [sem=V, agr=Agr, inv=I, vform=VF, vform=(\(partizip\)), subcat=nx0vnx1, subj_sem_n_type=SubjSem, obj_sem_n_type=ObjSem, pp_sem_pp_type=PPSem, aux_v=AuxV, is_aux_v=none, part_type=Part, voice=Voice, präfix=n, gapsin=null, gapsout=null],
  np: [sem=Np, sem_n_type=ObjSem, kasus=akk, gapsin=GIn, gapsout=QNext],
  optional_pp: [sem=Pp, sc=n, sem_pp_type=PPSem, sem_pp_type=(\(time\ passive\ by\ frequency\)), gapsin=QNext, gapsout=QOut],
  optional_pp: [sem=OptPp, sc=y, gapsin=null, gapsout=null].

We will now first take a look at the vbar rule which rewrites the vbar as a verb:
This rule is characterised by the fact that the \textit{vbar} consists of a conjugated verb (v) and an \textit{optional\_adverb} which, however, is not required for the sentence. This is why in the parse tree the \textit{optional\_adverb} has the value \textit{null}.

Next comes a noun phrase (np) which consists of a determiner (d) and a nbar rule. Since a number as a determiner is needed in the sentence, the following rule is used which defines that a number can be used as a determiner:

\begin{verbatim}
d: [sem= @spec_number_sem(N), agr=Agr, det_type=numerical, prenumber=n, gender=Gender, kasus=Kasus] --> number: [sem=N, agr=Agr, gender=Gender, kasus=Kasus].
\end{verbatim}

The nbar rule used in the noun phrase consists of a simple noun (n), in this case \textit{Tisch}, which can be seen below:

\begin{verbatim}
nbar: [sem= @nbar_n_sem(Sem), agr=Agr, gender=Gender, sem_n_type=NType, n_of_mod_type=ModType, takes_det_type=DetType, kasus=K, num_type=none] --> n: [sem=Sem, agr=Agr, gender=Gender, sem_n_type=NType, n_of_mod_type=ModType, takes_det_type=DetType, kasus=K].
\end{verbatim}

Prepositional Phrase for People and two Prepositional Phrases

As has been seen in the vp rule, there are two optional prepositional phrases. In this example sentence, we only need one of these prepositional phrases, hence why the value of the second one is \textit{null}.

Below we can see the pp rule which is used for the \textit{optional\_pp}. It consists of a preposition (prep) and a noun phrase (np):

\begin{verbatim}
pp: [sem= @pp_sem(Sem), agr=Agr, subcat=SC, vform=VF, inv=n, part_type=Part, aux_v=AuxV, is_aux_v=IsAux, voice=Voice, präfix=Pr],
optional_adverb: [sem=Advp].
\end{verbatim}
The noun phrase (np) is described in the following nbar rule, which stipulates that it only consists of a number. This allows us to have the sentence "Ich möchte einen Tisch für vier" and not only structures with full np's as in "Ich möchte einen Tisch für vier Personen".

Of course the vocabulary has also to be declared. This is done in the lexicon file (callslt_lex.regulus), as has been described in chapter 4.2.

4.5 Transfer Rules

In chapter three the role of transfer or translation rules has already been mentioned; they are important in order to map the source representation with the interlingua and vice versa.

There are two major functions for which the transfer rules are essential, which are namely to verify the semantic correctness of the learner’s utterance on the one hand and to produce a correct sample sentence for the help function on the other hand.

Since this procedure to produce help sentences has not been discussed in the thesis (as it will only be introduced in future versions of CALL-SLT) this kind of transfer rules – so-called reverse transfer rules – will only be explained shortly in this chapter.

Figure 4.1 displays how transfer rules map the source representation with the interlingua to verify the semantic correctness of the student’s utterance which is a crucial step in CALL-SLT and in Figure 4.2 can be seen how reverse transfer rules map the interlingua with the source representation to produce a sentence for the help function.
There are two major kinds of transfer rules [Rayner et al. 2006]:

- **transfer_lexicon**
- **transfer_rule** and **reverse_transfer_rule**

**transfer_lexicon** rules specify lexical correspondences. They map attribute-value pairs one-to-one, as can be seen in the following examples:

\[
\text{transfer_lexicon}([\text{getränk}, \text{bier}], [\text{drink}, \text{beer}]).
\]
\[
\text{transfer_lexicon}([\text{gericht}, \text{hamburger}], [\text{food}, \text{hamburger}]).
\]

These **transfer_lexicon** rules simply link the source semantic values for *Bier* and *Hamburger* to their interlingua equivalents (*beer* and *hamburger*).

**transfer_rules** and **reverse_transfer_rules** on the other hand, are used for structural transfers. This kind of rule is needed when one-to-one mapping does not work because of the syntactic divergence between languages. These rules map a list of attribute-value pairs to another list of attribute-value pairs, as can be seen in the following example:
transfer_rule([[utterance_type, dcl], [pron, ich], [verb, möchten],
    [tense, present]], [null=[politeness, polite],
    null=[utterance_type, request]]).

This transfer_rule maps the interlingua structure polite request to the German
equivalent of a declarative utterance of the first person singular with the verb ‘möchten’
in its present tense or, in other words, a structure of the following kind: Ich möchte.

4.5.1 Source language to Interlingua

In this section, we will explore how the semantic correctness of the language learner’s utterances are verified by CALL-SLT through the transfer rules. The functioning of these transfer rules will now be explained by means of some specific examples.

We first have a rule which invokes the macro and looks as follows:

@n_oclock_to_n_plus_12(7, 19).

The invoked macro creates the transfer rule for time structures as have been described in section 4.3.2 (e.g. "um sieben Uhr fünf"; English: "at seven oh five"). This macro can be seen below:

macro(n_oclock_to_n_plus_12(N, NPlus12),
    transfer_rule([[plus, [N, Mins]], [zeit,uhr]],
    [[time, [NPlus12, Mins]]]))).

In concrete terms, this macro creates a transfer rule that maps the source representation of the time structure "um sieben Uhr fünf" which is displayed as [[um=[plus, [7, 5]]], (um=[zeit,uhr])] with the interlingua which is displayed as [[._29722=[time, [19,5]]]]. The transfer rule would then look as follows:

transfer_rule([[plus, [7, 5]], [zeit,uhr]],
    [[time, [19, 5]]]).

If the student says "I would like to book a table at seven oh five", the interlingua result will then be BOOK POLITELY 1 TABLE 19:05.
The macro for the transfer rule of time structures as have been introduced in section 4.3.3 (e.g. "um fünf nach sieben") looks similar, with the only difference being that \texttt{zeit,uhr} is not included. This macro creates the transfer rule that maps the source representation \((\text{um}=[\text{plus},[7,5]])\) to the interlingua \((\_205628=[\text{time},[19,5]])\).

\begin{verbatim}
macro(n_oclock_to_n_plus_12(N, NPlus12),
    transfer_rule([[plus, [N, Mins]]],
        [[time, [NPlus12, Mins]]])).
\end{verbatim}

The macro that creates the transfer rules for time structures as have been discussed in section 4.3.4 (e.g. "um fünf vor sieben") appears as follows:

\begin{verbatim}
macro(n_oclock_to_n_plus_12_minus_1(N, NPlus12Minus1),
    transfer_rule([[moins, [N, Mins]]],
        [[time, [NPlus12Minus1, @mins_to(Mins)]]])).
\end{verbatim}

Via this macro the transfer rule maps the source representation \((\text{um}=[\text{moins},[7,5]])\) with the interlingua \((\_191686=[\text{time},[18,55]])\) and it again links the 12-hour-format time specifications to their equivalent 24-hour-format. However in this example it is deducted by one, in order to map "at five to seven" to the 24-hour equivalent of "at 18:55". The macro for \texttt{NPlus12Minus1} is applied as can be seen below:

\begin{verbatim}
@n_oclock_to_n_plus_12_minus_1(7, 18).
\end{verbatim}

The macro \texttt{mins_to} makes it possible to transform so-called "to-minutes" as in "five to seven" into "past-minutes" as in "55 past six". An example of which can be seen here:

\begin{verbatim}
macro(mins_to(5), 55).
\end{verbatim}

Hence why the source utterance "I would like a table at five to seven" would be mapped to the following interlingua: \texttt{BOOK POLITELY 1 TABLE 18:55}, since "seven" is automatically transformed to "18:00" and "five to" is automatically transformed to "55 past".
4.5.2 Interlingua to source language

Transfer rules which act from the interlingua to the source language (so-called reverse transfer rules) differ somewhat from the transfer rules which have been described in 4.5.1. Instead of linking each possible structure to an interlingua representation via a separate transfer rule as can be seen in Figure 4.1, they link the interlingua representation to one source language structure, which then produces a semantically correct example sentence for the help function (cf. Figure 4.2).

An example of a reverse transfer rule which maps the prompt AKS-FOR POLITELY – which is displayed as [(null=[politeness,polite]),(null=[utterance_type,request]]) in the interlingua representation – with the source language sentence "Ich möchte" – which is in the source representation displayed as [(null=[tense,present]),(null=[utterance_type,dcl]),(null=[verb,möchten]),(null=[voice,active]),(subj=[pron,ich])] – can be seen here below:

reverse_transfer_rule([[utterance_type, dcl], subj=[pron,ich], [verb,möchten], [voice,active], [tense, present]], [null=[politeness, polite], null=[utterance_type, request]]).

4.6 Interlingua Surface Form

The interlingua surface form is an important element of the CALL-SLT project because it allows the system to link the interlingua to a human-readable form which can then be presented to the student [Rayner et al. 2010]. The advantage of these interlingua surface form prompts is that they are abstract enough that they are not too closely linked to the learner’s L1 (cf. 3.3.2).

There is a special file (german_surface_syntax.regulus) in which an atom of the interlingua, e.g. ([politeness, neutral_greeting]), is linked to a surface form which is presented to the student when s/he asks for a new prompt. An example of this can be seen here below:

politeness:[sem=[[politeness, neutral_greeting]]] --> @l('NEUTRAL-GREETING').

macro(l('NEUTRAL-GREETING'), 'GUTEN_TAG').
This macro says that the atom ([politeness, neutral_greeting]) whose surface form is \textit{NEUTRAL-GREETING} in English will become \textit{GUTEN_TAG} in German.

For the interlingua surface form, the English file has again been adapted by simply translating the English prompt into a German surface prompt. In this file, we can also define the differing word order of the surface form for each language. Since the German surface form has been linked to the English one, there are some differences which must be adjusted. In the following example, we can see how such an inversion in the word order is done:

macro(time_of_day_date(Time, Date), (Date,Time)).

We can see that in English the prompt would be something like \textit{BOOK TABLE JUNE_12 7:45} and in German the prompt would then look as follows: \textit{RESERVIEREN TISCH 7:45 12_JUNI}.

4.7 Conclusion

In this chapter, we have explored the various steps which are necessary to write a functional German CALL-SLT system. Besides the actual grammar rules, it is also important to declare lexicon entries and to have a corpus which covers the various structures. The interlingua surface form is also an important component of the CALL-SLT application as it creates human-readable interlingua prompts which are not too closely linked to the learner’s L1. Finally the transfer rules are essential to compare the interlingua of the uttered sentence with the interlingua of the initial prompt in order to be able to control the semantic correctness of the user’s utterance. In the next chapter, we will test and evaluate the CALL-SLT application with the underlying grammar which has been described in the course of this chapter.
5 Evaluation

In the final chapter of my Master’s thesis, I will present the results of a small evaluation which was conducted as a part of this project. There are two different evaluations which both test another version of the CALL-SLT application. The first one which is an evaluation of the first prototype of CALL-SLT for the language pair French (L1) - English (L2) was executed in November 2009. It will be described in section 5.1. The second evaluation was carried out in May 2010 and focuses on the German system developed for this thesis, therefore a prototype for the language pair French (L1) - German (L2) was tested. This second evaluation will be the topic of section 5.2.

Both evaluations were aimed to find out how satisfied the users are concerning a number of points (e.g. prompts, domain, help function, etc.), as well as the performance of the CALL-SLT application and the effect on the learners progress.

5.1 Evaluation of CALL-SLT for L2 = English

This first evaluation was conducted quite early on in the project. A first prototype has been tested by around 500 francophone collège students in the context of the open door days for collège students at the University of Geneva – one of the many events organised for the 450th anniversary of the university.

The exercise was divided into three parts: the students first listened to a presentation in which some basic knowledge about CALL and the CALL-SLT project was given. The last part of the presentation was dedicated to the CALL-SLT application, where the students were introduced to the prototype that they were about to test. The functionality and the meta command buttons were explained to them so that they were aware of how the application worked and how they had to proceed. The second part of the exercise consisted in the students’ testing the prototype CALL-SLT program for the restaurant domain which can be compared to the one developed in this thesis for German. Since the collège students all had French as their mother tongue, they tested a prototype through which they could learn English. The third
part of the workshop was dedicated to the students’ feedback in which they filled out a questionnaire with their personal information, as well as another questionnaire with questions concerning the CALL-SLT translation game.

Due to the large number of students, there was a high level of noise at all times, which negatively influenced the test results.

The personal information included questions about their age and gender, their previous language knowledge as well as their ability to work in an electronic environment. The students were in average between 16 and 18 years old. 83% of all participants were female. Only small percentages of the students either spoke English at home (6%) or have lived in an English-speaking country before (11%). Most of those testing the system had a basic knowledge of English (70% said that their English is at an intermediate level), 15% had an advanced level of English and 15% were beginners.

About two thirds of the students stated that they feel at ease when working with computers which facilitates the use of the application.

This personal information about the participants is important as it gives a good impression of the testers background.

The second questionnaire has been evaluated according to the language level that they indicated in the first questionnaire.

In Table 5.1, we can see the results of the beginners’ questionnaires:

<table>
<thead>
<tr>
<th>Beginner (15% of total)</th>
<th>++</th>
<th>+</th>
<th>=</th>
<th>–</th>
<th>—</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 The game was too difficult</td>
<td>2%</td>
<td>9%</td>
<td>45%</td>
<td>36%</td>
<td>8%</td>
</tr>
<tr>
<td>2 The game was too easy</td>
<td>3%</td>
<td>6%</td>
<td>38%</td>
<td>50%</td>
<td>3%</td>
</tr>
<tr>
<td>3 I enjoyed learning English with the game</td>
<td>23%</td>
<td>37%</td>
<td>26%</td>
<td>10%</td>
<td>4%</td>
</tr>
<tr>
<td>4 The game helped me improve my English</td>
<td>1%</td>
<td>33%</td>
<td>42%</td>
<td>16%</td>
<td>8%</td>
</tr>
<tr>
<td>5 I would use this game again if I could</td>
<td>11%</td>
<td>27%</td>
<td>44%</td>
<td>14%</td>
<td>4%</td>
</tr>
<tr>
<td>6 I would use a similar game which covered another topic</td>
<td>11%</td>
<td>27%</td>
<td>44%</td>
<td>9%</td>
<td>9%</td>
</tr>
<tr>
<td>7 I would recommend the game to my friends</td>
<td>14%</td>
<td>35%</td>
<td>36%</td>
<td>10%</td>
<td>5%</td>
</tr>
<tr>
<td>8 I would prefer a game which focused only on vocabulary</td>
<td>1%</td>
<td>10%</td>
<td>32%</td>
<td>50%</td>
<td>7%</td>
</tr>
<tr>
<td>9 I would prefer a game which was more interactive</td>
<td>3%</td>
<td>32%</td>
<td>41%</td>
<td>23%</td>
<td>1%</td>
</tr>
<tr>
<td>10 It was easy to understand what I was meant to say</td>
<td>13%</td>
<td>33%</td>
<td>36%</td>
<td>12%</td>
<td>6%</td>
</tr>
<tr>
<td>11 The help system helped me find what I was meant to say</td>
<td>50%</td>
<td>27%</td>
<td>17%</td>
<td>5%</td>
<td>1%</td>
</tr>
<tr>
<td>12 The help system helped me find what I had mispronounced</td>
<td>20%</td>
<td>23%</td>
<td>31%</td>
<td>18%</td>
<td>8%</td>
</tr>
<tr>
<td>13 It was easy to get the machine to understand me</td>
<td>1%</td>
<td>18%</td>
<td>26%</td>
<td>29%</td>
<td>26%</td>
</tr>
</tbody>
</table>

Table 5.1: Results of beginners’ questionnaire
In this table, we can see that the level of the translation game was just about perfect for beginners, since a majority stated that the game was neither too difficult nor too easy. Half of the beginners also enjoyed playing the game and only 14% did not like it. One thing that stands out is that a big majority perceived the help function as being useful, both to help them find out what they have to say (written help), as well as to help them improve their pronunciation (spoken help). The main point of criticism was that they were not very well understood by the application, meaning that the speech recognition did not perform very well or that their pronunciation was not good enough to be recognised (81% of all beginner’s participants were not happy with the performance of the speech recognition).

In Table 5.2, we can see the evaluation of the intermediate users. It differs slightly from the beginner’s evaluation as they rather found the game too easy but still most of them were fine with the level. An interesting point is that a quarter stated that the help function did not help them improve their pronunciation which may be because the vocabulary covered by the prototype was easy enough for the intermediate users as they did not need the spoken help function; however, this cannot be said with certainty. Again, the speech recognition was evaluated as being rather poor (62%).

<table>
<thead>
<tr>
<th>Intermediate (70% of total)</th>
<th>++</th>
<th>+</th>
<th>=</th>
<th>-</th>
<th>--</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 The game was too difficult</td>
<td>2%</td>
<td>8%</td>
<td>24%</td>
<td>52%</td>
<td>15%</td>
</tr>
<tr>
<td>2 The game was too easy</td>
<td>4%</td>
<td>18%</td>
<td>34%</td>
<td>41%</td>
<td>3%</td>
</tr>
<tr>
<td>3 I enjoyed learning English with the game</td>
<td>20%</td>
<td>42%</td>
<td>26%</td>
<td>9%</td>
<td>3%</td>
</tr>
<tr>
<td>4 The game helped me improve my English</td>
<td>6%</td>
<td>34%</td>
<td>33%</td>
<td>18%</td>
<td>9%</td>
</tr>
<tr>
<td>5 I would use this game again if I could</td>
<td>7%</td>
<td>32%</td>
<td>43%</td>
<td>14%</td>
<td>4%</td>
</tr>
<tr>
<td>6 I would use a similar game which covered another topic</td>
<td>9%</td>
<td>31%</td>
<td>52%</td>
<td>6%</td>
<td>2%</td>
</tr>
<tr>
<td>7 I would recommend the game to my friends</td>
<td>12%</td>
<td>35%</td>
<td>33%</td>
<td>17%</td>
<td>3%</td>
</tr>
<tr>
<td>8 I would prefer a game which focused only on vocabulary</td>
<td>5%</td>
<td>13%</td>
<td>35%</td>
<td>42%</td>
<td>5%</td>
</tr>
<tr>
<td>9 I would prefer a game which was more interactive</td>
<td>7%</td>
<td>31%</td>
<td>41%</td>
<td>26%</td>
<td>1%</td>
</tr>
<tr>
<td>10 It was easy to understand what I was meant to say</td>
<td>23%</td>
<td>46%</td>
<td>21%</td>
<td>9%</td>
<td>1%</td>
</tr>
<tr>
<td>11 The help system helped me find what I was meant to say</td>
<td>44%</td>
<td>30%</td>
<td>22%</td>
<td>4%</td>
<td>0%</td>
</tr>
<tr>
<td>12 The help system helped me find what I had mispronounced</td>
<td>14%</td>
<td>30%</td>
<td>26%</td>
<td>25%</td>
<td>5%</td>
</tr>
<tr>
<td>13 It was easy to get the machine to understand me</td>
<td>3%</td>
<td>16%</td>
<td>25%</td>
<td>33%</td>
<td>23%</td>
</tr>
</tbody>
</table>

Table 5.2: Results of intermediates’ questionnaire

Table 5.3 shows the results of the advanced users. Here, we can clearly see that 99% of all users did not find the translation game too difficult. It is, however, interesting that only a third said that the game was too easy. It can therefore be assumed that
some of the tasks were also challenging for advanced students. Another very inter-
esting point is that the advanced users also said that the system did not understand
them. This is interesting since one would think that the performance of the speech
recognition is automatically better if the pronunciation is good.

<table>
<thead>
<tr>
<th>Advanced (15% of total)</th>
<th>++</th>
<th>+</th>
<th>=</th>
<th>-</th>
<th>--</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 The game was too difficult</td>
<td>1%</td>
<td>0%</td>
<td>18%</td>
<td>52%</td>
<td>29%</td>
</tr>
<tr>
<td>2 The game was too easy</td>
<td>8%</td>
<td>32%</td>
<td>34%</td>
<td>26%</td>
<td>0%</td>
</tr>
<tr>
<td>3 I enjoyed learning English with the game</td>
<td>12%</td>
<td>30%</td>
<td>27%</td>
<td>18%</td>
<td>12%</td>
</tr>
<tr>
<td>4 The game helped me improve my English</td>
<td>0%</td>
<td>22%</td>
<td>38%</td>
<td>23%</td>
<td>17%</td>
</tr>
<tr>
<td>5 I would use this game again if I could</td>
<td>4%</td>
<td>17%</td>
<td>49%</td>
<td>13%</td>
<td>17%</td>
</tr>
<tr>
<td>6 I would use a similar game which covered another topic</td>
<td>8%</td>
<td>27%</td>
<td>44%</td>
<td>7%</td>
<td>14%</td>
</tr>
<tr>
<td>7 I would recommend the game to my friends</td>
<td>11%</td>
<td>22%</td>
<td>39%</td>
<td>14%</td>
<td>14%</td>
</tr>
<tr>
<td>8 I would prefer a game which focused only on vocabulary</td>
<td>4%</td>
<td>12%</td>
<td>39%</td>
<td>35%</td>
<td>10%</td>
</tr>
<tr>
<td>9 I would prefer a game which was more interactive</td>
<td>9%</td>
<td>31%</td>
<td>43%</td>
<td>16%</td>
<td>1%</td>
</tr>
<tr>
<td>10 It was easy to understand what I was meant to say</td>
<td>32%</td>
<td>31%</td>
<td>25%</td>
<td>9%</td>
<td>3%</td>
</tr>
<tr>
<td>11 The help system helped me find what I was meant to say</td>
<td>24%</td>
<td>34%</td>
<td>30%</td>
<td>4%</td>
<td>8%</td>
</tr>
<tr>
<td>12 The help system helped me find what I had mispronounced</td>
<td>18%</td>
<td>39%</td>
<td>18%</td>
<td>21%</td>
<td>4%</td>
</tr>
<tr>
<td>13 It was easy to get the machine to understand me</td>
<td>3%</td>
<td>12%</td>
<td>23%</td>
<td>40%</td>
<td>22%</td>
</tr>
</tbody>
</table>

Table 5.3: Results of advanced questionnaire

As a conclusion of this first evaluation, we can say that a vast majority liked the
CALL-SLT translation game and also thought that they can improve their English
with the application. Many of the users also said that they would use such a system
again to improve their language and pronunciation skills. The main negative point
which was mentioned throughout the various language levels was the poor perfor-
mance of the speech recognition. In order to satisfy the language learners fully, the
speech recognition has to be improved; another point which could be improved is the
vocabulary, which many testers thought was too limited. This can easily be remedied
by adding more vocabulary to the lexicon and is certainly a goal for a future version
of the English CALL-SLT.

5.2 Evaluation of CALL-SLT for L2 = German

In order to get an evaluation of the German CALL-SLT, which has been developed
for the purpose of this thesis, I asked about ten university students to test the pro-
totype version. The tested version worked with French prompts and was destined at
non-native German speakers; however, in order to get a more extensive evaluation,
I also asked a native speaker to test the system, as well as another person who had no previous knowledge of the German language whatsoever. All other testers had some basic knowledge of German which had been acquired during German classes at school between a few months and twelve years. The age range of the people who tested the application was between 23 and 30 years.

In Table 5.4, we can see the results of the questionnaires, filled out by the testers once they had tested the German CALL-SLT. In order not to falsify the results, the answers of the native German speaker have been displayed in a separate table (Table 5.5).

<table>
<thead>
<tr>
<th>Non-native German speaker</th>
<th>++</th>
<th>+</th>
<th>-</th>
<th>-</th>
<th>-</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>The game was too difficult</td>
<td>0%</td>
<td>0%</td>
<td>0%</td>
<td>17%</td>
</tr>
<tr>
<td>2</td>
<td>The game was too easy</td>
<td>0%</td>
<td>17%</td>
<td>33%</td>
<td>50%</td>
</tr>
<tr>
<td>3</td>
<td>I enjoyed learning German with the game</td>
<td>83%</td>
<td>17%</td>
<td>0%</td>
<td>0%</td>
</tr>
<tr>
<td>4</td>
<td>The game helped me improve my German</td>
<td>34%</td>
<td>66%</td>
<td>0%</td>
<td>0%</td>
</tr>
<tr>
<td>5</td>
<td>I would use this game again if I could</td>
<td>50%</td>
<td>0%</td>
<td>33%</td>
<td>17%</td>
</tr>
<tr>
<td>6</td>
<td>I would use a similar game which covered another topic</td>
<td>50%</td>
<td>17%</td>
<td>33%</td>
<td>0%</td>
</tr>
<tr>
<td>7</td>
<td>I would recommend the game to my friends</td>
<td>66%</td>
<td>34%</td>
<td>0%</td>
<td>0%</td>
</tr>
<tr>
<td>8</td>
<td>I would prefer a game which focused only on vocabulary</td>
<td>0%</td>
<td>0%</td>
<td>0%</td>
<td>34%</td>
</tr>
<tr>
<td>9</td>
<td>I would prefer a game which was more interactive</td>
<td>0%</td>
<td>17%</td>
<td>33%</td>
<td>33%</td>
</tr>
<tr>
<td>10</td>
<td>It was easy to understand what I was meant to say</td>
<td>50%</td>
<td>50%</td>
<td>0%</td>
<td>0%</td>
</tr>
<tr>
<td>11</td>
<td>The help system helped me find what I was meant to say</td>
<td>100%</td>
<td>0%</td>
<td>0%</td>
<td>0%</td>
</tr>
<tr>
<td>12</td>
<td>The help system helped me find what I had mispronounced</td>
<td>33%</td>
<td>33%</td>
<td>0%</td>
<td>0%</td>
</tr>
<tr>
<td>13</td>
<td>It was easy to get the machine to understand me</td>
<td>34%</td>
<td>66%</td>
<td>0%</td>
<td>0%</td>
</tr>
</tbody>
</table>

Table 5.4: Results of the German CALL-SLT questionnaire

<table>
<thead>
<tr>
<th>Native German speaker</th>
<th>++</th>
<th>+</th>
<th>-</th>
<th>-</th>
<th>-</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>The game was too difficult</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>The game was too easy</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>I enjoyed learning German with the game</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>The game helped me improve my German</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>I would use this game again if I could</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>I would use a similar game which covered another topic</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>I would recommend the game to my friends</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>I would prefer a game which focused only on vocabulary</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>I would prefer a game which was more interactive</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>It was easy to understand what I was meant to say</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>The help system helped me find what I was meant to say</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>The help system helped me find what I had mispronounced</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>It was easy to get the machine to understand me</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 5.5: Results of the German CALL-SLT questionnaire by a native-speaker

There are some interesting points which can be derived from this evaluation.

To start we will take a look at the positive feedback that was given. First of all, the level of difficulty seemed to be adequate for most users; however, it has to be
mentioned that the 17% who stated that the game was too easy used a lot of help and for this reason this result must be kept relative. A very positive result can be seen in question three where a large majority stated that they liked learning German with CALL-SLT. The result of question four is also highly encouraging since all of the non-native speakers thought that the application helped them improve their German. Another point which can clearly be extracted from the questionnaire results is that none of the users would want a game which only focuses on vocabulary, which means that the project is going in the right direction since it is planned to introduce more focused grammar lessons. The prompts seem to be quite clear, as all of the testers thought that they easily understood what they were supposed to say and also the speech recognition seemed to have improved since the first evaluation of the English translation game. One reason is certainly that these tests have been conducted in a quiet environment where no background noise disturbed the recording. As far as the help function is concerned, we can see that all testers liked both the written and spoken help. Furthermore, all of the testers stated in their comments that they were overall satisfied with the CALL-SLT application and that they enjoyed playing with the program.

The negative questionnaire results and comments include that most users would like the application to cover not only the restaurant domain, but also other scenarios. As far as the help function is concerned, some of the testers mentioned that they were not always certain about which part of their utterance has been judged as being faulty; however, this should have been clear when comparing the interlingua of the uttered sentence with the interlingua of the prompt. Furthermore, many of the testers suggested that pictures could help to illustrate the vocabulary, since not everyone was familiar with all the special vocabulary from the restaurant domain, as for example "leg of mutton". The help function was criticised for giving the whole sentence at once: the testers have stated that they would prefer a help function which gives help in various steps in order to separate the sentence structure from vocabulary. Another approach to the help function might be to insert a phonetic transcription of difficult vocabulary into the written help, as an intermediate step between the written and the spoken help. It has also been suggested that it would be
helpful if the learner had to repeat the sentence without the help being displayed\textsuperscript{12} as well as to hear what one had said in order to find the mistakes.

Another interesting point which can be seen in Table 5.5 is that the native speaker only marked a "+" for question thirteen which suggests that the speech recognition obviously did not recognise all utterances from the native speaker which is rather unsatisfactory.

Since speech recognition has proven to be an important issue in the evaluation of the English CALL-SLT prototype, I took a closer look at the performance of speech recognition in this second evaluation. I therefore evaluated the speech recognition with four different categories, namely "correct sentences which have been recognised as being correct" (true positive), "correct sentences which have been recognised as being incorrect" (false negative), "incorrect sentences which have been recognised as being incorrect" (true negative) and "incorrect sentences which have been recognised as being correct" (false positive).

The analysis resulted in a Recall Rate (True Positive Rate) of 88.4%. This rate indicates the percentage of actual positives that are correctly identified as such. The Precision Rate lies at 96.8% and indicates the percentage of correctly identified sentences that are actual positives. Table 5.6 displays the detailed results.

\begin{table}[h]
\centering
\begin{tabular}{|c|c|}
\hline
true positive & false positive \\
80.8\% & 2.6\% \\
\hline
false negative & true negative \\
10.6\% & 5.9\% \\
\hline
\end{tabular}
\caption{Table 5.6: Results of the German CALL-SLT speech recognition performance}
\end{table}

From this evaluation, we can see that a good 80% of all utterances are correctly recognised by the CALL-SLT application. 10% are not recognised as being correct, even though they should be, since they are correct utterances. The system correctly analyses 6% as being incorrect; however, a little more than 2% are being recognised as being correct, even though they are false. In sum, it can be said that around 87% are correctly recognised and only around 13% are incorrectly recognised. This is an\textsuperscript{12} This function is already available in the most recent version of CALL-SLT
acceptable rate, considering that we are talking about a first prototype version. The most frustrating recognition faults are probably those which are falsely misrecognised (namely the 10% of the false negatives), hence why an effort should be made to avoid such errors.

5.3 Conclusion

The two evaluations described in this chapter indicate that most users liked learning a second language with the CALL-SLT application and managed to understand the prompts fairly easily. The main negative point which has been mentioned is the performance of the speech recognition which has been poor in some cases, mainly in the first evaluation of the English CALL-SLT. Other points of criticism regarded the limited domain covered by the application, as well as the unstructured approach to grammatical issues; however, these are all points upon which can be improved in future versions. Some of the changes which shall be realised in the next step will be discussed in the final conclusion in the next chapter.
6 Conclusion

CALL has lived through important developments over the past fifty years. After the initial attempts at Computer Assisted Language Learning with learning machines which could automatically correct multiple choice questions, the technologies were improved and real-world scenarios were included in the applications. But it was only in the early 21st century that an important improvement in speech recognition allowed for CALL to include pronunciation exercises in which the students could actually speak with the computer.

The most promising CALL applications, due to their relatively easy construction and good performance, are the so-called translation game applications. The Translation Game was introduced by Wang and Seneff in 2006 and has then been further developed and improved upon in another project: the CALL-SLT Project. As we have seen in this thesis, the project shows prompts in a human-readable language which is not too closely linked to the student’s L1 and uses speech recognition in order to allow the student to speak to the computer and improve his/her pronunciation skills. The project is based on a rule-based approach with which the grammaticality of the learner’s utterances can be guaranteed. The student’s utterances are compared to the interlingua representation of the initial prompt via transfer rules so that the application can give immediate feedback on the correctness of the student’s answer. Finally, the help function gives correct examples in either the written or spoken form. These two help options can inspire the student with a number of different and correct sentences which are covered by the interlingua of the prompt, as well as being able to help the student to pronounce the sentence correctly.

The main focus of this thesis lay on the practical section in which I developed a version of CALL-SLT for German and evaluated the resulting application. This was a very interesting experience since I had the opportunity to work in a real project and therefore not only gained insight into the technical, but also the developmental and organisational components of such a project. As far as the development of the
German CALL-SLT is concerned, it was fascinating to see how an existing grammar (the one developed for the MedSLT application) can be adapted to a new domain fairly easily by adding the necessary rules and vocabulary. I learned a lot during the process of writing this thesis, mainly in terms of working with the Regulus platform of which I only had basic knowledge in the beginning.

The most gratifying fact about this thesis is that the developed CALL-SLT application for German works nicely and as the evaluation showed, most testers were satisfied with the result.

The work carried out for this thesis is interesting in two points: firstly, a functioning prototype of the German CALL-SLT application has been developed and secondly, this prototype lays the foundation for further developments and improvements which can lead to an efficient software for learning German as a second language.

The next step in the project will be to implement a lesson plan, so that the student can specifically choose the lesson s/he wants to practice. For example greetings and politeness ("hello", "good evening", "please", etc.), asking for things ("I would like + noun phrase"), asking for things using questions ("Could I have + noun phrase"), numbers ("I would like a table for three people") or times ("I would like a table for six o’clock")[Rayner et al. 2010].

This initial lesson plan will then be further developed to specify single grammatical issues which are specific to each language, as for example masculine nouns ("Ich hätte gerne einen Tisch") and feminine nouns ("Ich hätte gerne eine Gabel"), plurals ("Ich hätte gerne drei Tische"), adjectives ("Ich hätte gerne einen grossen Tisch") or infinitive forms ("Ich möchte einen Tisch um sieben Uhr reservieren") for German.

Once this lesson plan has been implemented, the CALL-SLT application will already be very efficient and useful for second language studies.
References


[54] M. Zock and J. Quint. Why have them work for peanuts, when it is so easy to provide reward? Motivations for converting a dictionary into a drill tutor. Proceedings PAPILLON 2004, 2004.

Appendix

A.1 GSL Poem

NUANCE GRAMMAR SPECIFICATION LANGUAGE (GSL)

Before you write a grammar file,
It only takes a little while
To learn the simple syntax rules
You need to use the Nuance tools.
The semicolon’s thrown away.
Just lowercase the words you say,
But uppercase each grammar name
Since big and little aren’t the same.
Put a dot before a grammar
That your app might have to scan for.
Grammars Nuance has approved
You can get with pound include.
Parentheses concatenate,
The brackets mark the alternates.
A question mark precedes a word
If sometimes it might not be heard.
Use plus or star to iterate,
But not to overgenerate.
A value a subgrammar passed
Must be put somewhere at last—
Use a colon and then say
The name for storing it away.
Unless you put it in a slot
The system acts like it forgot.
Use curly braces for NL—
That’s the stuff you use to tell
Your program what each sentence means
Using slots as go-betweens.
With string and structure, int and list,
Several slot types do exist.
Strings you see set off by quotes
Ints are numbers, so are floats.
But there inside the curly braces,
Delimiters show different faces—
With angle brackets you’ve assigned
A slot a value and a kind.
Within brackets inside braces,
Angles fill in structure places!
Parentheses mean function call,
But wait a minute—that’s not all.
A list is set off by them, too.
They also mark return value.
A function call is not like C—
No comma must there ever be.
Use a dollar sign to get
The variables you have set.
Don’t forget that you must list
Each slot your app needs to exist
In another separate file
If you want things to compile.
When you’ve learned these rules well,
You can program GSL!

Reference: http://www.adducive.com/software_design/nuance_grammar_rules.html
B.1 German CALL-SLT

B.1.1 Corpus

sent('hallo').
sent('guten tag').
sent('servus').
sent('grüss gott').
sent('guten abend').

%food/drinks
sent('ich möchte ein bier').
sent('ich möchte gerne ein bier').
sent('ich hätte gerne ein bier').
sent('dürfte ich ein bier haben').
sent('ich nehme ein bier').
sent('geben sie mir bitte ein bier').
sent('ein bier bitte').

sent('ich hätte gerne sechs gläser weisswein').
sent('ich hätte gerne fünf biere').
sent('ich möchte drei hamburger').

sent('ich möchte einen hamburger').
sent('ich möchte gerne einen hamburger').
sent('ich hätte gerne einen hamburger').
sent('dürfte ich einen hamburger haben').

sent('ich möchte zwei hamburger').
sent('ich möchte gerne zwei hamburger').
sent('ich hätte gerne zwei hamburger').
sent('dürfte ich zwei hamburger haben').
sent('zwei hamburger bitte').

sent('einen hamburger und pommes bitte').

sent('ich möchte gerne etwas wasser').
sent('ich möchte etwas wasser').
sent('ich hätte gerne etwas wasser').
sent('dürfte ich etwas wasser haben').
sent('ich möchte ein wasser').
sent('ich möchte gerne ein wasser').
sent('ich hätte gerne ein wasser').
sent('dürfte ich ein wasser haben').

sent('wir hätten gerne eine flasche wein').
sent('ich möchte eine flasche wein').
sent('ich möchte gerne eine flasche wein').
sent('dürfte ich eine flasche wein haben').
sent('ich hätte gerne ein glas wein').
sent('könnte ich ein glas rotwein haben').
sent('dürfte ich ein glas weisswein haben').
sent('wir hätten gerne eine zweite flasche des selben weines').
sent('wir hätten gerne noch eine flasche des selben weines').

sent('ich hätte gerne ein glas wasser mit kohlensäure').
sent('ich möchte ein glas sprudelwasser').
sent('ich möchte ein sprudelwasser').
sent('ich möchte ein glas wasser mit kohlensäure').
sent('ich möchte gerne ein glas wasser mit kohlensäure').
sent('dürfte ich ein glas wasser mit kohlensäure haben').
sent('könnten wir drei flaschen wasser ohne kohlensäure haben').
sent('ich nehme ein stilles wasser').
sent('könnte ich ein glas stilles wasser haben').
sent('ich hätte gerne ein glas wasser ohne kohlensäure').
sent('ich möchte ein glas wasser ohne kohlensäure').
sent('ich möchte gerne ein glas wasser ohne kohlensäure').
sent('dürfte ich ein glas wasser ohne kohlensäure haben').

sent('ich hätte gerne einen kaffee und ein glas wasser').

sent('ich hätte gerne eine tasse kaffee').
sent('ich möchte eine tasse kaffee').

sent('könnte ich bitte eine heisse schokolade haben').
sent('ich hätte gerne eine heisse schokolade').
sent('ich möchte eine heisse schokolade bitte').
sent('ich möchte gerne eine heisse schokolade').
sent('dürfte ich eine heisse schokolade haben').
sent('wir hätten gerne vier heisse schokoladen').

sent('ich hätte gerne einen pfefferminztee').
sent('könnte ich bitte einen pfefferminztee haben').
sent('ich möchte einen pfefferminztee bitte').
sent('ich möchte gerne einen pfefferminztee').
sent('dürfte ich einen pfefferminztee haben').

sent('ich hätte gerne einen whisky').
sent('könnte ich bitte einen whisky haben').
sent('ich möchte einen whisky bitte').
sent('ich möchte gerne einen whisky').
sent('dürfte ich einen whisky haben').

sent('ich hätte gerne einen orangensaft').
sent('könnte ich bitte einen orangensaft haben').
sent('ich möchte einen orangensaft bitte').
sent('ich möchte gerne einen orangensaft').
sent('dürfte ich einen orangensaft haben').

sent('ich hätte gerne etwas reis').
sent('könnte ich bitte etwas reis haben').
sent('ich möchte etwas reis bitte').
sent('ich möchte gerne etwas reis').
sent('dürfte ich etwas reis haben').
sent('ich hätte gerne den reis').
sent('könnte ich bitte den reis haben').
sent('ich möchte den reis bitte').
sent('ich möchte gerne den reis').
sent('dürfte ich den reis haben bitte').
sent('ich hätte gerne etwas salz').
sent('könnte ich bitte etwas salz haben').
sent('ich möchte etwas salz bitte').
sent('ich möchte gerne etwas salz').
sent('dürfte ich etwas salz haben').

sent('ich hätte gerne etwas pfeffer').
sent('könnte ich bitte etwas pfeffer haben').
sent('ich möchte etwas pfeffer bitte').
sent('ich möchte gerne etwas pfeffer').
sent('dürfte ich etwas pfeffer haben').

sent('ich hätte gerne etwas essig').
sent('könnte ich bitte etwas essig haben').
sent('ich möchte etwas essig bitte').
sent('ich möchte gerne etwas essig').
sent('dürfte ich etwas essig haben').

sent('könnten wir bitte etwas zucker haben').
sent('ich hätte gerne etwas zucker').
sent('ich möchte etwas zucker bitte').
sent('ich möchte gerne etwas zucker').
sent('dürfte ich etwas zucker haben').

sent('könnte ich bitte etwas süßstoff haben').
sent('ich hätte gerne etwas süßstoff').
sent('ich möchte etwas süßstoff bitte').
sent('ich möchte gerne etwas süßstoff').
sent('dürfte ich etwas süßstoff haben').

sent('könnten wir bitte mehr brot haben').
sent('ich hätte gerne mehr brot').
sent('ich möchte mehr brot bitte').
sent('ich möchte gerne mehr brot').
send('dürfte ich mehr brot haben').
send('könnten wir bitte etwas mehr brot haben').

sent('ich hätte gerne einen salat').
send('könnte ich bitte einen salat haben').
send('ich möchte einen salat bitte').
send('ich möchte gerne einen salat').
send('dürfte ich einen salat haben').

send('ich hätte gerne die schweinskoteletten').
send('könnte ich bitte die schweinskoteletten haben').
send('ich möchte die schweinskoteletten bitte').
send('ich möchte gerne die schweinskoteletten').
send('dürfte ich die schweinskoteletten haben').

send('ich hätte gerne das schweinsschnitzel').
send('könnte ich bitte das schweinsschnitzel haben').
send('ich möchte das schweinsschnitzel bitte').
send('ich möchte gerne das schweinsschnitzel').
send('dürfte ich das schweinsschnitzel haben').

90
sent('ich hätte gerne den lammrücken').
sent('könnte ich bitte den lammrücken haben').
sent('ich möchte den lammrücken bitte').
sent('ich möchte gerne den lammrücken').
sent('dürfte ich den lammrücken haben').

sent('ich hätte gerne das rindsragout').
sent('könnte ich bitte das rindsragout haben').
sent('ich möchte das rindsragout bitte').
sent('ich möchte gerne das rindsragout').
sent('dürfte ich das rindsragout haben').

sent('ich hätte gerne das ragout').
sent('könnte ich bitte das ragout haben').
sent('ich möchte das ragout bitte').
sent('ich möchte gerne das ragout').
sent('dürfte ich das ragout haben').

sent('ich hätte gerne das kalbsragout').
sent('könnte ich bitte das kalbsragout haben').
sent('ich möchte das kalbsragout bitte').
sent('ich möchte gerne das kalbsragout').
sent('dürfte ich das kalbsragout haben').

sent('ich hätte gerne die hammelkeule').
sent('könnte ich bitte die hammelkeule haben').
sent('ich möchte die hammelkeule bitte').
sent('ich möchte gerne die hammelkeule').
sent('dürfte ich die hammelkeule haben').

sent('ich hätte gerne die lammkeule').
sent('könnte ich bitte die lammkeule haben').
sent('ich möchte die lammkeule bitte').
sent('ich möchte gerne die lammkeule').
sent('dürfte ich die lammkeule haben').

sent('ich hätte gerne die lammkoteletten').
sent('könnte ich bitte die lammkoteletten haben').
sent('ich möchte die lammkoteletten bitte').
sent('ich möchte gerne die lammkoteletten').
sent('dürfte ich die lammkoteletten haben').

sent('ich hätte gerne ein steak').
sent('könnte ich bitte ein steak haben').
sent('ich möchte das steak bitte').
sent('ich möchte gerne ein steak').
sent('dürfte ich das steak haben').

sent('ich hätte gerne zwei steaks').
sent('könnte ich bitte zwei steaks haben').
sent('ich möchte zwei steaks bitte').
sent('ich möchte gerne zwei steaks').
sent('dürfte ich zwei steaks haben').

sent('ich hätte gerne ein ribeyestek').
sent('könnte ich bitte das ribeyestek haben').
sent('ich möchte ein ribeyestek bitte').
sent('ich möchte gerne das ribeyesteak').
sent('dürfte ich das ribeyesteak haben').

sent('ich hätte gerne das roastbeef').
sent('könnte ich bitte das roastbeef haben').
sent('ich möchte das roastbeef bitte').
sent('dürfte ich das roastbeef haben').

sent('ich hätte gerne das wild').
sent('könnte ich bitte das wild haben').
sent('ich möchte das wild bitte').
sent('dürfte ich das wild haben').

sent('ich hätte gerne das kalbfleisch').
sent('könnte ich bitte das kalbfleisch haben').
sent('ich möchte das kalbfleisch bitte').
sent('dürfte ich das kalbfleisch haben').

sent('ich hätte gerne das hähnchen').
sent('könnte ich bitte das hähnchen haben').
sent('ich möchte das hähnchen bitte').
sent('dürfte ich das hähnchen haben').

sent('ich hätte gerne die ente').
sent('könnte ich bitte die ente haben').
sent('ich möchte die ente bitte').
sent('dürfte ich die ente haben').

sent('ich hätte gerne das kaninchen').
sent('könnte ich bitte das kaninchen haben').
sent('ich möchte das kaninchen bitte').
sent('dürfte ich das kaninchen haben').

sent('ich hätte gerne den kanincheneintopf').
sent('könnte ich bitte den kanincheneintopf haben').
sent('ich möchte den kanincheneintopf bitte').
sent('dürfte ich den kanincheneintopf haben').

sent('ich hätte gerne das filet').
sent('könnte ich bitte das filet haben').
sent('ich möchte das filet bitte').
sent('dürfte ich das filet haben').

sent('ich hätte gerne das schweinsfilet').
sent('könnte ich bitte das schweinsfilet haben').
sent('ich möchte das schweinsfilet bitte').
sent('dürfte ich das schweinsfilet haben').
ich hätte gerne die austern'.
könnte ich bitte die austern haben').
ich möchte die austern bitte').
ich möchte gerne die austern').
dürfte ich die austern haben').

ich hätte gerne den fisch').
könnte ich bitte den fisch haben').
ich möchte den fisch bitte').
ich möchte gerne den fisch').
dürfte ich den fisch haben').

ich hätte gerne die lasagne').
könnte ich bitte die lasagne haben').
ich möchte die lasagne bitte').
ich möchte gerne die lasagne').
dürfte ich die lasagne haben').

ich hätte gerne ein dutzend austern').
ich hätte gerne zwei dutzend austern').

ich hätte gerne die quiche').
könnte ich bitte die quiche haben').
ich möchte die quiche bitte').
ich möchte gerne die quiche').
dürfte ich die quiche haben').

ich hätte gerne eine pizza ohne sardellen').

ich hätte gerne das kalbsbries').
könnte ich bitte das kalbsbries haben').
ich möchte das kalbsbries bitte').
ich möchte gerne das kalbsbries').
dürfte ich das kalbsbries haben').

ich hätte gerne den seebarsch').
könnte ich bitte den seebarsch haben').
ich möchte den seebarsch bitte').
ich möchte gerne den seebarsch').
dürfte ich den seebarsch haben').
den seebarsch bitte').

könnten wir bitte einen aschenbecher haben').
ich hätte gerne einen aschenbecher').
dürfte ich einen aschenbecher haben').
könnte ich bitte eine gabel haben').
einen aschenbecher bitte').
ich hätte gerne eine gabel').
dürfte ich eine gabel haben').
könnte ich bitte ein messer haben').
ich hätte gerne ein messer').
dürfte ich ein messer haben').
könnte ich bitte einen löffel haben').
ich hätte gerne einen löffel').
dürfte ich einen löffel haben').
könnte ich bitte einen teelöffel haben
ich hätte gerne einen teelöffel
dürfte ich einen teelöffel haben
könnte ich bitte einen suppenlöffel haben
ich hätte gerne einen suppenlöffel
dürfte ich einen suppenlöffel haben
könnte ich bitte einen teller haben
ich hätte gerne einen teller
dürfte ich einen teller haben
könnte ich bitte ein glas haben
ich hätte gerne ein glas
dürfte ich ein glas haben
könnte ich bitte einen zahnstocher haben
ich hätte gerne einen zahnstocher
dürfte ich einen zahnstocher haben
könnte ich bitte eine serviette haben
ich hätte gerne eine serviette
dürfte ich eine serviette haben
könnte ich bitte eine tasse haben
ich hätte gerne eine tasse
dürfte ich eine tasse haben
ich möchte einen tisch um sieben uhr
könnte ich bitte einen tisch um sieben uhr haben
ich möchte einen tisch um sieben uhr bitte
ich möchte gerne einen tisch um sieben uhr
einen tisch um sieben uhr bitte
ich möchte einen tisch um neunzehn uhr
könnte ich bitte einen tisch um neunzehn uhr haben
ich möchte einen tisch um neunzehn uhr bitte
ich möchte gerne einen tisch um neunzehn uhr
einen tisch um neunzehn uhr bitte
ich möchte einen tisch um fünf nach sieben
könnte ich bitte einen tisch um fünf nach sieben haben
ich möchte einen tisch um fünf nach sieben bitte
ich möchte gerne einen tisch um fünf nach sieben
einen tisch um fünf nach sieben bitte
ich möchte einen tisch um sieben uhr fünf
könnte ich bitte einen tisch um sieben uhr fünf haben
ich möchte einen tisch um sieben uhr fünf bitte
ich möchte gerne einen tisch um sieben uhr fünf
ich möchte einen tisch um zehn nach sieben
könnte ich bitte einen tisch um zehn nach sieben haben
ich möchte einen tisch um zehn nach sieben bitte
ich möchte gerne einen tisch um zehn nach sieben
ich möchte einen tisch um sieben uhr zehn
könnte ich bitte einen tisch um sieben uhr zehn haben
ich möchte einen tisch um sieben uhr zehn bitte
ich möchte gerne einen tisch um sieben uhr zehn
ich möchte einen tisch um fünfhundert zehn nach sieben
könnte ich bitte einen tisch um fünfhundert zehn nach sieben haben
ich möchte einen tisch um fünfhundert zehn nach sieben bitte
ich möchte gerne einen tisch um fünfhundert zehn nach sieben
ich möchte einen tisch um fünfzehn nach sieben bitte'.

ich möchte gerne einen tisch um fünfzehn nach sieben'.

ich möchte einen tisch um sieben uhr fünfzehn'.

könnte ich bitte einen tisch um sieben uhr fünfzehn haben'.

ich möchte einen tisch um sieben uhr fünfzehn bitte'.

ich möchte gerne einen tisch um sieben uhr fünfzehn'.

ich möchte einen tisch um viertel nach sieben'.

könnte ich bitte einen tisch um viertel nach sieben haben'.

ich möchte einen tisch um viertel nach sieben bitte'.

ich möchte gerne einen tisch um viertel nach sieben'.

ich möchte einen tisch um zwanzig nach sieben'.

könnte ich bitte einen tisch um zwanzig nach sieben haben'.

ich möchte einen tisch um zwanzig nach sieben bitte'.

ich möchte gerne einen tisch um zwanzig nach sieben'.

ich möchte einen tisch um sieben uhr zwanzig'.

könnte ich bitte einen tisch um sieben uhr zwanzig haben'.

ich möchte einen tisch um sieben uhr zwanzig bitte'.

ich möchte gerne einen tisch um sieben uhr zwanzig'.

ich möchte einen tisch um fünfundzwanzig nach sieben'.

könnte ich bitte einen tisch um fünfundzwanzig nach sieben haben'.

ich möchte einen tisch um fünfundzwanzig nach sieben bitte'.

ich möchte gerne einen tisch um fünfundzwanzig nach sieben'.

ich möchte einen tisch um sieben uhr fünfundzwanzig'.

könnte ich bitte einen tisch um sieben uhr fünfundzwanzig haben'.

ich möchte einen tisch um sieben uhr fünfundzwanzig bitte'.

ich möchte gerne einen tisch um sieben uhr fünfundzwanzig'.

ich möchte einen tisch um fünf vor halb sieben'.

könnte ich bitte einen tisch um fünf vor halb sieben haben'.

ich möchte einen tisch um fünf vor halb sieben bitte'.

ich möchte gerne einen tisch um fünf vor halb sieben'.

ich möchte einen tisch um sechs uhr dreissig'.

könnte ich bitte einen tisch um sechs uhr dreissig haben'.

ich möchte einen tisch um sechs uhr dreissig bitte'.

ich möchte gerne einen tisch um sechs uhr dreissig'.

ich möchte einen tisch um halb sieben'.

könnte ich bitte einen tisch um halb sieben haben'.

ich möchte einen tisch um halb sieben bitte'.

ich möchte gerne einen tisch um halb sieben'.

ich möchte einen tisch um fünfundzwanzig vor sieben'.

könnte ich bitte einen tisch um fünfundzwanzig vor sieben haben'.

ich möchte einen tisch um fünfundzwanzig vor sieben bitte'.

ich möchte gerne einen tisch um fünfundzwanzig vor sieben'.

ich möchte einen tisch um sieben uhr fünfunddreissig'.

könnte ich bitte einen tisch um sieben uhr fünfunddreissig haben'.

ich möchte einen tisch um sieben uhr fünfunddreissig bitte'.

ich möchte gerne einen tisch um sieben uhr fünfunddreissig'.
ich möchte einen tisch um fünf nach halb sieben.
könnte ich bitte einen tisch um fünf nach halb sieben haben.
ich möchte einen tisch um fünf nach halb sieben bitte.
ich möchte gerne einen tisch um fünf nach halb sieben.

ich möchte einen tisch um zwanzig vor sieben.
könnte ich bitte einen tisch um zwanzig vor sieben haben.
ich möchte einen tisch um zwanzig vor sieben bitte.
ich möchte gerne einen tisch um zwanzig vor sieben.

ich möchte einen tisch um sieben uhr vierzig.
könnte ich bitte einen tisch um sieben uhr vierzig haben.
ich möchte einen tisch um sieben uhr vierzig bitte.
ich möchte gerne einen tisch um sieben uhr vierzig.

ich möchte einen tisch um fünfzehn vor sieben.
könnte ich bitte einen tisch um fünfzehn vor sieben haben.
ich möchte einen tisch um fünfzehn vor sieben bitte.
ich möchte gerne einen tisch um fünfzehn vor sieben.

ich möchte einen tisch um sieben uhr fünfzehn vor sieben.
könnte ich bitte einen tisch um sieben uhr fünfzehn vor sieben haben.
ich möchte einen tisch um sieben uhr fünfzehn vor sieben bitte.
ich möchte gerne einen tisch um sieben uhr fünfzehn vor sieben.

ich möchte einen tisch um viertel vor sieben.
könnte ich bitte einen tisch um viertel vor sieben haben.
ich möchte einen tisch um viertel vor sieben bitte.
ich möchte gerne einen tisch um viertel vor sieben.

ich möchte einen tisch um zehn vor sieben.
könnte ich bitte einen tisch um zehn vor sieben haben.
ich möchte einen tisch um zehn vor sieben bitte.
ich möchte gerne einen tisch um zehn vor sieben.

ich möchte einen tisch um sieben uhr fünfzig.
könnte ich bitte einen tisch um sieben uhr fünfzig haben.
ich möchte einen tisch um sieben uhr fünfzig bitte.
ich möchte gerne einen tisch um sieben uhr fünfzig.

ich möchte einen tisch um fünf vor sieben.
könnte ich bitte einen tisch um fünf vor sieben haben.
ich möchte einen tisch um fünf vor sieben bitte.
ich möchte gerne einen tisch um fünf vor sieben.

ich möchte einen tisch um sieben uhr fünfundfünfzig.
könnte ich bitte einen tisch um sieben uhr fünfundfünfzig haben.
ich möchte einen tisch um sieben uhr fünfundfünfzig bitte.
ich möchte gerne einen tisch um sieben uhr fünfundfünfzig.

ich möchte einen tisch für sieben uhr reservieren.
kann ich einen tisch für sieben uhr dreissig reservieren.
ich hätte gerne einen tisch für viertel vor sieben reserviert.
ich möchte gerne einen tisch für sechs uhr fünfundvierzig reservieren.
ich möchte einen tisch für eine person.
könnte ich einen tisch für eine person haben.
haben sie einen tisch für vier auf der terrasse bitte.
ich möchte einen tisch für zwei.
ich hätte gerne einen tisch für zwei.
könnte ich einen tisch für drei haben.
haben sie einen tisch für vier.
ich möchte gerne einen nichttrauchertisch für eine person.
ich möchte gerne einen tisch für eine person.
haben sie einen nichttrauchertisch für eine person.
haben sie einen tisch für eine person.
einen tisch für zwei bitte.
einen tisch für zwei bitte.
könnte ich einen nichttrauchertisch für drei Personen haben.
könnte ich einen tisch für drei Personen haben.
ich hätte gerne einen tisch für drei Personen.
ich hätte gerne einen tisch für vier.
haben sie einen tisch für vier.

haben sie einen tisch draussen.
ist draussen noch ein tisch frei.
könnten wir einen tisch beim fenster haben.
haben sie einen tisch in der Nähe des fensters.
könnte ich einen tisch in der ecke haben.
kann ich einen tisch in der ecke haben.
ich hätte gerne einen tisch im raucherabteil.
ich hätte gerne einen tisch im nichtraucherabteil.
raucher bitte.
nichtraucher bitte.

ich möchte um viertel vor acht einen tisch für vier Personen auf der terrasse.
könnte ich morgen um viertel nach sieben einen nichttrauchertisch für drei Personen in der Nähe des fensters haben.
ich hätte gerne einen tisch für drei Personen im raucherabteil.
ich möchte gerne einen tisch für zwei um sieben Uhr fünfzehn reservieren.
ish möchte gerne einen tisch für sechs um viertel nach sieben reservieren.
kann ich einen tisch für zwei Personen um acht Uhr reservieren.
haben sie einen tisch für zwei um halb acht.
haben sie einen tisch für zwei um sieben Uhr dreissig.
haben sie draussen einen tisch für zwei um halb neun.
haben sie draussen einen tisch für zwei um halb eins.
haben sie draussen einen tisch um fünf vor halb zwei.
haben sie draussen einen tisch um fünf nach halb zwei.
haben sie draussen einen tisch um vier teil nach zwölf.
haben sie einen nichttrauchertisch für eine Person um viertel vor sieben.
haben sie einen tisch für eine Person um viertel vor sieben.
kann ich einen tisch für morgen abend um sieben Uhr reservieren.
ish möchte gerne einen tisch für morgen um sechs Uhr fünfundvierzig reservieren bitte.
kann ich für morgen um achtzehn Uhr fünfundvierzig einen tisch reservieren bitte.
kann ich für morgen um viertel vor acht einen tisch reservieren.
kann ich für morgen abend einen tisch für zwei reservieren.
sent('kann ich einen tisch um halb sieben reservieren').
sent('kann ich einen tisch um zwanzig nach sieben reservieren').
sent('kann ich einen tisch um viertel nach sieben reservieren').
sent('kann ich einen tisch um viertel vor acht reservieren').
sent('kann ich einen tisch um zehn vor acht reservieren').
sent('kann ich einen tisch für heute abend reservieren').
sent('haben sie für morgen abend um etwa sieben uhr einen tisch für drei').
sent('haben sie für morgen abend um etwa neunzehn uhr einen tisch für drei').

reservation
sent('ich habe eine reservation auf den namen smith').
sent('ich sollte eine reservation auf den namen smith haben').
sent('ich sollte eine reservation auf den namen smith für einen tisch draussen haben').

sent('könnten wir die speisekarte haben').
sent('ich möchte gerne die speisekarte').
sent('könnten sie uns die speisekarte bringen').
sent('könnten wir ‡ la carte essen').
sent('können wir die weinkarte haben').
sent('ich würde mir gerne die weinkarte ansehen').
sent('haben sie eine weinkarte').
sent('dürfte ich die dessertkarte haben').
sent('könnte ich die dessertkarte haben').
sent('würden sie mir diesen wein empfehlen').
sent('welchen wein würden sie empfehlen').
sent('welchen wein würden sie zum kaninchen empfehlen').
sent('welches ist das heutige tagesgericht').

pay
sent('könnten wir die rechnung haben').
sent('könnten wir bitte die rechnung haben').
sent('dürfte ich die rechnung haben').
sent('ich würde gerne zahlen bitte').
sent('könnten sie uns bitte die rechnung bringen').
sent('könnte ich eine quittung bekommen').
sent('ich hätte gerne eine quittung bitte').
sent('dürfte ich bitte eine quittung haben').
sent('die rechnung bitte').
sent('nehmen sie euros').
sent('nehmen sie schweizer franken').
sent('nehmen sie kreditkarten').
sent('kann ich in schweizer franken zahlen').
sent('können wir in euros zahlen').
sent('ich möchte mit der kreditkarte zahlen').
sent('ich möchte in schweizer franken zahlen').
sent('könnte ich zahlen bitte').
sent('dürften wir zahlen bitte').
sent('wir möchten gerne zahlen').

find
sent('wo sind die toiletten').
sent('wo ist das telefon').
B.1.2 Transfer rules for temporal aspects

%% um sieben uhr fünf
macro(n_oclock_to_n_plus_12(N, NPlus12),
    transfer_rule([[plus, [N, Mins]], [zeit,uhr]],
                   [[time, [NPlus12, Mins]]])
).

%% um fünf nach sieben
macro(n_oclock_to_n_plus_12(N, NPlus12),
    transfer_rule([[plus, [N, Mins]]],
                   [[time, [NPlus12, Mins]]])
).

reverse_transfer_rule([[plus, [N, Mins]], [zeit,uhr]],
                     [[time, [N, Mins]]]).

%% Mins to N
macro(n_oclock_to_n_plus_12_minus_1(N, NPlus12Minus1),
    transfer_rule([[moins, [N, Mins]], [zeit,uhr]],
                   [[time, [NPlus12Minus1, @mins_to(Mins)]]])
).

%% fünf vor sieben
macro(n_oclock_to_n_plus_12_minus_1(N, NPlus12Minus1),
    transfer_rule([[moins, [N, Mins]]],
                   [[time, [NPlus12Minus1, @mins_to(Mins)]]])
).

transfer_rule([[moins, [7, 20]]],
               [[time, [18, 40]]]).

@n_oclock_to_n_plus_12(1, 13).
@n_oclock_to_n_plus_12(2, 14).
@n_oclock_to_n_plus_12(3, 15).
@n_oclock_to_n_plus_12(4, 16).
@n_oclock_to_n_plus_12(5, 17).
@n_oclock_to_n_plus_12(6, 18).
@n_oclock_to_n_plus_12(7, 19).
@n_oclock_to_n_plus_12(8, 20).
@n_oclock_to_n_plus_12(9, 21).
@n_oclock_to_n_plus_12(10, 22).

@n_oclock_to_n_plus_12_minus_1(1, 12).
@n_oclock_to_n_plus_12_minus_1(2, 13).
@n_oclock_to_n_plus_12_minus_1(3, 14).
@n_oclock_to_n_plus_12_minus_1(4, 15).
@n_oclock_to_n_plus_12_minus_1(5, 16).
@n_oclock_to_n_plus_12_minus_1(6, 17).
@n_oclock_to_n_plus_12_minus_1(7, 18).
@n_oclock_to_n_plus_12_minus_1(8, 19).
@n_oclock_to_n_plus_12_minus_1(9, 20).
@n_oclock_to_n_plus_12_minus_1(10, 21).
macro(n_oclock_to_n(N),
    transfer_rule([[plus, [N, Mins]], [temporal, heure]],
        [[time, [N, Mins]]]))
).

% Mins to N
macro(n_oclock_to_n_minus_1(N, NMinus1),
    transfer_rule([[moins, [N, Mins]], [temporal, heure]],
        [[time, [NMinus1, @mins_to(Mins)]]])
).

@n_oclock_to_n(11).
@n_oclock_to_n(12).
@n_oclock_to_n(13).
@n_oclock_to_n(14).
@n_oclock_to_n(15).
@n_oclock_to_n(16).
@n_oclock_to_n(17).
@n_oclock_to_n(18).
@n_oclock_to_n(19).
@n_oclock_to_n(20).
@n_oclock_to_n(21).
@n_oclock_to_n(22).
@n_oclock_to_n(23).

@n_oclock_to_n_minus_1(11, 10).
@n_oclock_to_n_minus_1(12, 11).

macro(mins_to(1), 59).
macro(mins_to(2), 58).
macro(mins_to(3), 57).
macro(mins_to(4), 56).
macro(mins_to(5), 55).
macro(mins_to(6), 54).
macro(mins_to(7), 53).
macro(mins_to(8), 52).
macro(mins_to(9), 51).
macro(mins_to(10), 50).
macro(mins_to(11), 49).
macro(mins_to(12), 48).
macro(mins_to(13), 47).
macro(mins_to(14), 46).
macro(mins_to(15), 45).
macro(mins_to(16), 44).
macro(mins_to(17), 43).
macro(mins_to(18), 42).
macro(mins_to(19), 41).
macro(mins_to(20), 40).
macro(mins_to(21), 39).
macro(mins_to(22), 38).
macro(mins_to(23), 37).
macro(mins_to(24), 36).
macro(mins_to(25), 35).
macro(mins_to(26), 34).
macro(mins_to(27), 33).
macro(mins_to(28), 32).
macro(mins_to(29), 31).
macro(mins_to(30), 30).
macro(mins_to(35), 25).
## C.1 Evaluation Comments

### C.1.1 CALL-SLT French-English

<table>
<thead>
<tr>
<th>Recommandations pour améliorer le logiciel :</th>
<th>Recommandations pour rendre le logiciel plus efficace pour vos besoins :</th>
</tr>
</thead>
<tbody>
<tr>
<td>Il faudrait pouvoir isoler la voix des bruits extérieurs qui entrave à l'identification des mots prononcés.</td>
<td>Améliorer la qualité de compréhension.</td>
</tr>
<tr>
<td>Problèmes de compréhension de l'ordinateur...mais bonne idée dans l'ensemble!</td>
<td></td>
</tr>
<tr>
<td>Parfois le logiciel reprend des mots qui n'ont pas du tout été dites. Il a même accepté des fautes comme par exemple : &quot;I would like two of pizza&quot;...</td>
<td></td>
</tr>
<tr>
<td>L'ordinateur ne me comprenait pas bien et pas toujours.</td>
<td></td>
</tr>
<tr>
<td>Meilleure compréhension de la part de l'ordinateur.</td>
<td></td>
</tr>
<tr>
<td>Le système est pas mal, mais au début, il ne comprenait rien du tout.</td>
<td></td>
</tr>
<tr>
<td>C'était bien car on apprenait des phrases usuelles, peut-être qu'il pourrait être un peu plus dur sur des thèmes un peu plus spécifiques...</td>
<td></td>
</tr>
<tr>
<td>Il faudrait qu'il comprenne mieux ce qu'on veut dire.</td>
<td></td>
</tr>
<tr>
<td>Quelques petits problèmes encore dans la reconnaissance vocale, mais c'est un bon logiciel quand même. Merci.</td>
<td></td>
</tr>
<tr>
<td>Etre plus souple dans la terminologie utilisée.</td>
<td></td>
</tr>
<tr>
<td>Exemple, &quot;fried potatoes&quot; n'est pas accepte comme traduction de &quot;fruits&quot;. OK, pas une réponse &quot;juste faux&quot; est trop limitatif. Une sorte de dialogue &quot;ordinateur&quot; - &quot;humain&quot; serait plus profitable. Comme écrit ci-dessus, j'apprécierais plus un dialogue que des phrases isolées.</td>
<td>Je suis conscient que c'est plus compliqué à réaliser.</td>
</tr>
<tr>
<td>Un peu plus de vocabulaire car finalement on ne connait pas beaucoup de mots</td>
<td></td>
</tr>
<tr>
<td>Je pense que les phrases devraient être plus variées, on commence toujours par : I would like...</td>
<td></td>
</tr>
<tr>
<td>Je trouve qui ne sert a rien car les phrases sont toujours les mêmes phrases il voudrait varier en tel sorte d'obtenir diverses tournure de phrase</td>
<td></td>
</tr>
<tr>
<td>Améliorer la reconnaissance de la voix et ajouter des manières de dire certaines phrases.</td>
<td></td>
</tr>
</tbody>
</table>
L’ordinateur ne reconnait pas facilement ce que l’on dit, ce qui est dommage, car il comprend de temps à autres des mots complètement différents, comme par exemple Tandoori & Satay...il les confond.

| Les mots que l’ordinateur trouve sont parfois très différents de ce qui est dit | Pas d’idée |
| Améliorer la reconnaissance vocale =) | Mettre des images sinon c’est trop triste =_(... Fail |
| Plus de vocabulaire et phrases moins répétitives. Plus de grammaire. | Plus de vocabulaire |
| Il serait peut-être intéressant de tenter d’affiner le niveau de compréhension du micro. | Axer le logiciel sur le vocabulaire. |
| Axer le logiciel sur le vocabulaire. | Pouvoir re-écouter ce qu’on a dit. |
| Contrôler les correspondances entre les exemples écrits et les exemples audio. Ça peut arriver qu’on ne sache pas quelle version l’ordinateur s’attend (enrichir les "connaissances" du logiciel) | |
| Moins demander sur le restaurant et les aliments mais aussi sur d’autres sujets. | |
| Améliorer le système de reconnaissance vocale car je n’ai pas toujours été compris par l’ordinateur. Néanmoins, ce programme est utile pour s’entraîner a parler clairement et distinctement, mais il n’apprend pas grand chose (vocabulaire de la nourriture). | |
| Peut-être faut-il avoir des micros plus performants. | |
| Il faudrait que le logiciel identifie plus précisément ce que nous disons. | |
| Le logiciel a du mal à comprendre ce qu’on dit, même si ce qu’on dit est juste! | |
| Améliorer la reconnaissance vocale | Il faudrait que le logiciel nous comprenne mieux... |
| Si améliorer la reconnaissance vocale est impossible, il faut au moins préciser de parler très clairement. Il faut aussi rajouter des possibilités de réponses. | |
| Thanks a lot | Thanks a lot |
| Pour améliorer le logiciel, il faudrait que le système "interprète" mieux nos dires. | Proposer un plus large choix de thèmes. |
L’aide est peut-être trop précise, ce qui ne nous aide pas forcément pour l’approfondissement du vocabulaire. Il faudrait pouvoir choisir de notre plein gros les mots dont nous souhaiterions avoir la traduction. La reconnaissance vocale est bonne mais il m’a fallu tout de même répéter plus lentement à plusieurs reprises, car je parlais un peu trop vite. De plus, on nous compte parfois des mots justes alors que nous ne les avons pas prononcées. Il serait donc peut-être possible d’améliorer la reconnaissance vocale.

Il est dommage que l’aide nous donne la réponse. Elle devrait nous indiquer uniquement les mots de vocabulaire utile pour la phrase. Il n’est question dans l’exercice que je viens de faire de faire "des phrases de manière polies", il serait plus intéressant de varier la manière.

Probablement que l’ordinateur est sensible à l’intonation des phrases, mais nous ne sommes pas mis au courant de nos fautes d’intonations, bien que je pense que cela soit difficile à programmer.

Ce logiciel reste intéressant et ludique, malgré quelques rates incompréhensibles.

| Varier les sujets, et les types de phrases. |
| Je n’ai pas pu faire l’exercice, c’est pourquoi je n’ai pas d’avis. |
| Le logiciel ne reconnaît pas les déterminants. |
| Il ne me comprend pas bien. Il interprète mal ce que nous disons. |
| J’aurai bien voulu qu’une voix me corrige pour la prononciation. |
| Se baser aussi sur la grammaire (les temps des verbes, verbe-to or -ing...) |
| More grammatic exercises or exercises with verbs. |
| Il faudrait qu’il souligne le mot mal prononcé dans la phrase dite par l’utilisateur et non mettre une autre phrase lorsqu’elle a mal été prononcée. |

Au niveau de la prononciation, l’ordinateur déforme encore beaucoup de mots.
<table>
<thead>
<tr>
<th>Améliorer la sensibilité de la reconnaissance vocale :)</th>
<th>Je trouve dommage que le logiciel oriente notre parole vers le vocabulaire spécifique de la leçon.</th>
</tr>
</thead>
<tbody>
<tr>
<td>No sound coming out for the examples.</td>
<td>Pouvoir écrire ce que l’on veut prononcer.</td>
</tr>
<tr>
<td>Quand on essaye de dire &quot;roast beef&quot; et que l’ordinateur comprend &quot;a plate please&quot;... C’est pas génial génial... Par contre quand au lieu de soy sauce, on essaye avec soja sausage, et que l’ordinateur nous dit que c’est juste, parce qu’il a entendu soy sauce please... pas super non plus...</td>
<td>Améliorer la sensibilité de la reconnaissance vocale :)</td>
</tr>
<tr>
<td>Je pense qu’il faudrait plus d’interactivité.</td>
<td>Je trouve dommage que le logiciel oriente notre parole vers le vocabulaire spécifique de la leçon.</td>
</tr>
<tr>
<td>Insérer une fenêtre de vocabulaire pour permettre de dire la phrase demandée même si on ne connaît pas une mot pour ne pas bloquer le processus de progression.</td>
<td>Une aide plus partielle et des phrases plus diverses rendraient le jeu plus intéressant et efficace.</td>
</tr>
<tr>
<td>Améliorer l’aide en plusieurs étapes.</td>
<td>Il faudrait pouvoir isoler la voix des bruits extérieurs qui entrave à l’identification des mots prononces.</td>
</tr>
<tr>
<td>Troubles de compréhension de l’ordinateur...mais bonne idée dans l’ensemble!</td>
<td>Meilleur reconnaissance vocale, et meilleure explication des fautes de prononciation/grammaire/ etc.</td>
</tr>
</tbody>
</table>