



UPPSALA
UNIVERSITET

Department of Linguistics and Philology
Språkteknologiprogrammet
(Language Technology Programme)
Master's thesis in Computational Linguistics
22 november 2006

Development of a Speech-Driven Automatic Telephone Service

Retrieving pronunciation and spelling of names

Marie Larsson

Supervisors:
Beata Megyesi, Uppsala Universitet
Rich Schulman, SpeechCraft
Mathias Johansson, SpeechCraft

Abstract

This thesis describes the development and evaluation of a speech-driven telephone application for Speechcraft (a company developing speech-driven solutions). The application is created to allow companies to let Speechcraft know the pronunciation along with the spelling of the names of their employees. The names and the spelling are needed to record prompts when Speechcraft creates an automatic telephone receptionist for the company. The user will submit the written names on a website and then call the application to record the corresponding names. The utterances from the user are automatically recognized and interpreted for the application to be able to answer the user.

The application was tested by twenty users recording a number of names. The results was evaluated using the PARADISE framework, including a survey. The satisfaction score resulted in 86%, which is comparable to other similar systems.

Contents

Abstract	2
Acknowledgements	5
1 Introduction	6
1.1 Purpose	6
1.2 Outline	7
2 Background	8
2.1 Spoken Dialogue Systems	8
2.1.1 Speech Recognition	9
2.1.2 Dialogue Management	9
2.1.3 Output Generator	11
2.2 VoiceXML	12
2.3 PHP	14
2.4 PARADISE	14
3 Implementation	16
3.1 Designing the Dialogue	16
3.2 Programming the Static Dialogue	17
3.2.1 Recording	17
3.2.2 Grammars	18
3.2.3 Error Handling	19
3.3 Integrating Dynamics	20
3.3.1 Saving the Audio File	21
3.4 The Web Interface	21
4 Evaluation	23
4.1 Method	23
4.1.1 Data Collection	23
4.1.2 Test Users	23
4.2 Results	24
5 Discussion	26
5.1 Future Improvements	27
6 Conclusions	29
Bibliography	30

Acknowledgements

I would like to thank my supervisor, Beata Megyesi at the Department of Linguistics and Philology at Uppsala University, for her encouragement and her constructive advice in the work of this thesis.

Further, I would like to thank everyone at SpeechCraft, especially Rich Schulman and Mathias Johansson, for the idea for this project and for creative suggestions and reliable support throughout the project.

I would also like to thank all the test users for making it possible to evaluate the result of my work.

Finally, thank you to all my friends and family, especially Göran Wallin, for his support, motivation and feedback.

1 Introduction

Communicating with computers through speech is getting more and more common in our every day life. Today we are able to use our voice to tell a computer what to do, and the computer can answer us using speech. Systems that communicate with humans through spoken human language are called spoken dialogue systems. Potential benefits of spoken dialogue systems are remote access from any phone, ease of use, efficiency and constant availability. The goal when developing a dialogue system is to make the interaction between human and computer as natural as possible, as if the user is talking to a human agent.

These days, we can come across a spoken dialogue system when booking a plane ticket, calling a telephone exchange or maybe even when visiting a museum. For example, when calling the Swedish railway company you will talk to a spoken dialogue system. The spoken dialogue system can be there to help us carry out a task, to educate us or simply to entertain us. These type of systems are currently employed by many companies and institutions to provide telephone-based information automatically.

A spoken dialogue system contains speech recognition, for "understanding" the user, a dialogue manager that decides what to do with the recognized speech and synthetic speech for using speech to respond to the user.

1.1 Purpose

The purpose of this thesis is to develop a speech-driven telephone service. This service will be an addition to an already existing speech-driven telephone exchange developed by Speechcraft. The telephone exchange is used by companies. People calling to the companies will, by this automatic telephone exchange, be connected to the person they want to speak to. The purpose of this new module is to let the companies that use the telephone exchange add names to the list of people a caller can be connected to, by recording the names. The user will also, through a web interface, submit the spelling of the names. This will result in retrieving both the spelling and the pronunciation of the names. This will help Speechcraft to know how to pronounce the names when they need a recording of the names. It can also be used as a preliminary solution until Speechcraft have the possibility to record the names in a studio. The recording is used to let the caller know that they are being connected to the right person.

1.2 Outline

This paper describes the development of a spoken dialogue system. Chapter 1 gives an introduction to the subject and describes the purpose of the thesis. In chapter 2, the background on dialogue systems are presented and there is an introduction to the programming languages used. The implementation of the telephone service is described in chapter 3. Chapter 4 covers the method for the evaluation and the result of the evaluation. In chapter 5, there is a discussion about the results and future improvements are suggested. The last chapter concludes the thesis.

2 Background

This chapter contains discussions of theories and different components in spoken dialogue systems. The programming tools, VoiceXML and PHP, that are used to develop the application are also described in this chapter.

2.1 Spoken Dialogue Systems

Vocal communication between human and computer contains text-to-speech (TTS) synthesis and automatic speech recognition (ASR). A dialogue system is a system that makes vocal interaction between human and computer possible. To be able to call a system a dialogue system it should be able to handle longer dialogues, not only one question followed by an answer. When the interaction is in the form of speech, it is called spoken dialogue systems. In spoken dialogue systems the input from the user is translated into text that the computer can parse by automatic speech recognition (ASR). The output from the recognizer is sent to a component that interprets the semantic meaning of the input. The interpretation is then used by the dialogue manager to determine what to do. Finally, the system communicates through spoken output, using either speech synthesis or a recorded prompt. A simple question/answer dialogue system can be summarized in three main components, a speech recognizer, a dialogue manager and an output generator. The components are illustrated in Figure 2.1.

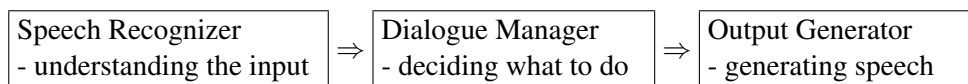


Figure 2.1: The components of a spoken dialogue system [Gustafson, 2002]

In every state in the dialogue either the user or the system has the initiative. When the same part controls the dialogue all the way through it is called single initiative. If the dialogue initiator changes over time it is called mixed initiative. If the model determines who has the initiative it is called fixed mixed, and dynamic mixed is when both dialogue parts can take the initiative at any time [Gustafson, 2002]. The most common type of spoken dialogue system uses system initiative. The structure of the application is what determines the structure of the dialogue when using system initiative. In that type of dialogue there are predefined slots that are filled by the users or the users are prompted with menu choices.

2.1.1 Speech Recognition

ASR is a technology that allows a computer to identify the words that a user speaks into a microphone or a telephone. The design of ASR has proved to be less successful than the design of TTS [O'Shaughnessy, 2000]. This is due to the many variabilities that can arise in producing and interpreting speech. For example, when developing an ASR system, one must take into consideration possible background noise and channel noise. For example, if the speaker is using a telephone and is placed in a noisy surrounding or if the connection is bad, they would want the system to function anyway.

One other difficulty is the variation between speakers. The computer must be adapted to different voices and dialects as input. This problem can be avoided by using a speaker-dependent (SD) system. SD systems only accept voices of speakers who have previously trained the system. These type of systems are for example common in mobile phones. If a caller has trained the system on the mobile phone to recognize his voice, when uttering a specific name, he can just say the name of the person he wants to call and the mobile phone will call that person's number. If one wants this type of system to be able to adapt new users, the new users' speech patterns must be entered. This means that the memory and training time in SD systems grow with the number of speakers.

Therefore it is better to use a speaker-independent (SI) system if the system will be used by a large number of users. Such a system is less accurate than the SD system because of the variabilities in different speakers. These differences among speakers create inter-speaker variability, the main difficulty for SI systems. For example, a human can understand other humans even if they speak a different dialect of the same language, so that is also demanded by the computer. Intra-speaker variability is another complexity when developing automatic speech recognition systems. It refers to differences within each speaker's utterances. A speaker will never say exactly the same thing twice. Understanding and taking into consideration all these different variabilities is a key to developing an automatic speech recognizer.

When the speech has been accepted and actual text can be extracted, the next step is to parse the input. An ASR system uses a grammar to recognize the words uttered by the user and the most similar match will be chosen. The grammar can be designed in several ways. Some systems use statistical grammars generated from large corpora of speech data. Other grammars are rule-based, where the best match for the input is searched for in a number of unchanging utterances.

After the speech has been recognized it is up to the dialogue manager to decide what to do and how to react to the user's input.

2.1.2 Dialogue Management

When building a spoken dialogue system one of the main difficulties is to anticipate how the users will speak to the system. The users' way of speaking will, however, be affected by the functionality of the system. Therefore, it can be a good idea to do user studies before deciding on the design of the dialogue system. There are a few different ways of dealing with the problem of predicting how the user will interact with a spoken dialogue system. Design by inspiration, design by observation and design by simulation [Gustafson, 2002]. When designing by inspiration the system designer uses his or her linguistic intuition to decide how to develop the system. The theory

behind that way of designing is that humans are experts in human language. Since the developer is a human he or she will, to some extent, be able to predict what the user will say to the system. However, it is usually not possible for the designer to think of all possible situations in advance. Designing by observation is when the designer observes human-human dialogues. The developer will see how humans chose to solve a specific task. If it is not possible to collect human-human dialogues one could also design by simulation. This method is called the Wizard-Of-Oz (WOZ) technique. In this technique some or all parts of the system are simulated so that the users think that they are using a fully automated system. That will make the user interaction more realistic.

In the conversational structure of human-human communication the main feature of a dialogue is that two or more parts contribute to the discourse. This feature is sometimes called the "chaining principle" [Good, 1979]. In this principle a dialogue is something consisting of a number of turns. Every turn is composed by smaller turn construction units consisting of sentences, clauses, phrases or words. The place where completion of a turn construction unit (TCU) is possible becomes a transition relevance place (TRP). When TCU reaches a TRP transfer of speakership may occur. The speakers can indicate when they are done using cue words, intonation, a longer pause or other prosodic cues. In that way the listeners know when it's their time to speak through recognizing TRPs. The study of TRPs can be especially interesting when developing spoken dialogue system, since the system needs to understand when a pause is a TRP or not.

The dialogue manager can be designed in different ways according to what the goal with the system is. The most common way to design a spoken dialogue system is the so called task-oriented design [Gustafson, 2002]. When a system is task-oriented the goal is to simplify the task that the user is carrying out. It can for example be a system for ordering tickets or for ordering goods. Most spoken dialogue systems are task-oriented because it makes it easier to build a dialogue manager since the language models and the dialogue rules are easier to pre-define. Another advantage of the task-oriented systems is that they often keep the turns quite short. Furthermore, evaluation of the system is easier, since the user has a well-defined goal, it is possible to determine if the dialogue was successful.

Other goals with a spoken dialogue system can for example be to educate or tutor the user, e.g. tourist information systems. Such systems are called explorative. An explorative system is harder to evaluate than the task-oriented system, since the user's goal with the interaction is harder to define. The user's goal is not to solve one specific task but to explore the possibilities.

An additional goal is to entertain the user. These type of dialogues focuses on who the dialogue partner is, objects in the spatial context and the actual situation of the dialogue partner. This kind of system, where the dialogue is created from the context, is called context-oriented systems [Gustafson, 2002]. It can e.g. be an embodied agent in an information kiosk or a character in a computer game, that can engage in a conversation with the user. This type of system is, like the explorative system difficult to evaluate since there is no well-defined goal. Which of these types of systems to use is determined by the goal of the system.

All the aspects mentioned above should be taken into consideration when designing the dialogue manager.

2.1.3 Output Generator

When the system has decided what to do according to what was designed in the dialogue manager it needs to answer the user. The answer can be in the form of a prerecorded audio prompt or text-to-speech synthesis. Prerecorded prompts can be used when the vocabulary is limited. The advantage of recorded prompts is that the user will hear a natural sounding voice as opposed to synthetic speech. When the output from a system is unique the usage of TTS systems is powerful.

Speech synthesis involves the conversion of an input text into speech. The generation of synthetic speech can be characterized as a two-stage process, consisting of analysis and synthesis. The process is illustrated in Figure 2.2.

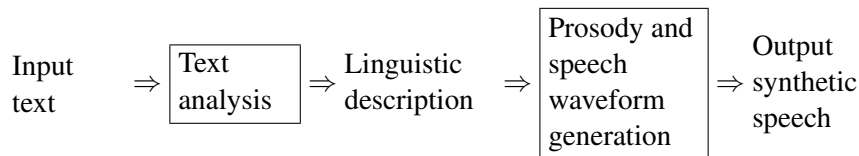


Figure 2.2: The conversion from text to speech as an analysis-synthesis process [Holmes and Holmes, 2001].

The first part of this process involves analysis of the text to determine underlying linguistic structures. The first task in text analysis often involves preprocessing the input text to convert it to a sequence of words. The next step is to convert the sequence of words into a linguistic description. A major function in the analysis process is to determine the pronunciation of the words. The pronunciation is usually obtained by using a combination of a pronunciation dictionary and letter-to-sound rules [Holmes and Holmes, 2001]. The dictionary usually contains a large number of words. These known words will thus be pronounced correctly, given the correct linguistic analysis.

However, there must be rules for handling unknown words as well. To determine word pronunciation most TTS systems include morphological analysis. This analysis determines the root of every word to avoid the need to include all derived forms in the dictionary. Syntactic analysis of the text is also needed to determine the pronunciation of some words. When pronunciation of the single word has been determined it is also necessary to take the pronunciation that occurs across word boundaries into consideration. Analysis of word-level stress and sentence-level stress is added to generate the prosody for the synthesized speech. If both the pronunciation of the word spoken in isolation and the word spoken in context are dealt with, the syntactic speech will sound more natural.

The second part of the TTS conversion process is to generate synthetic speech from the linguistic description. The information from the text analysis is used to generate prosody for the utterance. The synthetic speech is generated in different ways depending on what kind of synthesis is used, concatenation synthesis or formant synthesis.

When using concatenation synthesis synthetic speech is generated through concatenating small stored prerecorded speech units, e.g. phones and diphones. Diphone synthesis is one of the most popular methods when creating synthetic speech from recordings, or samples of a particular person. The advantage of using a diphone (i.e. two adjacent half-phones) is that the center of a phonetic realization is the most stable region. The diphone cuts the speech units at the points of relative stability, rather than at the volatile phone to phone transition, where coarticulatory effects appear.

This means that a diphone is easier to model than the transition from one segment to another [Black and Lenzo, 2000].

In formant synthesis one does not use human speech samples. Instead, the synthetic output is created using an acoustic model. A number of parameters, such as formant frequencies and bandwidth for the first 4 or 5 formants, are specified to create a waveform of artificial speech.

2.2 VoiceXML

VoiceXML (Voice Extensible Markup Language) is a markup language for creating voice user interfaces that use automatic speech recognition and text-to-speech synthesis. Just as HTML is a markup language for describing text, VoiceXML is a markup language to describe dialogues between humans and computers, via any device that has sound as input and output, such as a standard phone. A number of large corporations have been working towards the integration of telephony and Internet technologies for many years. This finally led to the creation of VoiceXML. The VoiceXML 1.0 specification was released in May 2000 [Andersson et al., 2001].

When browsing with VoiceXML the user makes a call and the telephone gets connected to the VoiceXML gateway. The VoiceXML gateway contains a number of subsystems [Abbott, 2002]. These subsystems are:

- *Network Interface*: Enables HTTP communications with Web servers
- *VoiceXML Interpreter*: The software that carries on a conversation with the user in the way that is specified in a VoiceXML document
- *TTS*: Translates text to spoken words as described in section 2.1.3
- *Audio*: Plays recorded audio prompts to the user and records user input
- *ASR*: Translates user utterance into text as described in section 2.1.1
- *Dual Tone Multi-Frequency (DTMF)*: Allows the user to use the keypad as input through translating keypad input into characters
- *Telephony Interface*: Enables communication with the telephone network

These subsystems are illustrated in Figure 2.3.

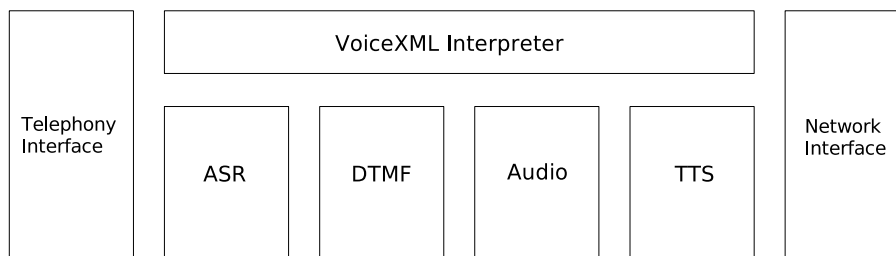


Figure 2.3: The subsystems of the VoiceXML gateway [Abbott, 2002]

A dialogue created in VoiceXML consists of speech from the VoiceXML interpreter and the person at the other end of the telephone call. The person and the

VoiceXML interpreter take turns speaking and listening. What the VoiceXML interpreter and the user can say in the dialogue is determined by what is specified in the VoiceXML documents. This is decided in a number of elements in the VoiceXML document. One such element is the grammar. The grammar contains a set of predefined responses that the user might say. When the user says one of the responses in the grammar it can be recognized by the computer. A grammar is the primary input to the voice recognition technology that underlies the VoiceXML interpreter.

An example of the simplest implementation in VoiceXML, resulting in speech output from the VoiceXML interpreter, can be seen below.

```
<?xml version = "1.0"?>
<vxml version ="2.0">
  <form>
    <block>
      Hello World!
    </block>
  </form>
</vxml>
```

This piece of code will produce the speech output "hello world".

A short dialogue where the system asks a question, listen for the response and act accordingly is displayed below.

```
<?xml version="1.0"?>
<vxml version="2.0">
  <menu id="Question">
    <prompt> Which of the following is not an animal?
      <enumerate/></prompt>
    <choice next="#IncorrectAnswer">
      dog
    </choice>
    <choice next="#IncorrectAnswer">
      cat
    </choice>
    <choice next="#CorrectAnswer">
      strawberry
    </choice>
    <choice next="#IncorrectAnswer">
      cow
    </choice>
  </menu>
  <form id="CorrectAnswer">
    <block>
      That is the correct answer.
      <disconnect/>
    </block>
  </form>
  <form id="IncorrectAnswer">
    <block>
```

```
        That is not the correct answer.  
    <disconnect/>  
  </block>  
</form>  
</vxml>
```

This dialogue sounds according to the following:

System: "Which of the following is not an animal? Dog, cat, strawberry, cow"

User: "dog"

System: "That is not the correct answer."

Or it could go like this:

System: "Which of the following is not an animal? Dog, cat, strawberry, cow"

User: "strawberry"

System: "That is the correct answer."

VoiceXML documents, audio files, command grammars etc. are all static content in a VoiceXML application. Once deployed, this content will not change. Thus, to make the application more powerful addition of dynamics is needed. Dynamic content can for instance be generated using a scripting language such as PHP.

2.3 PHP

PHP is especially suited for dynamic web development and is often embedded into HTML, but it can also be embedded into VoiceXML. PHP was originally a shortening for *Personal Home Page tools*, but these days the abbreviation is a recursive acronym for *PHP: Hypertext Preprocessor* [Overgaard et al., 2004]. PHP is an open source scripting language mainly used for developing web based software applications. The first version of PHP was developed in 1994 by Rasmus Lerdorf [Lerdorf, 1998].

When PHP code is embedded into HTML the code is inserted into the HTML of a web page. When a PHP page is accessed the PHP code is parsed by the server the page resides on. The output from the PHP functions on the page are typically returned as HTML code which can be read by the browser. Because the PHP code is transformed into HTML before the page is loaded users cannot view the PHP code on a page.

PHP is mainly focused on server-side scripting, allowing the user to create dynamic content and interact with databases. To make server-side scripting with PHP work three things are needed, the PHP parser, a web server and a web browser.

2.4 PARADISE

The PARADISE (PARAdigm for DIAlogue System Evaluation) method was proposed by Walker et al. [1997b] as a potential general methodology for evaluating spoken dialogue systems. The PARADISE framework models an overall performance result for a spoken dialogue system as a linear combination of measures reflecting task success and dialogue costs [Walker et al., 1997b]. This is illustrated in Figure 2.4.

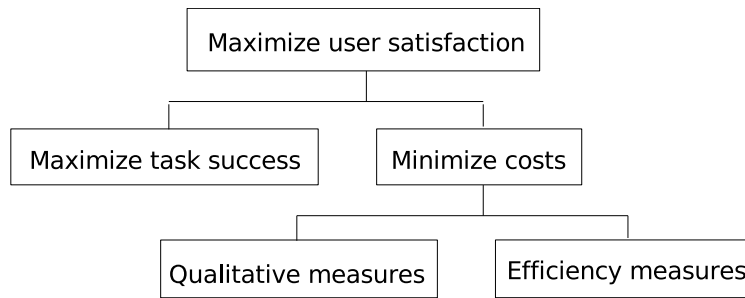


Figure 2.4: PARADISE's structure of objectives for spoken dialogue performance [Walker et al., 1997b]

The top-level objective, user satisfaction, depends on two potential contributors, namely task success and dialogue costs. Task success include whether the user was able to complete the task or not, while dialogue costs for example cover number of utterances to completion of the dialogue and number of times the user has to repeat an utterance. Dialogue costs are in turn dependent on dialogue efficiency and dialogue quality. While the efficiency measures are intended to illustrate how efficient the system is in helping the user to complete a task, the purpose of the qualitative measures is to capture the features of the system that affect the users' subjective ratings.

A general evaluation framework requires a task representation that decouples *what* is accomplished from *how* it is being done [Walker et al., 1997b]. This makes it possible to compare different dialogue strategies.

3 Implementation

The application records and saves user utterances of names during a phone call. When using the application the user must first submit a list of the names that will be recorded. A website is constructed to make it possible to submit the list. On the website one can fill in the names in a field or upload a file containing the names. Retrieving a written list of the names makes it possible to give the recorded files the right name. This way the system will generate both the pronunciation and the spelling of a name.

For the system to be able to understand the user input, the speech recognizer within VoiceXML is used. The output from the system are prerecorded audio prompts. The dialogue manager of the system was primarily implemented using VoiceXML. PHP was also used to make the system more powerful and dynamic.

3.1 Designing the Dialogue

Before the actual implementation of the code can begin, the dialogue must be designed. Since this is a simple system the dialogue is designed by inspiration. The goal with this system is to help the user to carry out a specific task, a so called task-oriented system, as discussed in section 2.1.2. The dialogue is designed to be single initiative, where the initiator is always the system. The system asks the user questions and predefined slots are filled with the user's answers. Since the transition relevance places in this system are at the end of a question or a request from the system, for instance "Say name number one", they will be easily detected by the users. Therefore, we suppose that the users will know when it is their turn to speak.

The dialogue is designed to be as efficient as possible, and to have as few turns as possible. In some cases the users might have a long list of names to add to the system. In such a case the users will want their task to be carried out as fast as possible. If the user still thinks that the dialogue takes too much time, the user have the possibility to interrupt the system when the system is speaking. If the user already knows what the system will say there is no need to listen to the whole sentence.

The whole dialogue can be contemplated upon as if in a flow chart, as seen in Figure 3.1. The first step in the dialogue is for the system to ask the user to record the first name. After the system has said this first request there is a predefined slot for the user to fill, a transition relevance place. The user will then tell the first name on the list that has been submitted through the website. When the user speaks, the utterance will be recorded. After the user has told the name the system will play the recording to the user and ask if the user wishes to keep that recording. The next step in the dialogue depends on whether the user is satisfied with the recording or not. If the users answers "no" the system will again ask the user to record the first

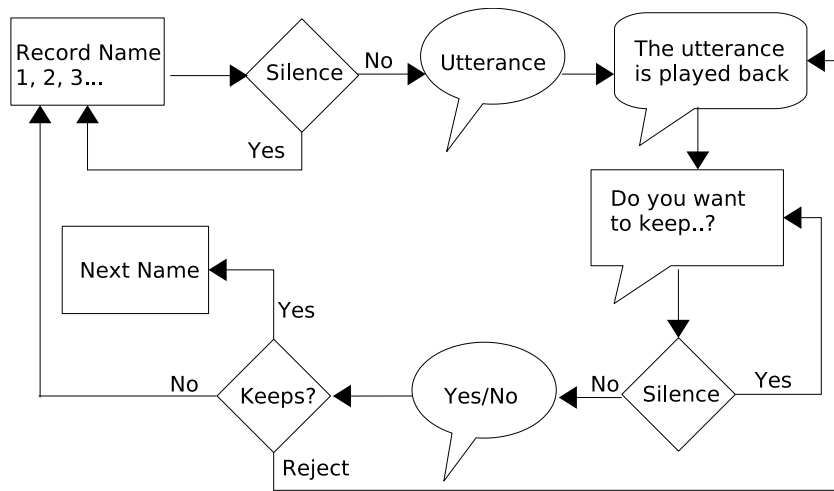


Figure 3.1: Flow chart of the dialogue

name. If the user instead answers "yes", the system will ask the user to record the second name. To make the dialogue as efficient as possible the system will only ask the whole question the first time. All other times the system will only say "Name" followed by the number of the name. The same thing is done when the system asks the user about saving the recording. Only the first time the whole sentence will be prompted. The other times the system will say "Save?". After the first round the user will know how the system works, which makes it possible to shorten the sentences and make the dialogue more efficient.

This dialogue will go on until the list of names is completed or until the user wishes to abort the call. When the last name on the list has been recorded the system will inform the user that the recordings have been saved and then terminate the call. A dialogue concerning a list of two names can look like in the example in Figure 3.2.

3.2 Programming the Static Dialogue

Once the design of the dialogue is decided, the programming can be initiated. The dialogue was initially statically programmed to see that the dialogue works when running it just once, i.e. there will only be a dialogue concerning one name. This was done using VoiceXML exclusively. In this static version the program code consists of one document that is followed linearly to record the name the caller wishes to add.

3.2.1 Recording

When a user first calls the application a prerecorded audio prompt will be played. At every occasion where the system speaks, one will be linked to recorded audio files. If the audio file cannot be found, speech synthesis will be used to read a text. The first prompt will urge the user to say the first name. The utterance from the user that is said after this prompt will be recorded. Recording in VoiceXML is made with the *record element*. The recording starts as soon as the system has finished speaking. The *record element* takes a number of attributes. The attribute that is important in this case

System: Säg namn nummer ett.
User: Anders Andersson.
System: "Anders Andersson"
System: Vill du spara inspelningen?
User: Ja.
System: Namn två.
User: Karl Karlsson.
System: "Karl Karlsson"
System: Spara?
User: Ja.
System: Inspe­lingarna har sparats. Hej då!

System: Say name number one.
User: Anders Andersson.
System: "Anders Andersson"
System: Do you wish to keep the recording?
User: Yes.
System: Name two.
User: Karl Karlsson.
System: "Karl Karlsson"
System: Save?
User: Yes.
System: Your recordings have been saved. Good bye!

Figure 3.2: Example dialogue

is *finalsilence*. In this attribute, duration of silence that will terminate the recording, will be specified. The silence in the recorded audio file should preferably be as short as possible, to avoid a long silence when later playing the audio file. Though the silence must not be so short that there is a risk of missing some part of the utterance, consequently the attribute is set to 600 ms

When the user is satisfied with the recording the system will save it as an audio file. This is described in section 3.3.1.

3.2.2 Grammars

As discussed earlier grammars are needed for the system to be able to identify the user speech. The first utterance from the user will be the name the user wishes to record. This utterance will only be recorded and not also recognized, since that combination is not allowed by the platform that is necessary for this application.

The second utterance from the user will be the answer to the question whether to keep the recording or to discard it. To recognize this answer a number of possible answers are stored in a grammar. The grammar for recognizing the answer to this question can be seen in Figure 3.4.

Rules are created to handle the different answers. An answer like "nej" (no) or "fel" (wrong) will return "nej" (no). If the user answer returns "nej" (no) the system will return to the beginning of the dialogue and repeat the utterance "Säg namn nummer ett" ("Say name number one"), and the user gets a new chance to record the

```

<grammar>
  [(ja) {return (ja)}
   (jo) {return (ja)}
   (ja tack) {return (ja)}
   (okej) {return (ja)}
   (jajamen) {return (ja)}
   (nej) {return (nej)}
   (ta om) {return (nej)}
   (backa) {return (nej)}
   (fel) {return (nej)}
   (nej det blev fel) {return (nej)}
   (nej usch) {return (nej)} ]
</grammar>

<grammar>
  [(yes) {return (yes)}
   (yeah) {return (yes)}
   (yes please) {return (yes)}
   (OK) {return (yes)}
   (no) {return (no)}
   (back) {return (no)}
   (wrong) {return (no)}
   (no that is wrong) {return (no)} ]
</grammar>

```

Figure 3.3: The grammar, first for Swedish, then for English

name.

If the answer from the user is "okej" (OK) or "jo" (yeah) the grammar will return "ja" (yes) and the static dialogue will be completed.

Rules are also created to handle the cases where the system can not recognize the user's answer. This is discussed below.

3.2.3 Error Handling

When the user's utterance is not in the grammar it cannot be recognized. This means that when the user answers something other than the words in the grammar the system will fail to recognize the utterance. These situations must be dealt with. The system will in such a case tell the user that the utterance was not understood and then repeat the question. If the user once again answers something the system does not understand the system will try to guide the user to a recognizable answer by saying "svara ja eller nej" (answer yes or no). A conversation like that is shown in Figure 3.4.

Rules for error handling must also be created if the system does not hear any utterance from the user. When it is the user's turn to speak and nothing is said, the system will say "Jag h rde inte" (I did not hear you) and then ask the question again. For an example, see Figure 3.5.

System: "Anders Andersson"
System: Vill du spara inspelningen?
User: Absolut.
System: Jag förstod inte. Vill du spara inspelningen?
User: Visst.
System: Jag förstod inte. Svara ja eller nej. Vill du spara inspelningen?
User: Ja.

System: "Anders Andersson" (Playing the user input back to the user)
System: Do you wish to save the recording?
User: Absolutely.
System: I did not understand you. Do you wish to save the recording?
User: Sure.
System: I did not understand you. Answer yes or no. Do you wish to save the recording?
User: Yes.

Figure 3.4: Example of error handling

System: Sä g namn nummer ett.
User: -
System: Jag hörde inte. Sä g namn nummer ett.
User: Anders Andersson

System: Say name number one.
User: -
System: I did not hear you Say name number one.
User: Anders Andersson

Figure 3.5: Example of error handling

3.3 Integrating Dynamics

To make it possible to go through a list, with any number of names, dynamics must be integrated in the code. This is carried out by using PHP code embedded in the VoiceXml code. The list of names that is submitted will be read by the application. This way it is possible to adjust the number of times the system will ask for a recording. The system will ask the user to record a name as many times there are names in the submitted list. When there are no more names in the list the system will complete the call by saying "The recordings have been saved. Good bye!".

To connect the pronunciation of a name with the spelling of the same name it is important that the user records the names in the right order. Therefore, the system will say "record name number one", "name two", "name three" etc. With the embedded PHP code it is possible to say a different number for every new audio prompt, as mentioned earlier, and the first request from the system can be made more detailed than the following.

3.3.1 Saving the Audio File

When the system says "Record name number one" the user will say the first name in the list. This way, the recording of the name can be saved in the name that is in the submitted list. For instance, the recording of the utterance "Anders Andersson" will be saved as anders_andersson.wav. That will result in having the correct pronunciation and the correct spelling in the same place.

A recorded utterance in VoiceXML is saved while the session is still running. The recording will be destroyed at the same time the dialogue is finished. To keep a specific utterance it must be saved in another file. When the users choose to save their utterance the recording of the utterance will be submitted to a file where it can be saved as an audio file.

3.4 The Web Interface

To make it possible to save a recording in the right name the written names are needed. An easy way to retrieve the names is by letting the user submit a list on a website. The website, created using HTML and PHP, is designed as a complement to the telephone service. Before calling the telephone application the user will login and submit the names on this site.

The username and password used by the user to log in are connected to the name of the company the user is calling for. For example, if a user wants to record a number of names for the company "Bbb", they will login with the login information given to them. When the login is successful a directory will be created in the name "Bbb". The recorded audio files will be saved in this directory. Also added to the name of the directory are the current date and time. This is added to be able to separate different recording sessions from the same company from each other.

When the user is logged in a new site will appear in the window. On this site the user is given the possibility to write the names he wishes to add in a text field or to upload a file containing the names. When the list of names is submitted the whole list will appear on the website, with the names numbered so that it is easier for the user to see which name is "name one", "name two", etc. As soon as the list is submitted the user can call the telephone application and simply follow the directions and say the names that are on the list shown on the screen. When all the names are recorded and the dialogue is completed the user can just hang up the phone and log out from the website. Figure 3.6 and Figure 3.7 illustrate the websites.

Du är inloggad som **Marie**.

Fyll i namnen som ska spelas in, eller ladda upp en fil med namnen. Ett namn (för- och efternamn) per rad.

Figure 3.6: Website for uploading the names

Ring nu upp telefonnumret **08-50520103**.
Knappa in sifferkoden **09107014** när du blir ombedd att ange en åttasiffrig kod.
Den pinkod som därefter efterfrågas är samma pinkod som du använde för att logga in här.
Följ sedan instruktionerna för att spela in namnen.

Namnlista

1. Krister Svensson
2. Terese Forslund
3. Anna Andersson
4. Nicole Jones
5. Erik Larsson
6. Michelle Stephens
7. Jonas Berglund
8. Monica Lindkvist
9. Peter Jonsson
10. Ben Smith

Figure 3.7: Website for when the names are uploaded

4 Evaluation

4.1 Method

Parts of the PARADISE method are used when evaluating the dialogue system created in the work of this thesis. The PARADISE framework does not specify which measures to collect when evaluating a spoken dialogue system. To evaluate this particular system a combination of dialogue quality and efficiency measures is used, as in Kamm et al. [1999]. In their report several different spoken dialogue systems are evaluated and compared in this way.

4.1.1 Data Collection

To be able to collect the data for the quality and efficiency measures, a few different methods are used:

- The users fill out a web page form after completing the task
- The entire dialogue is automatically saved as a text log
- Everything the user says is automatically recorded

The form that is filled out by the test users is the same as in the survey in Kamm et al. [1999], and gives us a measure of user satisfaction. The survey consists of six statements the user will take a position on. There are five possible answers ranging from *I completely agree* to *I do not agree at all*.

From the text log and the recorded prompts it is possible to transcribe the dialogue between the system and the user. When the transcription is made, one can derive performance measures for speech recognition, check whether the user barged in on the system utterance, check the timing of the interaction and calculate the elapsed time of the dialogue. To evaluate the speech recognizer, the number of timeout prompts and recognizer rejections are calculated. Whereas the timeout prompts are played when there is no user response, recognizer rejections will occur when the system's confidence in its understanding of the user utterance is too low. The number of system turns and user turns are also calculated using the text log.

For the evaluation the test users are given the task of submitting a given list of names on a website, and then recording them when speaking to the dialogue system.

4.1.2 Test Users

The intended users for this system are mainly women over the age of thirty with average computer skills. The test users are twenty women ranging from the age of 31 to 63 years.

Table 4.1: Result of the user satisfaction measure

Survey questions	Average score (%)
1. Det var lätt att förstå vad rösten sa. (It was easy to understand what the system said.)	87%
2. Systemet förstod vad du sa. (The system understood what you said.)	83%
3. Takten på samtalet var bra. (The pace of the interaction with the system was appropriate.)	91%
4. Du visste vad du kunde säga vid varje steg av dialogen. (You knew what you could say at each point of the dialogue.)	83%
5. Systemet svarade tillräckligt snabbt. (The system replied fast enough.)	89%
6. Systemet fungerade som du hade förväntat dig. (The system worked the way you expected it to.)	84%

4.2 Results

Of the twenty test users, everyone were successful in completing the task, to the extent that all of them managed to record the ten names they were given. While some of them finished the dialogue without any problems at all and subsequently gave the highest rating on every question in the survey, others had difficulties in one or more situations in the dialogue. Some of the users said that it was not obvious, from the instructions, that they were supposed to save the recordings. Consequently, when the system asked "Do you want to save the recording?", they did not know how to respond. However, they all figured it out eventually and managed to complete the task. One of the users commented that when the system only says "Save?" in the latter part of the dialogue, it was hard to understand what to do.

The results from the web page form can be seen in Table 4.1. Most of the users had no problem understanding the recorded prompts, while two of the users gave a low rating. Overall, 87% thought it was easy to understand the system.

Statement 2 in the survey gives a measure of automatic speech recognition. 83% was happy with the system's understanding of the user. This was the lowest rating in the survey together with the statements concerning user expertise and expected behavior, which was 83% and 84% respectively

91% was satisfied with the interaction pace and 89% with the system response.

During the conversations, seven of the test users experienced recognizer rejections. For a few of them they occurred several times. The overall result for occurrences of recognizer rejections is 0.7 times per dialogue.

Five of the users encountered timeout prompts, one of them three times. One of the users that had the most recognizer rejections was the same person that had the most timeouts. The result of the timeout prompts is 0.35 times per dialogue.

There were not many situations where the user barged in on the system utterance. This only happened for three of the test users, and resulted in the points 0.2.

The number of user turns varied from 20 to 26 with an average of 21.35. The number of system turns was always one more than the number of user turns, consequently they ranged from 21 to 27 with the average 22.35.

Table 4.2: Performance measure means per dialogue

Measure	Average score
Recognizer Rejects	0.7
Timeouts	0.35
Barge In	0.2
User Turns	21.35
System Turns	22.35
Elapsed Time	126 s
User Satisfaction	25.85 or 86%

The time of the conversation varied from 103 seconds to 189 seconds. The average time of the interaction was 126 seconds.

The values for all the responses from the survey were summed, resulting in a user satisfaction measure of 25.85 for each dialogue. Possible results of user satisfaction range from 6 to 30. Table 4.2 summarizes the measures that were collected from the dialogues.

5 Discussion

The most frequently occurring problem during the testing was that the system did not understand the user. In 70% of the dialogues a recognizer rejection occurs. In most of these cases the user has answered "aa" instead of "ja", which results in the system's confidence in its understanding of the user utterance being too low to pass as a "ja" (yes). The text logg shows that the system's confidence is between 34% and 46% in these cases. If "aa" is added to the grammar this problem will be solved.

In two of the cases where recognizer rejections occurred, they occurred because the user barged in on the system utterance. These situations appear when the user makes a too long pause between the first name and the surname. The system is programmed to terminate the recording after a duration of silence of 600 ms. When the recording is terminated the system will continue to the next stage in the dialogue, which is to ask the user to save the recording. However, the system will not ask the question since it is interrupted by the user trying to record the surname. This will lead to the system trying to recognize the surname spoken by the user as a "yes" or a "no". In other words, this causes disorder in the dialogue, but can be avoided either by letting the pause between the names be longer or by not allowing the user to barge in on the system when it is uttering the question for saving.

An additional reason for recognizer rejections was the system hearing its own utterance. One of the users had four recognizer rejects, all of them caused in this way. When expecting an utterance from the user, the system instead hears what it self just uttered. The system will not be able to recognize this utterance since it is not in the grammar. This only happened to one of the users, possibly because of a high volume on the phone.

Four of the users encountered one timeout prompt each. The user that had problems with the system hearing its own utterance experienced three timeout prompts. The rest of the users carried out the task without timeouts. It is harder to clarify the reasons of timeout prompt than recognizer rejections, since it is difficult to know the reason for the user not to speak. Silences are only noted in the text logg and not recorded.

Since the output prompts were made shorter latter in the dialogue there is not really enough time to barge in on the system. The times where barge in occurred were when the system repeated the save question after a recognizer rejection, and, as mentioned earlier, when the user made a longer pause between first name and surname.

The number of user and system turns varied between the users whether they encountered timeouts and recognition rejections or not. They could also vary if the user was not satisfied with their recording and answered "no" to the save question.

The elapsed time of the dialogue varies considerably between the users. Partly because of the same reason as to the variation between turns. However, two dialogues

with the same number of turns can have quite different time length. There is no specific period in the dialogue where the time difference is caused. Every part of the dialogue where the system speaks is a little slower. The average time for a dialogue is 126 seconds.

In spoken dialogue system it is common to have a help request for the user to turn to if there is something they do not understand. This system lacks such a request. If the user have some trouble with the system there is no help to receive.

In Kamm et al. [1999], they compare a number of different dialogue systems, ANNIE [Kamm et al., 1998] and ELVIS [Walker et al., 1997a] among others. ANNIE is tested both by experts and a novices on dialogue system. The results for ELVIS and ANNIE show that the occurrences of recognizer rejections are more common in these systems than in our project. The measures for recognizer rejects for ELVIS are 0.98. The same results for ANNIE tested by experts and ANNIE tested by novices are 1.8 and 6.6 respectively. The reason why our system is getting better results, with respect to this measure, is probably because the only utterances our system needs to recognize are various ways of saying yes and no. The timeout prompt are also more frequent in the other systems. ELVIS 2.24, ANNIE Exps 0.64 and ANNIE Novs 2.4 compared to our 0.35. The other systems also have more barge ins than ours. As mentioned earlier there are few stages in the dialogue were the user have the possibility to barge in.

The user satisfaction value for our system is 25.85 on a scale from 6 to 30. The results for the same value for ELVIS, ANNIE Exps and ANNIE Novs are 28.9, 32.8 and 23.0 on a scale from 8 to 40. The different numbers of questions in the survey makes it harder to compare the values. However, if compared per cent our result is slightly higher than the result for ANNIE Exps, 86% and 82% respectively.

5.1 Future Improvements

There are some suggestions in the discussion for how to avoid some of the problems occurring in the evaluation. Increasing the grammar with commonly used words is an easy way of making an improvement of the speech recognizer.

There are two ways suggested in the discussion to avoid the barge ins that can occur when the user says the surname and the system has moved on to the next stage; either to prolong the time before termination of the recording or to deny barge in in the following stage. If prolonging the duration, it is hard to know were to set the limit so the problem will not occur again, and making a longer duration causes a longer waiting for the user. However, not allowing the user to barge in on the system when it is asking about saving will result in avoiding the occurring disorder. The user will in this way hear the whole question and their own recorded utterance, and can choose to do a new recording if only the first name was recorded.

One of the survey statements that the user gave the lowest ratings was "You knew what you could say at each point of the dialogue". To improve the system a *help request* can be added. If the user have some problem understanding what to do they can say help and get more detailed instructions.

In the beginning of the dialogue, when calling the application, the user is directly thrown into the recording part. A suggestion to avoid the abrupt start is adding a prompt with an opening phrase where the user is welcomed.

If the recorded prompts later are to be used in Speechcraft's telephone exchange as a way of letting the caller know whom they will be connected to, they must be included in that system.

6 Conclusions

The purpose of this thesis was to create a telephone application where companies can record the names of their employees and in that way let Speechcraft know how to pronounce the names. This was implemented using VoiceXML and PHP. When the users call the telephone application they will be asked to say the name they want to record. After the recording the users will hear their own recording and be asked if they are satisfied with it and want to save it. If the users have a list of names to record, the application will go on asking for the next name, if the caller is satisfied with the previous recording, as many times as there are names submitted on the website.

The speech of the caller is identified through the automatic speech recognizer within VoiceXML. Instead of speech synthesis, audio prompts recorded by a human are used to respond to the user.

A website where the user will log in and write down and submit the names to be recorded was created using HTML and PHP. The recordings are subsequently saved in the corresponding name, and will be stored in a catalogue with the name of the company.

The telephone application was tested and evaluated by twenty women ranging from the age of 31 to 63 years. Their task was to log in on the website, submit ten names and then call the application and record the names. The results were evaluated using the PARADISE framework. As part of the evaluation, the test users were given a form to fill in for measuring user satisfaction, which gave the total result of 86%.

Bibliography

- Kenneth R. Abbott. *Voice Enabling Web Applications: VoiceXML and Beyond*. Apress, 2002.
- Eve Astrid Andersson, Stephen Breitenbach, Tyler Burd, Nirmal Chidambaram, Paul Houle, Daniel Newsome, Xiaofei Tang, and Xiaolan Zho. *Early Adopter VoiceXML*. Wrox Press, 2001.
- Alan Black and Kevin Lenzo. Diphone collection and synthesis. 2000.
- C Good. *Language as social activity: Negotiating conversation*. Journal of Pragmatics 3, 1979.
- Joakim Gustafson. *Developing Multimodal Spoken Dialogue Systems Empirical Studies of Spoken Human-Computer Interaction*. PhD Thesis, Kungliga Tekniska Högskolan, 2002.
- Anna Hjalmarsson. Evaluating adapt a multi-modal conversational, dialogue system using paradise. Master's thesis, KTH Royal Institute of Technology, 2002.
- John Holmes and Wendy Holmes. *Speech Synthesis and Recognition*. Taylor & Francis, 2001.
- DreamTech Inc. *VoiceXML 2.0 Developer's Guide*. The McGraw-Hill Companies, 2002.
- Candace Kamm, Diane Litman, and Marilyn A. Walker. From novice to expert: The effect of tutorials on user expertise with spoken dialogue systems. *ICSLP98*, 1998.
- Candace Kamm, Marilyn Walker, and Diane Litman. Evaluating spoken language systems. *AVIOS*, 1999.
- Rasmus Lerdorf. Dynamic web pages with php3. *WebTechniques*, 1998.
- Douglas O'Shaughnessy. *Speech Communications*. IEEE Press, 2000.
- Jörgen Overgaard, Ulrika Eriksson, and Jesper Ek. *PHP programming*. Pagina Förlags AB, 2004.
- Marilyn A. Walker, Donald Hindle, Giuseppe Di Fabbrizio Jeanne Fromer, and Craig Mestel. Evaluating competing agent strategies for a voice email agent. *EUROSPEECH97*, 1997a.
- Marilyn A. Walker, Diane J. Litman, Candace A. Kamm, and Alicia Abella. Paradise: A framework for evaluating spoken dialogue agents. *ACL/EACL*, 1997b.

A Instructions for test users

Instruktioner för testning

Jag heter Marie Larsson och gör mitt examensarbete på Språkteknologiprogrammet. Jag har utvecklat en röststyrd telefontjänst som spelar in uttal av namn. Innan systemet tas i bruk måste det utvärderas. Jag skulle vara mycket tacksam om du kunde medverka i testningen. Det tar ca. 10 minuter och innebär att du läser upp ett antal namn som spelas in. Du är anonym och alla uppgifter om dig kommer att raderas i redovisningen av resultatet.

För att testa systemet följ instruktionerna nedan:

1. Gå in på websidan www.speechcraft.com/vxml/login.php
2. Logga in med ditt efternamn (med inledande stor bokstav) som användarnamn och pinkoden 7313.
3. Skriv in namnen i listan nedan och klicka på skicka.

Namnlista:

Krister Svensson
Terese Forslund
Anna Andersson
Nicole Jones
Erik Larsson
Michelle Stephens
Jonas Berglund
Monica Lindkvist
Peter Jonsson
Ben Smith

4. Ring upp telefonnumret 08-50520103.
5. När du blir ombedd att ange en åttasiffrig kod knappa in sifferkoden 09107014. (OBS! Inloggningen sker på engelska) Den pinkod som därefter efterfrågas är 7313.
6. Följ sedan instruktionerna du får i luren för att spela in namnen i namnlistan.
7. När samtalet är avslutat, fyll i formuläret som finns på websidan och klicka på skicka.
8. Logga ut.

Tack för din medverkan!

Marie Larsson