Mixed-Initiative Intent Recognition Using Cloud-Based Cognitive Services

Masterarbeit an der Universität Ulm

Vorgelegt von:
Matthias Kraus
matthias.kraus@uni-ulm.de

Gutachter:
Prof. Dr. Dr. Wolfgang Minker
Prof. Dr. Susanne Biundo-Stephan

Betreuer:
Dr. Florian Nothdurft

2016
Abstract

In spoken dialogue systems, understanding the intention of a user is essential for a successful and natural perceived human-machine interaction. Therefore, in the scope of this thesis, the creation and evaluation of a dialogue system is presented, which relies on the cloud-based Language Understanding Intelligent Service (LUIS) from Microsoft in order to recognize automatically the intention of a user from an utterance. For the evaluation of the cloud-based approach, the LUIS-based system is tested against a conservative hand-crafted grammar-based system by conducting an online study, with regard to the degree of the complexity of the user input. The main focus of the evaluation is put on system performance and user experience. For comparison of the different approaches, a mixed-initiative dialogue is designed for each system.
Contents

1 Introduction .............................................. 1
  1.1 Motivation ........................................... 1
  1.2 Task Description ..................................... 2
  1.3 Outline ............................................... 2

2 State of the Art .......................................... 3

3 Background and General Information .................. 5
  3.1 Spoken Dialogue Systems ............................. 5
    3.1.1 Automatic Speech Recognition ................... 6
    3.1.2 Language Understanding ........................... 7
    3.1.3 Dialogue Management ............................... 8
    3.1.4 Language Generation ............................... 10
    3.1.5 Text-To-Speech Synthesis ......................... 11
  3.2 Microsoft Cognitive Services ........................ 12
    3.2.1 Language Understanding Intelligent Service .... 12
    3.2.2 Microsoft Bot Framework .......................... 15
    3.2.3 Skype Calling API ................................ 16
  3.3 Workout Domain ....................................... 18

4 System Design and Implementation .................... 21
  4.1 Creation Of Cloud-Based Spoken Dialogue System .... 21
    4.1.1 Call Establishment and System Initialization .... 22
    4.1.2 Bot Speech Recognition ............................ 26
    4.1.3 Bot Language Understanding ....................... 27
    LUIS-Based System ...................................... 27
    Grammar-Based System ................................... 32
    4.1.4 Bot Dialogue Management ......................... 34
  4.2 Mixed-Initiative Dialogue Design .................... 39
    4.2.1 LUIS-Based Dialogue Design ....................... 39
    4.2.2 Grammar-Based Dialogue Design .................... 42

5 Evaluation and Results .................................. 45
  5.1 Evaluation ............................................ 45
  5.2 Results ............................................... 46
List of Figures

3.1 The general architecture of an SDS ........................................... 5
3.2 The architecture of the ASR module ........................................... 6
3.3 A lexical tree for the sentence “The Customer Accepts The Contract” .... 7
3.4 The four major paradigms in DM .............................................. 10
3.5 Overview over the LUIS framework GUI .................................... 13
3.6 The JSON object of the LUIS response ...................................... 14
3.7 Microsoft Bot Connector Interface ......................................... 15
3.8 The interplay between a user, the Skype bot platform and the bot ...... 17
4.1 SDS architecture ................................................................. 21
4.2 Skype configuration ............................................................. 22
4.3 Call establishment ............................................................... 24
4.4 UML for UserIntents, Entities and UserUtterance classes ............. 31
4.5 UML for State, History and DialogueProfile classes .................... 35
4.6 Dialogue Flow of the LUIS-Based System ............................... 40
4.7 Dialogue Flow of the Grammar-Based System ............................ 42
5.1 Diagram of the objective metrics for the overall evaluation ............ 47
5.2 Results of the DCR per individual dialogue ............................... 48
5.3 Results of the TSR per individual dialogue ................................ 49
5.4 Example dialogue, demonstrating common ASR mistakes ............. 54
List of Tables

5.1 Results of the evaluation of the LUIS-based and grammar-based bot . . . . . . . . . 47
5.2 Results of the individual DCR of each dialogue . . . . . . . . . . . . . . . . . . . . 48
5.3 Results of the individual TSR of each dialogue . . . . . . . . . . . . . . . . . . . . 49
5.4 The user-chosen complexity for the fourth, individual creatable dialogue . . . . . 50
5.5 The results of the final questionnaire, which covers the overall interaction experi-
ence with both systems . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 51
5.6 An overview of the significant results of the questionnaire for the individual dia-
logues with both systems . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 52
5.7 Medians for five representative statements of the evaluation, depending on the
used device type . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 55

A.1 The results of the questionnaire for the first dialogue . . . . . . . . . . . . . . . . 60
A.2 The results of the questionnaire for the second dialogue . . . . . . . . . . . . . . . 61
A.3 The results of the questionnaire for the third dialogue . . . . . . . . . . . . . . . . 62
A.4 The results of the questionnaire for the fourth dialogue . . . . . . . . . . . . . . . 63
List of Abbreviations

ADL Average Dialogue Length
AM Acoustic Model
API Application Programming Interface
ASR Automatic Speech Recognition
BN Bayesian Network
CDLDA Conceptual Dynamic Latent Dirichlet Allocation
CFG Context Free Grammar
CRF Conditional Random Field
DCR Dialogue Completion Rate
DM Dialogue Management
DTMF Dual-tone Multi-Frequency Signaling
GUID Globally Unique Identifier
HDC Hand-crafted Dialogue Management
HMM Hidden Markov Model
HTTP Hypertext Transfer Protocol
JSON JavaScript Object Notation
LG Language Generation
LM Language Model
LU Language Understanding
Luis Language Understanding Intelligent Service
MDP Markov Decision Process
NP Noun Phrase
PC Personal Computer
POMDP Partially Observable Markov Decision Process
<table>
<thead>
<tr>
<th>Short Form</th>
<th>Full Form</th>
</tr>
</thead>
<tbody>
<tr>
<td>REST</td>
<td>Representational State Transfer</td>
</tr>
<tr>
<td>RCNN</td>
<td>Recurrent Convolutional Neuronal Network</td>
</tr>
<tr>
<td>SDK</td>
<td>Software Development Kit</td>
</tr>
<tr>
<td>SDS</td>
<td>Spoken Dialogue System</td>
</tr>
<tr>
<td>SVM</td>
<td>Support Vector Machines</td>
</tr>
<tr>
<td>TSR</td>
<td>Task Success Rate</td>
</tr>
<tr>
<td>TTS</td>
<td>Text-To-Speech</td>
</tr>
<tr>
<td>UG</td>
<td>Unification Grammar</td>
</tr>
<tr>
<td>URL</td>
<td>Uniform Resource Locator</td>
</tr>
<tr>
<td>URI</td>
<td>Uniform Resource Identifier</td>
</tr>
<tr>
<td>UML</td>
<td>Unified Modeling Language</td>
</tr>
<tr>
<td>VP</td>
<td>Verb Phrase</td>
</tr>
<tr>
<td>WAV</td>
<td>RIFF WAVE</td>
</tr>
<tr>
<td>XML</td>
<td>Extensible Markup Language</td>
</tr>
</tbody>
</table>
1 Introduction

1.1 Motivation

In linguistic theory the terms syntax, semantics, and pragmatics play a huge role in understanding the process of language communication. Syntax defines rules or principles how words need to be structured in order to generate well-formed grammatical sentences or phrases (Kim & Sells, 2007). It determines the sequence in which lexical categories of words (subject, verb, preposition, . . . ) may appear in a sentence. For example, the sentence, “gain want stamina I to” is hard to understand because the English syntax is violated. In contrast, “I want to gain stamina” is correct syntax and hence better understandable. Semantics are encoded in the grammatical structure of an utterance and attach literal meaning. For instance, in “I want to gain stamina”, “I” represents the speaker of the utterance, “want to” implies that the speaker wishes for a particular thing, “gain” means to get something useful, in this case “stamina”, which stands for the ability to withstand fatigue. Pragmatics define what the speaker is really conveying with his utterance, hence what is the speaker’s intention in a certain context (Korta & Perry, 2015). Intentions are very important for successful communication, as speakers want to produce an intended effect on the listener by providing a certain utterance (Grice, 1957). In a dialogue, a user intends to receive an appropriate response by the addressee as effect. Consider the sentence “I want to gain stamina”, which is perceived differently under various circumstances. Uttering this sentence to a colleague while writing a master’s thesis, the addressee understands that the speaker wants to increase his mental stamina in order to be able to work for a longer period of time without pause. As an effect, the listener may provide useful tips for better concentration. By contrast, uttering this sentence in a fitness center to a workout trainer, the coach receives the message that the speaker wants to increase body stamina. As a result, the trainer may create an according fitness plan. Therefore, it is obvious that the main task of a listener is to understand the intention, which the speaker wants to convey in an utterance in order to take an appropriate action. In spoken dialogue systems (SDSs), this constitutes the task of the dialogue manager. However, current systems rely heavily on prescribed, inflexible grammars, which often lack to fully convey the intention of a user and may lead to unnatural dialogues. Furthermore, the creation of such grammars is often very time-consuming. Hence, statistical intention recognition of spoken utterances form a recent trend in human-machine interaction as promising approach in order to avoid these problems. In addition, these approaches may lead to a better performance of an SDS and a more natural perception by the users.
1 Introduction

1.2 Task Description

In this work, the creation of a complete SDS is presented, which relies on cloud-based Language Understanding Intelligent Service (LUIS) from Microsoft in order to recognize automatically the intention of a user from an utterance. Therefore, a statistical model has been set up and trained. This model is made accessible by the Microsoft Bot framework, which is used as template for the SDS. Within the bot, methods for receiving and transmitting spoken utterances via Skype and a dialogue manager module in order to process information are also implemented. The goal of this thesis is to show what impact statistical intention recognition approaches like LUIS have in human-machine interaction, regarding system performance and user experience. For this purpose, a baseline grammar-based bot system has been created, which is evaluated against a bot, relying on a LUIS model by conducting an online study. For comparison of the different approaches, a mixed-initiative dialogue is designed for each bot. The main focus of the evaluation is the processing of complex user inputs containing multiple intentions. In addition, it is examined how this effects the user’s experience and satisfaction with the system.

1.3 Outline

The outline of this work is as follows: in Chapter 2, an overview of the current state-of-the-art techniques of intention recognition in dialogue systems is provided. Background and necessary information for understanding the content of this thesis is provided in Chapter 3. There, the general components of an SDS are described, as well as insight is given on the cloud-based cognitive services used for the implementation of the dialogue system. Furthermore, the domain of the dialogue is presented. In Chapter 4, the implementation and the design of the experimental setup is described. This chapter provides detailed information about the creation of the cloud-based SDS, and discusses the design of the mixed-initiative dialogues. Chapter 5 deals with the evaluation of the two created systems and a discussion of the results is presented. Finally, in Chapter 6, the work is summarized in a conclusion and a forecast on future work is presented.
2 State of the Art

In the context of SDSs, the intention of a user is incorporated in speech acts or dialogue acts, which provide an abstract representation of an action a user wants to express in an utterance (Traum, 1999). For example, such actions may express the user’s intention to inform a dialogue partner about a specific topic or to request new information. In order to detect the intended actions in an utterance, there exist several suggestions in current research.

Klüwer et al. (2010) use syntactic and semantic relations as features for dialogue act recognition, whereby the system is trained on an annotated dataset corpus. As syntactic relation features, predicate argument structures of an utterance are used, while as semantic relation features, manual annotated semantic predicate classes and semantic roles, like “agent”, “patient” and “topic”, are used. The semantic information are carried out using VerbNet (Schuler, 2005) as source. Furthermore, context features like the preceding dialogue act and the equality of the topic of the last and current utterance are applied for recognition. The classification of the dialogue acts is performed using a Bayesian network (BN) classifier.

Similar to this approach, Král & Cerisara (2014) use features derived from deep sentence parse trees for dialogue act recognition. These features comprise syntactic features like dependencies between words in sentences, as well as so-called “baseline” features like part-of-speech tags. The dialogue act recognition is implemented on word-level basis, where a conditional random field (CRF) takes the features as input for tagging individual words of a sentence. Therefore, each word of a sentence is assigned a dialogue act. The authors claim that every single word contributes to the representative dialogue act of the whole sentence. The overall dialogue act of the sentence can be derived either using majority voting, or deploying a BN classifier in order to determine the best appropriate dialogue act.

Another novel approach to detect dialogue acts in utterances is the use of word embedding features and density-based clustering (Jang et al., 2016). Word embedding maps words to an n-dimensional number vector, which allows to identify words occurring in similar contexts. Density-based clustering is performed on the vectors to use word similarity values from the word embedding space as lexical features. These features are used in order to train a support vector machine (SVM) classifier. Using this classifier, dialogue acts can be predicted from a user utterance.

In Yeh et al. (2016), a conceptual dynamic latent Dirichlet allocation (CDLDA) model for topic detection in utterances is proposed. CDLDA uses speech acts and hypernym information of words as features to detect the relation between an utterance and a specific topic. A hypernym is a phrase or a word, representing a semantic field and consists of a number of hyponyms, which define instances of this semantic field. For example, “Color” is the hypernym for “Red” and
“Blue”. Contrary to the other presented state-of-the-art techniques, this approach does not use syntax relations in utterances to detect the intention of an utterance. Instead, it applies speech acts for recognizing hypernyms of words, as speech acts correlate with certain topics. Hence, the semantics addressed in an intention can be defined and extracted, which also plays a great role in this thesis.

Liu et al. (2015) provide an approach, which combines intention recognition and topic detection from spoken utterances. Here, a recurrent convolutional neuronal network (RCNN) is applied, which is basically based on a traditional convolutional neuronal network. The recurrence factor of the approach is achieved by additionally using conversation information from previous dialogue steps, i.e., the intention and topic of the last user utterance. The CNN derives feature vectors over a word sequence, represented as a concatenation of word vectors. These vectors include positional, as well as syntactic information of the word in the sequence. The features are derived by using a fixed-sized overlapping convolutional window, which glides over the observed word sequence. In order to determine the intention of an utterance, the sum of the feature vectors are used as input by a linear classifier. For topic detection, a CRF is used, which tags the sequence of interest with a specific topic label. By adding the intention features and topic labels from the previous interaction step, a performance gain of the suggested approach is reached according to the authors. In this work, a similar approach is used for the semantic parsing module in a SDS.

In current research, only the development of statistical intention recognition modules is covered, while mostly concentrating on the evaluation of the recognition accuracy of intentions in utterances against other design approaches. However, to my best knowledge, it is not examined how these approaches effect the user experience while interacting with an SDS and how intention recognition performs against conventional semantic grammars. Therefore, these topics are dealt with in the scope of this thesis. In the following chapter, background and general information of this thesis is presented by describing the components of an SDS and by providing an overview of the used cognitive services.
3 Background and General Information

In this chapter, some background for the understanding of this thesis is presented. First, SDSs in general are examined, what they are and how they function. Afterwards, the cloud services provided by Microsoft are described. These are the LUIS framework for intelligent language understanding, the Microsoft Bot Framework, which is used as shell for the input/output capabilities of the embedded dialogue manager, and the Skype Calling API for enabling speech input and voice call handling. The last section of this chapter deals with the characteristics of the workout domain, which is used for the dialogue design.

3.1 Spoken Dialogue Systems

In this section, the general architecture of an SDS will be described along with its main components. The purpose of such a system is to interact with machines or software systems through natural speech. This is conducted in a turn taking manner. A turn is a period of time, in which one participant in the dialogue is speaking, whereas the other is listening. After the speech receiving dialogue partner has processed the input of his counterpart and has made a decision, the roles are switched. This dialogue cycle is represented in the system's architecture. As depicted in figure 3.1, the architecture resembles a pipeline like structure, as the components are ordered blockwise, while the output of each module serves as input for the next following one.

![Figure 3.1: The pipeline-like architecture of an SDS consisting of automatic speech recognition, semantic analysis or language understanding, a dialogue manager, a language generation module, and text-to-speech synthesis.](image)
Subsequent modules form an SDS: Automatic Speech Recognition (ASR), Language Understanding (LU), Dialogue Management (DM), Language Generation (LG), and Text-To-Speech Synthesis (TTS).

3.1.1 Automatic Speech Recognition

![Figure 3.2: The architecture of the ASR module, consisting of signal processing and feature extraction, the language and acoustic model, as well as the hypothesis search. (Yu & Deng, 2012)](image)

The task of the first module in the pipeline is to capture the user’s speech input and to transcribe it into a word sequence. As illustrated in figure 3.2, the common architecture of an automatic speech recognition system comprises four subcomponents (Yu & Deng, 2012).

First, the audio signal is preprocessed in order to enhance the signal by removing noise and channel distortions. After the audio signal is transformed from time-domain to frequency domain, relevant feature vectors for the acoustic models (AMs) are extracted. State-of-the-art ASR systems usually use the mel-frequency cepstral coefficient or relative spectral transform-perception linear prediction as feature vectors. The AM contains information about the acoustic and phonetic characteristics and uses these feature vectors to estimate the AM score $P(O|W)$, i.e., the likelihood of a set of acoustic observations $O$, given a certain word sequence $W$. In the past, this score was often estimated using Gaussian mixture hidden Markov models (Juang & Rabiner, 1991). More recently, discriminative hierarchical models like deep neural networks or convolutional neural networks are applied (Yu & Deng, 2012).

The language model (LM) estimates the probability of the occurrence of a certain sequence of words $P(W)$ by learning statistical dependencies between words, e.g., using n-gram models while relying typically on a text training corpora. This estimate or LM score can be enhanced by using prior knowledge of the domain.
Both scores, LM and AM, are used to determine the best hypothesis for the word sequence by maximizing the following equation, deploying the Bayesian approach by Yu & Deng (2012):

$$W^* = \arg\max_{W} P(W|O) = \arg\max_{W} \frac{P(O|W)P(W)}{P(O)}$$  \hspace{1cm} (3.1)

The meaning of the most probable word sequence is then determined by the language understanding module, which is described in the following section.

### 3.1.2 Language Understanding

According to De Mori et al. (2008) spoken LU "...is the interpretation of signs conveyed by a speech signal", in order to obtain "...a conceptual representation of natural language sentences". In the past, this was accomplished by using handcrafted syntactic/semantic grammars, e.g., context-free grammars (CFGs) or unification grammars (UGs), while not taking into account contextual information.

![Figure 3.3: A lexical tree, examining the sentence “The Customer Accepts The Contract”. “The Customer” consists of a determiner an a noun and constitutes a noun phrase, representing an “Agent”. “Accept” is the verb of the sentence and as consequence the “Action”. “The Contract” forms also a noun phrase, but is this time labeled as “Theme”. (De Mori et al., 2008)](image)

Figure 3.3: A lexical tree, examining the sentence “The Customer Accepts The Contract”. “The Customer” consists of a determiner an a noun and constitutes a noun phrase, representing an “Agent”. “Accept” is the verb of the sentence and as consequence the “Action”. “The Contract” forms also a noun phrase, but is this time labeled as “Theme”. (De Mori et al., 2008)

Furthermore, a parser analyzed the input word sequence with the grammar. An example for such a syntactic and semantic analysis is depicted in 3.3. Here, a syntactic tree is generated to determine the composition of the sentence. Therefore, the sentence is divided syntactically into subphrases like noun phrase (NP) or verb phrase (VP), according to a specific grammar. Afterwards, the components of the parse tree are semantically labeled (De Mori et al., 2008).

Nowadays, mostly statistical methods are applied, using deep semantic processing, because the process of manual grammar authoring is expensive, time consuming, and is dependent on the expertise of the developer. As described in the state-of-the-art section of this thesis, deep semantic processing requires to find the most probable user intention in an utterance using reasoning about the current task and context.
3 Background and General Information

Therefore, LU is described as pattern recognition problem in an analogous way as in ASR (Wang et al., 2005):

\[
M^* = \arg\max_M P(M|W) = \arg\max_M P(W|M)P(M)
\]  

(3.2)

Here, the goal is to find the semantic representation of the meaning \( M \) with the maximum a posteriori probability \( P(M|W) \), given a certain word sequence \( W \). \( P(M) \) defines the semantic prior model and assigns a probability to the meaning \( M \). \( P(W|M) \), as the lexicalization model, assigns a probability to a sequence of words \( W \), depending on the present meaning. The Hidden Understanding Models are an implementation of these models, using Hidden Markov chains (Miller et al., 1994). These chains consist of slots as states, while the transition probability and the Hidden Markov Model (HMM) topology form the prior probability \( P(M) \), and the emission distributions define the lexicalization model \( P(W|M) \).

As described in Chapter 2, the semantic representation of these intentions is mostly structured in semantic frames also called dialogue acts, where each frame can have several slots (Wang et al., 2005). For example, the frame name “InformTrainRide” could have the slots “DepartureCity” and “ArrivalCity”, where both slots are of the filler type “City”. For example, a user act, “I want to go from X to Y by train”, would address the frame name “InformTrainRide” and fill the slots “DepartureCity” with “X” and “ArrivalCity” with “Y”. The main topic of this thesis is to compare both variants, hand-crafted grammars and statistical intent recognition.

The next step in the dialogue system’s pipeline is to choose the best system action, depending on the intention and the meaning of the last user action.

3.1.3 Dialogue Management

The dialogue management component forms the decision making module of the architecture. After the LU module has decoded the semantic content of the user utterance, the dialogue manager chooses an appropriate system action out of a set of system actions \( a \in A \) as response to the user action \( u \in U \).

In doing so, the manager takes into account the last observed user act and the current dialogue state or belief state \( s \in S \), which contains the whole dialogue history, i.e., an internal representation of the past exchanges between a user and the system. After each exchange, the belief state is updated via a transition function \( T \) which maps \( T : S \times A \times U \rightarrow S \) (Thomson, 2013). Furthermore, a dialogue manager possesses a dialogue policy \( \pi \) that controls its behaviour. It is responsible for determining the next system action. Therefore, \( \pi \) maps the states to a set of Actions, \( \pi : S \rightarrow \Pi(A) \) (Thomson, 2013). In statistical systems, \( \Pi(A) \) denotes a probability distribution over all possible system actions. In addition, the management module uses strategies in order to define the behaviour of a dialogue system. The arguably most important strategies in the field of dialogue systems constitute those, regarding the initiative and confirmation. The chosen initiative structures the flow of the dialogue. According to Litman & Pan (2002), there exist three initiative strategies: the system-initiative strategy, the user-initiative strategy, and the mixed-initiative strategy.
With applying the system-initiative strategy, the system is in full control of the dialogue. Therefore, it is the most conservative approach of the three strategies and forms the most inflexible, while typically increasing the dialogue length (Litman & Pan, 2002). An advantage of this approach is that the user responses are shorter and less out of the system’s speech grammar, as the system asks more precise questions, regarding usually only one requested piece of information, e.g., “What is your fitness level?”.

By contrast, the user-initiative strategy is the freest approach and occurs most often in conversational dialogue systems. The strategy leaves it up to the user how many pieces and which information to provide. Therefore, often the user opens the dialogue in user-initiative systems, while the system tries to respond to the utterance. This may be very problematic, as the system’s LU must be broad enough in order to understand what the user has said, as well as the ASR has to be very good, too. Hence, this strategy is only effective if the user knows exactly how to use the system.

A combined approach of these strategies is the mixed-initiative strategy. This strategy is used for the systems in this thesis and therefore is the most relevant. Here, the user may start the dialogue or respond to open-ended questions like, “How may I help you?”. However, after the first input the system switches to more precise questions in order to gain the missing information for completing the dialogue, but unlike the system-initiative strategy, it allows for multiple pieces of information per question, and also lets the user ignore the specific question, e.g. the user may give input about a piece of information, which is not directly asked for. For example, as a response to the system’s question, “On which dates do you want to train?”, the user may say, “On three days per week and for one hour per day.”. As a result, this approach leads to an easier information flow, which may be more efficient.

In order to identify and recover from misrecognitions by the ASR and LU module, confirmation strategies are indispensable. This confirmation of information can happen in two ways: Either the system asks explicitly or implicitly for confirmation. A system using an explicit-confirmation strategy directly asks for validation of a user input. For example, the system has understood that the user said he would be a beginner in fitness. Then, it would try to ground this information, i.e., let the user know what it understood, by asking, “You are a beginner in fitness. Is that correct?”

Contrary to this approach, the implicit-confirmation strategy would result in the following question, “As a beginner, how often do you want to train per week?”. Hence, the system simultaneously asks for confirmation indirectly and directly for new information. The frequency of confirmations can also be adjusted. The affirmation can either take place at the end of the dialogue to confirm all necessary information at once, or after each exchange, which contains a new piece of information. In this thesis, an explicit-confirmation strategy is used after each time the user gives new input, although only if the confidence score of said input is lower than 0.85. Confidence means the degree of how sure the system is about the recognized user utterance.

According to Thomson (2013), there exist four major paradigms in DM: Hand-crafted approaches (HDCs), Markov Decision Process models (MDPs), Bayesian Network approaches (BNs), and Partially Observable MDP approaches (POMDPs).
Hand-crafted dialogue modelling is at the most basic level, as the concepts of dialogue state, transitions of state, and policy are all directly defined by the system developer. For more complex dialogues, i.e., dialogues, which have a large number of states and therefore an increased number of state transitions and policy decisions as well, statistical approaches, which deploy uncertainty models are chosen. In figure 3.4, the differences between each technique is depicted.

A hand-crafted dialogue design process by using a frame-based dialogue manager is chosen in this work (Goddeau et al., 1996). A frame-based management module allows the user to give information about certain concepts called slots, which have already been described in the LU section. The current number of filled slots are part of the dialogue state. Depending on which slots are already filled, a pre-specified system action is executed. This rather basic approach is used in this work's context, as the whole dialogue system design has been developed in a prototyping kind of style.

For the other paradigms, there exist numerous examples. For instance, in Levin et al. (2000) a MDP based approach is described which relies on a bigram model being trained with a user simulator. Lison & Kennington (2015) use a BN design, whereby the dialogue domains are modelled in XML and use probabilistic rules for state transitions. A POMDP dialogue manager is presented in Young et al. (2010). Here, the Hidden Information state model is applied in a tourist information domain.

In the next sections, it is described how the semantic representation of the system action is transformed into natural speech output and presented to the user.

### 3.1.4 Language Generation

The first part of the transmission of information to the user is language generation. The task of this component is to transcribe the semantic representation, which is provided by the dialogue manager, into text. The most common approach is to use templates. In template-based systems, the non-linguistic semantic representation is directly put into a linguistic pre-specified template structure (Reiter & Dale, 1997).
3.1 Spoken Dialogue Systems

For example, the system dialogue act, “Inform(DepartureCity=X,ArrivalCity=Y)” would be transformed into following pre-defined template sentence, “You want to go from “X” to “Y”. “. The template-based approach is effective, if the number of system actions is manageable. As this is the case in this work, this approach has been implemented in the here used system.

In order to make language generation more variable, other, more sophisticated techniques have been developed. Oh & Rudnicky (2000) presents a stochastic, corpus-based approach. In a first step, a content planner is built, using a bigram model. Afterwards, an n-gram language model is used, in order to stochastically generate utterances.

Mairesse & Walker (2007) have developed a system, which also uses stochastic models to generate language. In addition, this framework manifests personality, notably extraversion, while using a highly parametersizable generator.

Finally, after having a linguistic representation of the system action, a text-to-speech synthesis is performed.

3.1.5 Text-To-Speech Synthesis

There exist multiple variants of synthesizing speech in SDSs. In the past, predominantly unit selection techniques were applied. This approach retrieves linguistic units from a large speech database in order to convert a text into a sequence of sounds (Taylor, 2009). The units are selected according to how well they match a specification of a target utterance structure and how well they join together. The specification, as well as the units, can be any mixture of acoustic and linguistic features, e.g, phonemes or diphones with pronunciation features. In order to find how well a specification matches a unit, a target cost function is applied, which measures the difference between these two feature structures. A join cost function determines how well two consecutive units fit together. The best sequence of units can be found by minimizing the cost functions, using the viterbi algorithm (Forney, 1973). Finally, the speech waveform is then synthesized from the concatenated units’ spectral and excitation parameters by using a speech synthesis filter. A realization of this approach is implemented in Clark et al. (2004).

Another, more modern technique is the use of HMM-based speech synthesis systems started by Zen et al. (2007). Here, instead of units, phonetic and linguistic context-dependent HMMs are trained from databases of natural speech similar to ASR systems. Instead of HMMs, also deep neural networks can be used to synthesize speech (Ze et al., 2013).
3 Background and General Information

3.2 Microsoft Cognitive Services

In contrast to the DM and LG module, which are implemented from scratch in this work, cloud-based services are used to model the other parts of an SDS: The Bing Speech Recognition API is used to provide ASR, the TTS function of the Skype Calling API is used for speech synthesis, and the LUIS API for LU. All of the components of the dialogue system are embedded in the Microsoft Bot Framework. As LU and DM form the focus point of this thesis, the speech recognition and TTS cloud-services are considered as given and are not further examined in this thesis. Consequently, the LUIS API is examined in detail in the following.

3.2.1 Language Understanding Intelligent Service

LUIS is a cloud-based visual beta-tool for the creation of active-learning LU models, without necessarily requiring experience in machine-learning (Williams et al., 2015). The development of a model starts with creating a LUIS application, which is based on the concepts of intents and entities:

**Intent** describes the intention of a user in an utterance (Williams et al., 2015). In this context, intention means a desired action, e.g., “AdressWorkoutType” is the intention triggered by the utterance, “I want to do upperbody workout”. The designer defines these intents along with an example trigger for each concept, relevant to a domain-specific topic (LUIS, 2016). For this work, the intents are designed for a workout-domain, which will be described in section 3.3. For each application there also exists a “None” intent, which defines an out-of-domain utterance.

**Entity** defines a word or a subset of words, which resemble “slots” in a slot-value semantic representation (Williams et al., 2015). The designer specifies an entity, e.g., “WorkoutType”, according to the domain and can label particular words as corresponding “values”. For example, in the sentence, “I want to do upperbody workout”, “upperbody workout” can be labeled as “WorkoutType”. Furthermore, it is possible to tag more than one entity in an utterance. The framework also allows the use of pre-built entities like numbers, temperatures, etc., but such entities are not necessary in the context of this thesis and thus are ignored. The entities, used for this work are consequently also modelled according to a workout-domain.

After the model has been initialized, it can be trained with each new entered utterance (LUIS, 2016). For each new input, LUIS tries to automatically determine the correct intent and entities in an utterance. This is an advantage in such a way that previously unseen data can nonetheless be recognized correctly. Moreover, LUIS gives information about how sure it is that the detected intent is correct and also suggests labels. An example of the active learning approach, in form of manual labeling is depicted in figure 3.5. As every input is logged by the system, all previous utterances can be used for learning.
3.2 Microsoft Cognitive Services

Figure 3.5: Overview over the LUIS framework GUI. On the left side, the existing intents and entities are listed. Furthermore, the developer is able to enter new utterances to be recognized, or to relabel recognized intents and entities with the GUI-elements, located at the center of the page.

If the intent of an utterance is misclassified, the designer can relabel the input by choosing the right intention from a drop-down menu and click on the “submit” button. The menu contains additionally confidence scores from 0.00 to 1.00 for each intent, according to how well the respective intent fits to the input. Furthermore, the entities are highlighted in their corresponding colour and can be relabeled, or other parts of the sentence can be chosen as entities. Hence, the visualization serves as indicator of the performance of the model, which is useful for the designer in order to track the capabilities of the system (Williams et al., 2015).

The training of LUIS is conducted by generalizing from all labeled examples. Internally, logistic regression is used for classifying intents and conditional random fields are used in order to determine entities (LUIS, 2016). Unfortunately, the specific methods and algorithms are not made public by Microsoft. For the sake of completion, the machine-learning techniques are described shortly in the following. Logistic regression is defined mathematically according to Press & Wilson (1978) as conditional probability,

$$P(E | \vec{x}) = \frac{1}{1 + e^{-\alpha - \beta \vec{x}}}$$

Transferred to the LUIS system, $E$ describes thus the event that a certain intent belongs to a specific object, in this context to an utterance. $\vec{x}$ denotes a profile vector containing the attributes of the object. $\alpha$ and $\beta$ are unknown parameters which can be estimated by data. The intent with the highest conditional probability is then associated to the utterance.
3 Background and General Information

Figure 3.6: The JSON object containing an n-best list for the LUIS intents and entities recognized from text.

CRFs are undirected graphical models, which are used to calculate the conditional probability of values of a set of output nodes, given the values of a set of input nodes (Lafferty et al., 2001). McCallum & Li (2003) use such CRFs for entity recognition in news articles, which is quite similar to the use of entities in LUIS. According to the authors, CRFs define the conditional probability of a state sequence $\vec{s}$ of dimension $T$, where each state is associated with a label, e.g., "WorkoutType", given an observed input data $\vec{o}$ of dimension $n$, which can be a sequence of words in an utterance. Mathematically this is described as,

$$P(\vec{s} | \vec{o}) = \frac{1}{Z_{\vec{o}}} \exp \left( \sum_{t=1}^{T} \sum_{k} \lambda_k f(s_{t-1}^{-1}, s_t, \vec{o}, t) \right)$$

(3.4)

Here, $Z_{\vec{o}}$ is a normalization function over all states and $\lambda_k$ is a learned weight for each arbitrary feature function $f(s_{t-1}^{-1}, s_t, \vec{o}, t)$. This feature function is usually defined with a value of 0 in most cases, but, for example, is defined 1 if a certain state sequence $s_{t-1}^{-1}, s_t$ matches a specific observation $\vec{o}$ at position $t$ (McCallum & Li, 2003). The weights are trained on data to maximize the conditional probability of the right entity-to-words association.

In order to use a LUIS model in an application, the designer is able to "publish" the model to an HTTP endpoint, which is accessible by any internet-connected device. The interaction with LUIS is on textual basis.

```json
{
    "query": "I want to lose weight and train three days a week",
    "intents": [
        {
            "intent": "WeightLimitations",
            "score": 0.803986822
        },
        {
            "intent": "AchieveAbstractEffect",
            "score": 0.64968298
        },
        {
            "intent": "AddressWorkoutType",
            "score": 0.00455573267
        },
        {
            "intent": "None",
            "score": 0.00560159737
        },
        {
            "intent": "WellDefinedSpecificMuscle",
            "score": 0.000610023
        },
        {
            "intent": "AdjustPreferences",
            "score": 0.00016570039
        },
        {
            "intent": "Confirmation",
            "score": 0.000643008-06
        },
        {
            "intent": "AddressSpecificPain",
            "score": 0.0049832-13
        }
    ],
    "entities": [
        {
            "entity": "lose weight",
            "type": "AbstractGoal",
            "startIndex": 10,
            "endIndex": 25,
            "score": 0.803986822
        },
        {
            "entity": "three days a week",
            "type": "TrainingDays",
            "startIndex": 35,
            "endIndex": 45,
            "score": 0.00560159737
        }
    ]
}
```
Hence, when a text message arrives at the endpoint, the message is processed and LUIS returns an object in JavaScript Object Notation (JSON) as response. In figure 3.6, the structure of the response is depicted. For the user utterance, “I want to lose weight and train three days a week”, the system labels the intent of the utterance as “SetLimitations” and tags “lose weight” and “three days a week” as entities of the types “AbstractGoal”, respective “TrainingDates”. In this work, the trained LUIS model of the workout domain is accessed by the *Microsoft Bot* framework, which is be described in the following section.

### 3.2.2 Microsoft Bot Framework

Microsoft’s *bot framework* enables to build and manage intelligent artificial agents, also called bots, which are able to interact naturally with users across different communication channels like Skype, Slack, Facebook messenger, and other services (Microsoft, 2016a). These channels can be configured and accessed by registering the bots to the "Bot Framework Developer Portal". In order to host a bot, a subscription to Microsoft’s *Azure cloud service* is mandatory (Microsoft, 2016b). The framework was introduced and made publicly accessible on March 30th, 2016 at Microsoft’s "Build developer conference" (Microsoft, 2016c).

The starting point of each bot is the "Bot Builder SDK". It contains tools for developing reusable dialogs of different types. These types of dialogues range from simple built-in prompts and command dialogues to more complex dialogues, using LUIS for LU, which was employed in this work. The main component of this SDK is the "Bot Connector", which connects the bot to the available channels, routes the messages from bot-to-user and user-to-bot and manages the state of the bot. Furthermore, the connector makes the bot scalable to different channels, without adjusting the application.

![Figure 3.7: The Microsoft Bot Connector and its description as interface between a bot and several channels.](Electric-News, 2016) (Used with permission from Microsoft.)

The role of the “Bot connector” as interface is depicted in figure 3.7. As the whole architecture is REST-based and hence uses webhooks for exchanging messages, there is no constraint on the programming language in the developement process.
Therefore, in this work, the bot is implemented, using the bot framework client library as .NET application in C#.

Regarding that the bot framework is primarily based on only text messages, the framework needs to be extended with the Skype Calling API in order to process voice calls. This API is described in the following section.

### 3.2.3 Skype Calling API

Another feature of the bot framework is the possibility to enhance the bot with the ability to receive and process voice calls by including the Skype Calling API (Skype, 2016b). Similarly to the messages webhook, which handles the purely text-based information flow to a bot and vice versa, the call-bot API uses the so-called calling webhook in order to notify the bot that it has received a voice call. As response to the call, the bot can execute actions according to a pre-defined workflow. A workflow consists of a set of basic actions for call handling: **Answer**, **Play Prompt**, **Record Audio**, **Speech-to-text**, **DTMF tones**, and **Hang up**. The speech-to-text component of the calling API and the DTMF tones are not used in this work, because they are either inflexible and not appropriate, regarding the task (Speech-to-text), or unnecessary for fulfilling it (DTMF tones). In the following, the functions of the actions are described:

**Answer** is the first step of the first workflow of a call, after the bot has been notified by the Skype bot platform. The answer function is used for accepting a Skype call. There also exists the possibility to reject a call instead of answering it. This could be useful, for example, when a blocked user calls. As this is not necessary in the context of this thesis, this feature is not implemented in the used bot.

**Play Prompt** is used to play back media files to the user or to provide TTS functionality, which is crucial in dialogue systems for outputting information to the user and hence is implemented in the bot designed for this thesis. The prompt function takes a string of words as input, converts it to natural speech, and plays it back to the user. In addition, the gender of the prompt voice, special pronunciation and emphasis options can be adjusted, but in the context of this work, the default mode for prompts is used, i.e., a male voice for prompts, and none specific pronunciation and emphasis options are chosen.

**Record audio** is the possibility to record speech input by the user and store it in WAV files. Originally, the saved sound files are intended for relistening purposes. However, in the development process of the here described bot, the sound files are used as input for the Bing speech recognizer. Furthermore, this feature contains lots of optional functions, for example, the maximum recording duration and the maximum allowed initial silence before the recording will be stopped can be set.

**Hang up** is the last action of a workflow. It allows the bot to terminate the ongoing call and is used in this work at the end of each dialogue with a user.
In order to better understand the interaction between a user and a Skype bot, a sample call flow protocol, as depicted in figure 3.8, is described. The flow starts with the user calling a bot via the Skype application. Afterwards, the Skype bot platform routes the incoming call and notifies the bot that a call has been placed by sending a Conversation request JSON object. This request contains information, regarding the user, the target of the call, and the initial state of the call. The bot takes this information and decides, whether to answer or to reject the call. In this representative case, the bot answers the call and sends a workflow containing the action “Answer”. This workflow establishes the call and informs the bot that the “Answer” action has succeeded. The result of a successful workflow action is sent to the bot in the form of a ConversationResult JSON object. The bot responds to this message by forwarding a new workflow with the command to record the next user input. After the the user's speech input was recorded, a ConversationResult JSON object is again sent to the bot and the record file is uploaded to a cloud, here called “Blob Store”. In addition, the cloud store lets the bot access the appropriate HTTP link. The next transmitted workflow contains the actions “Play Prompt” with the file URI of the recording. As a result, the Skype bot platform executes the actions and interacts with the user by playing the content from the retrieved recording file. After that, the bot receives again a result message and sends as last interaction a workflow with the action “Hang up”, which ends the current call.
3 Background and General Information

Before, in the next chapter, the whole process of the development of the dialogue system and the dialogue design is described in detail, a brief introduction of the domain, modelled for this work is provided.

3.3 Workout Domain

All of the system-user dialogues evaluated for this thesis take place in the workout-domain, where a user wants to define a specific workout goal with the help of an SDS. The workout-domain is chosen as use-case because this work is done in addition to the work by Behnke et al. (2015), where also the fitness-training scenario is used as potential use-case. In their work, a user wants to create an individual training by interacting with a planning system. The overall-goal defining dialogue modelled for this thesis can be viewed as entry step for the planner system. An example dialogue could look as follows:

```
system: Hello, I'm the Workout Bot! How may I help you?
user: I want to gain more stamina and train two days a week.
system: How long do you want to train per day?
user: Twenty minutes per day.
system: What is your fitness level?
user: I'm a beginner and want to do upperbody workout.

system: This is your goal: AbstractGoal = Gain more stamina, TrainingDates = Two days a week, DurationPerDay = Twenty minutes, FitnessLevel = Beginner.
```

As indicated in the dialogue, the main concepts or semantic slots of are the following:

**AbstractGoal** defines the overall goal (effect) a user wants to achieve. For example, “Gain more stamina” would be a value for this slot. Besides, instead of defining an abstract goal, the user may alternatively address a specific pain or injury, e.g, “upper back pain”.

**TrainingDates** comprises the dates per week a user wants to train on, e.g, “Two days a week”. The user may also give information about specific days of the week by saying, for example, “I want to train on Mondays and Fridays”.

**DurationPerDay** is the time the user wants to spend on training per day. In the example dialogue, the user mentions exemplary that he wants to train twenty minutes per day.

**FitnessLevel** defines the degree of experience a user has already obtained in workout training. A corresponding value for this slot could be, “Beginner”, as seen in the example dialogue.
These are the mandatory slots for the designed dialogue. Furthermore, there exist optional slots, which are not necessary to complete a dialogue:

DefineMuscle is the slot the user addresses, if he wants to train a specific muscle or muscle group, e.g. the training of the biceps.

WorkoutType comprises all the different types of workout, which are necessary in the planning of a particular training session, but not in the definition of the overall goal. An example for a value for this slot could be “Upperbody workout”.

Preferences forms the possibility for the user to address personal preferences, e.g. special training equipment that he wants to use for workout.

The dialogue domain is modelled and implemented, according to these semantics. How the domain is used to build the SDS and to design the dialogues is described in the next chapter.
3 Background and General Information
4 System Design and Implementation

The creation of two dialogue systems and their components is described in this chapter. One of the system uses conventional grammar in order to understand user input, while the other system relies on statistical intention recognition by observing user utterances. Besides, the development of the mixed-initiative dialogue designs is examined.

4.1 Creation Of Cloud-Based Spoken Dialogue System

Figure 4.1: A general overview of the SDS’s architecture, consisting of the Skype API, embedded in the Microsoft Bot Framework.

Figure 4.1 shows the general architecture of the developed SDSs for this thesis. Every time the user provides speech input, the HTTPS endpoint of the bot receives the utterance and the Skype Calling API, which is embedded in the Microsoft Bot framework, records the call. After that, the function for speech recognition is called, which sends the WAV file, containing the speech signal to the Bing Speech Recognition cloud-service. There, the voice call is converted to text. Thereupon, it is dependent on the system how the recognized text is processed. The SDS, using grammar-based LU tries to match the text with a pre-defined grammar in order to understand what was said.
By contrast, the LUIS-based system uses a function to send the recognized text to the LUIS cloud-service, which delivers the estimated intents and entities of the utterance as response. As the grammar semantics, as well as the intents and entities received from LUIS are unstructured string-object types, it is necessary to prepare them, before they are forwarded to the dialogue manager. The DM module then chooses the next system action and updates the state, according to the user input. Finally, the system action is prompted to the user, using the TTS function of the Skype Calling API. Before all these steps of the architecture of the system are described in detail, a closer look is taken on how a call is established by the bot system.

### 4.1.1 Call Establishment and System Initialization

In order to access the dialogue system from everywhere using Skype, the implemented bot has to be registered with Microsoft’s own Azure service, as mentioned in the previous chapter in section 3.2. After registration, the bot can be enabled for calling by configuring the Skype channel in the Bot Framework developer portal. However, the Skype calls are only available in a preview mode, which limits the full processing capability.

![Figure 4.2: The GUI to configure the Skype calling settings for the bot application, e.g. enable the bot for Skype calls, and allow text or 1:1 calls to a bot. (Used with permission from Microsoft.)](image-url)
The bots developed in this work are configured to only allow speech input and no text. Therefore, as depicted in figure 4.2, text as message type is disabled and the calling webhook for speech input is enabled. When a user calls the bot, all content is sent to this URL. The routing of a call is handled by the following controller, which inherits from the Bot framework API's ApiController class. The structure of this class is prescribed by the API for calling and describes the interaction between the Skype bot platform and the bot itself:

```csharp
[BotAuthentication]
[RoutePrefix("api/calling")]
public class CallingController : ApiController
{
    public CallingController() : base()
    {
        CallingConversation.RegisterCallingBot(c => new Bot(c));
    }

    [Route("callback")]
    public async Task<HttpResponseMessage> ProcessCallingEventAsync()
    {
        return await CallingConversation.SendAsync(Request,
            CallRequestType.CallingEvent);
    }

    [Route("call")]
    public async Task<HttpResponseMessage> ProcessIncomingCallAsync()
    {
        return await CallingConversation.SendAsync(Request,
            CallRequestType.IncomingCall);
    }
}
```

The class is decorated with BotAuthentication (line 1) in order to validate the bot credentials over HTTPS and with RoutePrefix("api/calling") (line 2) to indicate that this class is addressed, when a call is placed. In the constructor in line 7, the factory method for creating the bot instance is registered for the Skype bot platform, i.e., the actions of this specific bot class should be used upon a user's call. The name of the intent recognition bot is “IntentBot”, while the name of the grammar bot is “GrammarBot”. For simplicity, the name “Bot” is used for the description of the architecture. Furthermore, CallingConversation is the top level composition class for calling, which contains the functions that are important to establish a call.

There exist two methods for the Skype bot platform to notify the bot: ProcessIncomingCallAsync (line 15-20) and ProcessCallingEventAsync (line 9-14). The first one is used to define that a call has been initiated by the user and establishes the connection between bot and user. The second method informs the bot that a workflow has been performed and asks for new orders. In figure 4.3, the reaction of the system upon a Skype call is depicted.
Figure 4.3: An overview of the establishment of an user-call. After the user initiated the call, the system provides methods in order to establish a connection for the transmission of information.

After the `ProcessIncomingCallAsync` method has set up the connection and an instance of “Bot” has been created, the bot decides which action to take next. For this reason, it holds a `OnIncomingCallReceived` event function besides a set of other actions for different events:

```csharp
private Task OnIncomingCallReceived(IncomingCallEvent incomingCallEvent)
{
    var id = Guid.NewGuid().ToString();
    incomingCallEvent.ResultingWorkflow.Actions = new List<ActionBase>
    {
        new Answer { OperationId = id },
        GetPromptForText("Welcome to the Workout Bot, How can I help you?")
    }

    string callerID = incomingCallEvent.IncomingCall.Id;
    _callStateMap[callerID] = new History();

    return Task.FromResult(true);
}
```
As a result to an incoming call, a new History object (line 12), which stores state information of the user, and a workflow (line 5-10) containing the next system actions is created. The History is mapped to the ID of the call (line 11) in order to allow for persistent state over the course of the dialogue for each user. In addition, this enables the bot to handle dialogues with more than one user at the same time. As described in section 3.2 of the previous chapter, the system needs to create a new Answer object (line 7) in order to accept the call. A Globally Unique Identifier (GUID) (line 4) is necessary to correlate an outcome of an action to itself. After that, the dialogue is started with the system prompting the welcome message to the user (line 8). The GetPromptForText method constitutes the TTS component of the dialogue system, which takes a string-object as input, automatically converts it into speech and prompts it to the user. Following the prompt, the OnPlayPromptCompleted function is activated, which starts a recording or hangs up the call, in case that the final state of the dialogue cycle has been reached. The decision to record the user utterance is made because Skype does not allow to directly gather the speech signal and the speech recognition module itself takes sound files as input. In the configuration settings for recording, the maximum duration time of one recording was set to ten seconds, the recording format was set to WAV, and the recording is stopped after one second of silence if speech has been detected. This is the lowest duration possible, for keeping the processing time of the system as short as it could be. Furthermore, no beep sound is played before the recording and if the user does not respond after three seconds, the record action is aborted and the system plays a reprompt of the last system action. The result of the recording is then used as input to speech recognition module, which is described in the following section.
4 System Design and Implementation

4.1.2 Bot Speech Recognition

Subsequently the recording action, the OnRecordCompleted function is called. This function is the main method in the dialogue process, as it comprises actions for ASR, LU, and DM. In the following source code, the section of the function which handles the speech recognition is presented:

```csharp
if (recordOutcomeEvent.RecordOutcome.Outcome == Outcome.Success)
{
    var record = await recordOutcomeEvent.RecordedContent;
    using (var writer = new FileStream(path, FileMode.Create))
    {
        await record.CopyToAsync(writer);
    }
    Attachment att = new Attachment()
    {
        ContentUrl = "file:///" + path,
        ContentType = "audio/wav",
    }
    s = DoSpeechReco(att);
}
```

As the ASR function DoSpeechReco (line 17) needs an “Attachment”-object as argument, some preprocessing, regarding the recorded content has to be performed. Objects of the data structure “Attachment” (line 11) are used in the bot framework to send media data, e.g, video, pictures, or audio content. For this reason, the object contains a link to the content (line 13) and a definition of the type of content (line 14). In order to use the link, the bot application needs to have access to the recorded content stream (line 3) Therefore, the non-accessible virtual path to the recording on the application server is converted to a accessible physical path (4-6). After that, a file stream is established to the physical path in order to load the content, and the recording is attached to it (line 7-10).

In the next process step, the DoSpeechReco function loads the content from the file and forwards it to the HTTPS-endpoint of the Bing speech recognition platform. There, the recording is processed and the result of the recognition is sent back to the bot as a string-object. The code for the transmission of the data objects to the Bing server and for receiving the result are implemented according to the suggestion in Bing (2016). Hence, the details of the connection implementation are not further observed, but the solution is similar to the connection implementation of LUIS, which is described in the next section. In order to use the Bing speech services, a subscription is necessary.
4.1 Creation Of Cloud-Based Spoken Dialogue System

For this work, a free subscription option is chosen which allows to call the cloud-service 5000 times a month. In the next section, it is described how the semantic representation of the utterance is obtained.

4.1.3 Bot Language Understanding

The implementation of two concepts of LU, a LUIS-based and a grammar-based approach, is presented in the following. The performance of both techniques will be compared in the evaluation section of this thesis. First, the LUIS approach is examined.

**LUIS-Based System**

Prior to the integration of LUIS to the bot, the development of the LUIS model, applied in this work is described. First of all, the possible intents of a user while defining his workout goal are created. The intents are designed according to the semantic slots of the domain, specified in section 3.3 of the previous chapter. The semantic slots themselves are used as entities in the LUIS model. An intent can be mapped to none, one, or more entities. For this model, 9 intents and 10 entities have been created, for which the following mappings exist:

- **AchieveAbstractEffect → {AbstractGoal}**: When a user informs the systems about an overall goal he want to achieve, the type of goal is stored in the AbstractGoal entity.
- **AddressWorkoutType → {WorkoutType}**: When a user addresses a type of workout in his utterance, a specific WorkoutType-value should be present in the input and stored as WorkoutType entity.
- **AddressSpecificPain → {SpecificPain}**: When a user utters that he suffers from a certain pain, the type of pain should be recognized as entity and stored in the SpecificPain property.
- **AdjustPreferences → {Preferences}**: When a user states his preferences, the respective preferences are stored in the Preferences entity.
- **AskForHelp → {}**: When a user asks for help during the progress of a dialogue, this is considered as meta-dialogue and no entity is mapped to this intent.
- **Confirmation → {AffirmInfo, DenyInfo}**: When a user is asked to confirm information, he can either affirm or deny the information. If the confirmation is correct, the value is stored in the AffirmInfo entity. By contrast, if the confirmation is false, the value is a DenyInfo entity property.
- **None → {}**: The None intent is used in the case that the user utterance is out-of-domain and cannot be used in this model’s context.
4 System Design and Implementation

- **SetLimitations → \{TrainingDates, DurationPerDay, FitnessLevel\}**: A user may address constraints regarding his workout by informing the system about his desired dates of training, duration of workout per day, and his fitness level. These constraints are stored, consequently, in their respective entities.

- **WellDefineSpecificMuscle → \{SpecificMuscle\}**: When a user desires to train a muscle or muscle group, the specified muscle is recognized and stored as a SpecificMuscle entity.

For training the LUIS model, 125 typed utterances are used. With the model being trained, the next step in the development process is the integration of the cloud-service in the bot framework. For this purpose, a function is created, which takes the recognized text as string-object input, forwards it to the LUIS endpoint and returns a JSON object, which contains a list in each case for the intents and entities as seen in 3.6. The method is called in OnRecordComplete after the speech recognition. The source code of the function is presented in the following:

```csharp
public static async Task<T> RequestAsync<T>(string input)
{
    var strEscaped = string.Empty;
    if (input != null)
        strEscaped = Uri.EscapeDataString(input);
    else
        strEscaped = "is not a input";

    var url = "$https://api.projectoxford.ai/luis/v1/application?id=60f7cd54-deae-4265-b28e-df81d4702fdf&subscription-key=ad84a633742444393fc18b89f6fe5f8&q={strEscaped}";

    using (var client = new HttpClient())
    {
        var response = await client.GetAsync(url);

        if (response.IsSuccessStatusCode)
        {
            var content = await response.Content.ReadAsStringAsync();
            return JsonConvert.DeserializeObject<T>(content);
        }
    }
    return default(T);
}
```
First, it is analyzed if the input string exists (line 4-7). If this is not the case, the string is set to "is not a input", which always yields a None intent from LUIS. Afterwards, the string is embedded in the target URL, which forms the LUIS endpoint (line 9-13). As well as the speech recognition platform, the use of this endpoint also requires a subscription. A free LUIS endpoint may be called 10000 times a month. In order to send an HTTP messages to the server, an instance of the "HTTPClient"-base class is created and a "Get" request is send to the LUIS endpoint to process the user utterance (line 15, respectively 17). If a connection to the cloud-service could be established (line 19), the result of the request is read out as a string and returned as a JSON object (line 21-23).

For the extraction of the intents and entities from the object, it is necessary to parse the JSON data. As in the current version of LUIS an utterance may only be labeled with one intent, another way of conceiving multiple intentions in one utterance has to be found. For this purpose, the n-best list of the intents and the possible mappings of the entities to the respective intents are used. The n-best list alone cannot be used because the confidence score of the intents after the first one are usually far to low, in order to be sure that the result is correct. Therefore, the intent-entity mapping is used. From the detected entity, the appropriate intent can directly be inferred, which boosts the confidence of the specific intent. For handling multiple intents, the following parser has been implemented:

```csharp
var result = await LuisSkypeCall.RequestAsync<LuisResult>(recognizedText);
int entityCount = 0;
foreach (EntityRecommendation eRec in result.Entities)
{
    string entityResult = eRec.Entity;
    double entityResultScore = eRec.Score;
    string entityResultType = eRec.Type;

    string intentResult = result.Intents.ElementAt(entityCount).Intent;
    double intentResultScore = result.Intents.ElementAt(entityCount).Score;

    entityCount++;

    string giveBack += intentResult + "$" + intentResultScore + "$" + entityResult + "$" + entityResultScore + "$" + entityResultType + "$";
}
```
For each existant entity in the result list of the JSON object, the individual properties are extracted (line 6-8). This includes the `entityResult`, which represents the recognized value of the slot `entityResultType`, and the confidence `entityResultScore` of the slot-value pair. For example, “gain stamina” would be the `entityResult` for the `entityResultType` “AbstractGoal”, with an `entityResultScore` of 0.87. The property that the positions of the items of the entity list always correspond to the positions of their mapped intents of the n-best list can be used to extract the intents from the JSON object (line 12-13). “AchieveAbstractEffect”, for example, would be stored in `intentResult` with an `intentResultScore` of 0.91, which is noted at the position of the respective entity. All of the items are then written to a `giveBack` string, which is extensible for handling multiple intents (line 17-19). The “&” character indicates the end of an intent-entity mapping, whereas the “$” character separates each item. If no entity is detected, only the best intent and its confidence are written to `giveBack`. The other items are then left empty.

After this extraction, the `giveBack` string object is processed to a list of array objects, with each string-array `msgArray` representing one intent-entity mapping with separated items. Using the example, the array would consist of the following items:

- `msgArray[0] = “AchieveAbstractEffect”`
- `msgArray[1] = “0.91”`
- `msgArray[3] = “0.87”`

This format is used to transform the item string objects of `giveBack` to `UserIntention` or `Entity` objects for more efficient dialogue handling. As multiple intentions in one utterance are possible, the intents and their respective entities are stored in one `UserUtterance` object per user input. In figure 4.4, the relations between the `UserIntention`, `Entity` and `UserUtterance` classes are depicted in an UML diagram, which will be used to describe the string-`UserIntention` and string-`Entity` transformation.

The start of the transformation is the `createUserIntention` function of the `UserIntention` class. In `OnRecordCompleted`, this method is called for each `msgArray`. The function creates a new `UserIntention` object and retrieves the `intentResult` (msgArray[0]) and `intentResultScore` (msgArray[1]) strings from the array. Afterwards, the `intentType` property is set according to the specific `IntentType` enum, which matches the name of the result string (“AchieveAbstractEffect”). The `intentConfidence` property is set according to the `intentResultScore` string.

In order to get the corresponding entity, `createEntity` of the `Entity` class is called within the function. Here, the items of `msgArray`, concerning an entity are retrieved. Depending on the name of `entityResultType` (msgArray[4]), a new children object of the `Entity` class, e.g., `AbstractGoal`, is created. The value of `entityResult` (msgArray[2]) is then attached to the object as `entityString`, with the respective `entityResultScore` (msgArray[3]) as `entityConfidence`. The created object is then set as `entityName` property of its mapped intent.
4.1 Creation Of Cloud-Based Spoken Dialogue System

Figure 4.4: The UML diagram for the UserIntent, Entity and UserUtterance classes. This classes form the compact representation of the semantics of an user utterance, and contain methods to manipulate their values.
4 System Design and Implementation

For a compact representation of an utterance, the UserIntention objects, created for each msgArray, are added to a list of intents in an UserUtterance object. Hence, one UserUtterance is created per user input and forwarded to the dialogue manager for further processing. However, prior to the description of DM for the bot, a closer look is taken on the grammar-based approach of the comparative system.

Grammar-Based System

The LUIS model is replaced by a hand-crafted grammar as LU component for the grammar-based approach. For the design of the grammar, a key-value collection is used, as presented in the following source code snippet:

```csharp
Dictionary<string,string[]> grammar = new Dictionary<string,string[]>()
{
"AbstractGoal", new string[] { "gain stamina", "gain weight", "lose weight",...},
"TrainingDates", new string[] { "Mondays", "Tuesdays", "Wednesdays", ...},
"DurationPerDay", new string[] { "five", "ten", "fifteen", "twenty",...}
...```

Here, the semantic slot of each entity is used as key of the dictionary, whereas an array of specifications of a slot is used as the corresponding value. The slot values are chosen in accordance to the trained data, mentioned in the previous section. As the intention of an utterance cannot be detected by using this grammar, “Pseudo” is used as intentType for each entity. For extracting the semantic content of an user input in this system, a different approach to the LUIS-based system had to be implemented.
Therefore, the parsing of the recognized text is handled in a different way, which is described in the following pseudo code:

```csharp
private static UserUtterance parseGrammarString(string recognizedText)
{
    List<string> responseGrammar = new List<string>();
    Dictionary<string, string[]> grammar = new Dictionary<string, string[]>()

    if (s != null)
    {
        foreach (KeyValuePair<string, string[]> kvp in grammar)
        {
            string[] parserArray = kvp.Value;
            foreach (string grammarElement in parserArray)
            {
                if (s.Contains(grammarElement))
                {
                    string parsedgrammarElement = kvp.Key + "$" + grammarElement;
                    responseGrammar.Add(parsedgrammarElement);
                }
            }
        }
    }

    // ... 
}
```

Here, it is examined if a particular semantic value is existant in the user utterance (line 14). Therefore, each key-value pair of the dictionary is observed separately (line 9). For example, the system gets input by the user and the ASR module recognizes the utterance, “I want to gain stamina and train on Mondays.”, which is forwarded to the parser. In a first iteration cycle, the value of the dictionary with the “AbstractGoal”-key would be examined (line 11). The parser iterates through all elements of the array until a match to a word or a word sequence in the recognized text is found (line 12-19). In case of our example, the grammar element “gain stamina” would match with a word sequence in the text. As a result, the parsed grammar element is attached to a string object along with the specific key, e.g. “AbstractGoal$gain stamina” (line 16-17). The $ character indicates the separation of entity and corresponding entity value. The result string, containing the entity, is then added to a list, which allows to store more than one entity per user utterance. For example, in our case, after parsing is completed, also “Training-Dates$Mondays” would be added to the list.
The transformation of the strings to *UserIntention* and *Entity* objects happens similar to the strategy described in the previous section. The elements of the result list are also stored in array objects, representing intent-entity mappings. The difference to the LUIS-based approach is that all created *Entity* objects are attached to *UserIntention* objects of type “Pseudo”, and because no confidence scores are available for the intents and entities, the confidence scores are fixed. The *intentConfidence* is set to 1 for each intent, as only one *IntentType* exists. As the creation of the *UserIntention* objects is fixed, the use of the `createUserIntention` function is obsolete and only `createEntity` is called, using the appropriate elements of the intent-entity mapping array. Furthermore, the *entityConfidence* value is also fixed to 0.5 for each entity.

Identically to the LUIS-based system, all *UserIntention* objects are structured in an *UserUtterance* object and passed to the dialogue manager, which handles user input independent of the kind of LU. The design of the DM module used in this work is described in the next section.

### 4.1.4 Bot Dialogue Management

For this work, a hand-crafted frame-based approach for DM is chosen for two reasons. First, for defining a goal in the workout domain, a user needs to specify certain goal items, e.g., overall goal effect and desired training dates, which can be represented as semantic slots, or here, *Entity* objects. Second, for the development of the dialogue manager no training dialogues is available. Therefore, no statistical frameworks could be used, which leaves frame-based DM as a reasonable alternative, while heavily relying on entities as state-transition criteria.

The dialogue manager used in this work is built on three classes: An abstract *State* class, containing methods for the selection of the appropriate system actions and updating the dialogue state, a *History* class, which keeps track of the dialogue’s history, and a *DialogueProfile* class, containing the dialogue description for each user, i.e., it contains the possible semantic slots of the domain and information about their state in the current dialogue.

In figure 4.5 the relations between the single classes are depicted in an UML. Based on that, the functioning of the dialogue manager is explained. The interface to the manager is the *History* class. At the beginning of each dialogue, a new instance of this class is created, as shown in the section describing the call establishment. In doing so, an empty list for *UserUtterance* objects and a new *DialogueProfile* is instantiated. Furthermore, an *OpenState* object, which inherits from the abstract class *State* is created and set as initial *currentState* property of the dialogue. After new user input has arrived, its semantic representation is captured in an *UserUtterance* object, as described in the previous sections.
4.1 Creation Of Cloud-Based Spoken Dialogue System

Figure 4.5: The UML Diagram describing the State, History and DialogueProfile classes. This classes form the dialogue manager and contain functions for receiving semantic input, choosing appropriate system actions and update state.
For processing the semantic content, this object is used as argument for the `addUtterance` function, which is called by the `History` instance, being active for this dialogue:

```java
public void addUtterance(UserUtterance utter)
{
    utteranceList.Add(utter);
    currentState.handleIntentEntity(utter, profile);
    currentState.checkProfile(this);
}
```

In a first processing step, the semantic utterance is added to the `utteranceList`, in order to keep track of the previous user actions (line 3). The next two steps update the semantic items of `DialogueProfile` and depending on that, select the next state and system action (line 4-5). In this work, the so-called “state-design-pattern” is applied for modelling an object-oriented state machine (SourceMaking, 2016). Using this pattern, several different “states” are defined, which inherit from one abstract base “state” class. This allows to implement state-specific behaviour for each class. In the following list, the states defined for the workout domain are shown:

- OpenState
- AbstractGoalState
- SpecificPainState
- TrainingDatesState
- DurationPerDayState
- FitnessLevelState
- ConfirmGoalState
- ConfirmSpecificPainState
- ConfirmTrainingDatesState
- ConfirmDurationPerDayState
- ConfirmFitnessLevelState
- SpecialConfirmState
- AskForHelpState
- EndState

These states are modelled according to the defined entities, including states to gain information about a specific semantic slot and states to confirm provided information. The `AbstractGoal` class is exemplarily assumed to be the current state in the following, in order to present the functioning of the state update and transitions between states.
Therefore, in the second processing step, the `handleIntentEntity` function of the `AbstractGoal` class is called. This method takes the current utterance as input and updates accordingly the user's dialogue profile. In our example, `handleIntentEntity` looks as follows:

```csharp
public override void handleIntentEntity(UserUtterance utter,
    DialogueProfile profile)
{
    foreach (UserIntention intent in utter.getIntentionList())
    {
        Entity entityName = intent.getEntity();

        if (entityName is AbstractGoal)
            profile.setAbstractGoal(entityName);

        if (entityName is DurationPerDay &&
            profile.getDurationPerDay() == null)
            profile.setDurationPerDay(entityName);

        if (intent.getIntentType() ==
            UserIntention.IntentType.None) {
            profile.setOutofGrammar(true);
        }
    }
}
```

This method iterates through all intentions in an `UserUtterance` (line 4) and decides depending on the type of the `Entity` object of the current `Intent`, which semantic slots of the user's profile to fill (line 6-16). In order to prevent entities to be overwritten, it has to be checked, whether a value for an `Entity` object is already existing in the user's dialogue profile. If this is not the case, the particular `Entity` may be filled (line 11-13). For example, when the utterance contains an `Intent` with `AbstractGoal` as entity, the “AbstractGoal” slot of the instance of `DialogueProfile` is set. In this case, the `AbstractGoal` entity cannot be overwritten, as the state itself makes sure that a value for this `Entity` has not been provided yet. For the opposite case, that an unknown action or no input by the user has been provided during a user's turn, the variable for “OutofGrammar” is set true, indicating an no out-of-domain move (line 17-20). The final processing step is the transition to the next state, depending on the provided slots.
For this reason, the `checkProfile` function checks which entities have already been set in the `DialogueProfile`. For example, the method in `AbstractGoal` is considered:

```csharp
public override void checkProfile(History h)
{
    LocalDialogueProfile profile = h.getProfile();

    if (profile.getAbstractGoal() != null
        && profile.getSpecificPain() == null)
    {
        if (profile.getAbstractGoal().getConfidence() < 0.85)
            h.setState(new ConfirmGoalState(h));
        else
        {
            if (profile.getTrainingDates() == null)
                h.setState(new TrainingDatesState());
            else if (profile.getDurationPerDay() == null)
                h.setState(new DurationPerDayState());
            else if (profile.getFitnessLevel() == null)
                h.setState(new FitnessLevelState());
        }
    }...
```

In our example, the `AbstractGoal` entity has been set in the profile, i.e., the value is unequal "null". As a result, the method steps into the if-clause in line 5. As the entities `AbstractGoal` and `SpecificPain` may not occur in the same dialogue, further checking is necessary (line 6). Afterwards, it is observed if the confidence of the given `Entity` object is below a particular value, here 0.85 (line 8). If the score is below this value, the next state in the dialogue would be a confirmation state in order to validate the given user input (line 9). Otherwise, it is assumed that a confirmation is not necessary, which results in a transition to a state, where new information is requested (line 10-20). For example, the confidence score of `AbstractGoal` is greater than 0.85 and also the values for the entities `TrainingDates` and `FitnessLevel` have already been provided. As consequence, no confirmation will take place and `FitnessLevelState` is chosen as next `State` object, where the user will be asked to give information about his fitness level.

In case that the user input is out-of-domain, the next state is set to `AskHelpState`, where the user receives help, regarding what he may say at this step of the dialogue. For the LUIS-based system, there also exists another option. If only the intent has been detected with a high confidence, but no corresponding entity, the next state is set to `SpecialConfirm`. In this state, the user is provided help by taking into account the recognized intention.
The next step in the dialogue process is to respond to the user with a system action. For this purpose, the `getResponse` function of a `State` object is used, for creating a system action as string-object. In our example, a state transition from `AbstractGoalState` to `FitnessLevel` occurs. Hence, the response function of the fitness-level state is called, which would return the string, “What is your fitness level?” as system response. This string is used in `OnRecordCompleted` of the Skype Calling API for prompting the message to the user. Therefore, after all previously described routines have been performed (ASR, LU, DM), a new workflow is created. This workflow executes the `GetPromptForText` function by using the response string provided by the dialogue manager as input. After the prompt is played to the user, the dialogue cycle starts anew until the end of the dialogue is reached. Based on this described dialogue process, a mixed-initiative dialogue has been designed for both systems. An overview over the design is provided in the next section.

### 4.2 Mixed-Initiative Dialogue Design

For the evaluation of the systems, a mixed-initiative dialogue in the workout domain is designed for both systems. In the context of this work, the mixed-initiative dialogue strategy is realised by starting each dialogue with an open question and letting the user give more complex user input, i.e., an utterance may contain more than one intention and hence more than one entities. This allows the user to give the system all necessary information in just one dialogue turn. In order for the systems to be comparable, the flow of the dialogues is designed as equal as possible. The only difference between the two designs is the use of the confidence score generated by the LUIS API in the LUIS-based system. The first design, presented in this section, is the LUIS-based dialogue design. After that, the grammar-based design is presented.

#### 4.2.1 LUIS-Based Dialogue Design

In figure 4.6, the dialogue flow of the LUIS-based system is depicted. The dialogue starts with an open request (“How may I help you?”) by the system. As response, the user may give as input a subset of entities. In case that the system has not understood the user and hence the speech input is out-of-domain, the system helps the user by giving information about possible answers and continues with an open request. After new recognized input has arrived, the system checks, whether there are mandatory entities (`AbstractGoal` or `SpecificPain`, `TrainingDates`, `DurationPerDay`, `FitnessLevel`) in the user input and if their confidence scores are above 0.85. As the other possible entities are optional, they are not explicitly asked for and require no confirmation. If the confidence score of a mandatory entity is below 0.85, the system asks for confirmation. There, the user may affirm correct or deny incorrect recognized information. Furthermore, the user may also give information about a new entity in addition to the confirmation. If the user’s input is out-of-domain, help is provided and the confirmation question is posed again.
Figure 4.6: Dialogue Flow of the LUIS-Based System, representing a mixed-initiative dialogue design.
As a result to an affirmation of an entity, the value for this entity is locked, i.e., a confirmed entity may not be overwritten and will not be asked for again over the course of the dialogue. A rejection of information deletes the value for the misrecognized entity, which enables the system to ask for it again during the course of the dialogue. After the user has affirmed or denied an entity, the system checks, whether there are still unconfirmed values and asks, if existing, explicitly for their confirmation.

Otherwise, the system looks for unfilled slots. In case that values for all mandatory entities have been provided, the system presents the user their goal and ends the dialogue. If there are still missing values, the system explicitly asks for a specific entity but allows the user to also give information about other, not asked and not yet confirmed information until all mandatory information is provided.

For better understanding, consider the following the example dialogue:

**system**: Hello, I’m the Workout Bot! How may I help you?

**user**: I want to gain more stamina and train two days a week. (AbstractGoal-Confidence: 0.91, TrainingDates-Confidence: 0.87)

**system**: How long do you want to train per day?

**user**: Twenty minutes per day. (DurationPerDay-Confidence: 0.7)

**system**: You want to train twenty minutes per day. Is that correct?

**user**: Yes.

**system**: What is your fitness level?

**user**: I’m a believer. (Not recognized)

**system**: I did not understand you. You can say, for example beginner or advanced. What is your fitness level?

**user**: I’m a beginner and want do upperbody workout. (FitnessLevel-Confidence: 0.96)

**system**: This is your goal: AbstractGoal = Gain more stamina, TrainingDates = Two days a week, DurationPerDay = Twenty minutes, FitnessLevel = Beginner.

In this dialogue, the user responds to the open request by giving information about his abstract overall goal and the desired number of training dates. As both confidence scores are higher than 0.85, the system accepts the information and continues by asking the user about his desired duration of training per day. This time the confidence score of the answer is below 0.85. This results in a confirmation question, which the user affirms. After that, the system questions the user about his fitness level, but the user input is misrecognized and out-of-domain. Therefore, the user receives help and is asked again about his fitness level. The next user response is then recognized correctly. As values for all mandatory entities have been provided at this dialogue step, the system informs the user about his defined goal and closes the dialogue. In the next section, the grammar-based dialogue design is described.
4.2.2 Grammar-Based Dialogue Design

In Figure 4.7, the dialogue flow of the grammar-based system is depicted. The difference to the LUIS-based system is the missing of the confidence scores for the recognized intentions and entities calculated by the LUIS model. Therefore, the confidence for each recognized entity is strictly set to 0.5, to make sure that no falsely recognized entity is accepted by the system. Equally to the LUIS-based system, the dialogue starts with an open question, to which the user may respond with an arbitrary number of entities. After the system has received allowed input, it asks for confirmation for all given mandatory entities automatically because the confidence score is always below 0.85. As a response, the user may again either affirm or reject the item, for which confirmation was asked for, and additionally provides a value for a non-filled entity. The system asks for confirmation as long as there exists unconfirmed information.
In case that all mandatory slots have been confirmed, the system returns the goal to the user and ends the dialogue. If there exist missing values for particular entities, the system specifically requests them until all slots are filled. Consider the following representative dialogue:

**system:** Hello, I’m the Workout Bot! How may I help you?
**user:** I want to gain more stamina and train two days a week.
**system:** You want to gain more stamina. Is that correct?
**user:** Yes.
**system:** You want to train two days a week. Is that correct?
**user:** Yes.
**system:** How long do you want to train per day?
**user:** Fourty minutes per day.
**system:** You want to train thirty minutes per day. Is that correct?
**user:** No, I want to train fourty minutes.
**system:** You want to train thirty minutes per day. Is that correct?
**user:** Yes.
**system:** What is your fitness level?
**user:** I’m a beginner.
**system:** You want are a beginner. Is that correct?
**user:** Yes.
**system:** This is your goal: AbstractGoal = Gain more stamina, TrainingDates = Two days a week, DurationPerDay = Twenty minutes, FitnessLevel = Beginner.

The grammar-based system starts the dialogue like the LUIS-based system with an open question. After the user has provided values for two entities (“gain more stamina” and “two days a week”), the system explicitly asks for confirmation of these slots. As both slots are confirmed, the system then asks the user about the desired duration of training per day. There, the system misrecognizes the user input by asking for confirmation of the value “thirty minutes”, instead of the true value “fourty minutes”. As a result, the user denies the believed value “fourty minutes” and informs the system about the intended value in the same utterance. This time the system understands the correct value and asks for confirmation, to which the user responds with an affirmation. In the following dialogue step, the user is requested for the final mandatory entity. The user provides the necessary information and is asked to confirm this slot in the next step. The user affirms, and as all mandatory entities have been provided, the system presents the defined goal to the user and ends the dialogue. In the following chapter, the design of the evaluation of the two systems is described and the results are presented.
5 Evaluation and Results

The evaluation of the two bot systems is conducted online, using the survey creation tool LimeSurvey (Schmitz et al., 2012). In order to take the survey, the participants require a Skype account for enabling the interaction with the bots. The bots are evaluated independently by dividing the test subjects in two groups. Each group interacts only with one bot-type (grammar-based or LUIS-based) throughout the evaluation. In the following, the procedure of the evaluation and the metrics of interest are described. Afterwards, the results are presented.

5.1 Evaluation

During the online survey, the participants interact five times with the respective bot. The first interaction serves as a training scenario, where the caller follows a prescribed, static dialogue structure in order to get familiar with the functioning of the system. As this step only has a training purpose, it is not included in the evaluation results.

In the following three interactions, the users are not bound to a specific dialogue structure. The independent variable in these dialogues constitutes the complexity of the user input after the initial welcome prompt by the system. The task of the first evaluated interaction is to provide one of the mandatory entities (AbstractGoal or SpecificPain, TrainingDates, DurationPerDay, FitnessLevel) at each dialogue turn. In order to complete the task of the second dialogue, the participants need to provide two of the mandatory entities after the welcome prompt. Finally, for task completion of the third interaction, three mandatory entities need to be given at once at the beginning. The fourth evaluated dialogue is free from any restrictions and aims to identify the complexity of input the participants prefer for interaction with the particular system. Here, the task success is to successfully complete the dialogue.

The objective performance of each bot-type is evaluated by using five dependent variables: the average dialogue length (ADL), the dialogue completion rate (DCR), the task success rate (TSR), the number of reprompts (#Reprompts), and the number of confirmations (#Confirmations). ADL is the average number of turns to complete a dialogue. The ratio of the number of completed dialogues to the number of dialogues, in which the user has hung up during the ongoing interaction, is denoted DCR. TSR describes the ratio of the number of dialogues, where the task of the specific dialogue is fulfilled as previously described, to the number of all dialogues. The count of reprompted system actions, caused by misrecognized user input per completed dialogue is denoted #Reprompts, while the count of confirmations per completed dialogue is denoted #Confirmations.
5 Evaluation and Results

In order to compute these metrics, the system and user utterances are logged and saved in a database after each dialogue turn. In contrast to the objective performance, also the user experience with each bot is evaluated. Therefore questionnaires, similar to those described in ITU-T Rec. P.851 (2003) are applied. At the beginning of the survey, the participants answer background related questions, which are relevant to the experiment. For example, personal information, like age, gender, language proficiency, and system-related information, regarding the experience with SDSs, is questioned. After each interaction, the test subjects fill in a questionnaire about the individual experience with the particular dialogue. There, the participants have to rate statements about the system’s interaction behaviour, perceived system personality, impression on the user, etc., on seven-point likert-scales, ranging in integer steps from “strongly disagree”, encoded to the number -3, to “strongly agree”, encoded to the number +3. At the end of the evaluation, a questionnaire related to the user’s overall impression of the system is answered. The mentioned seven-point likert-scale is used again for rating the overall experience with the system. The items to be rated comprise the perceived usability, the user's degree of enjoyment, the expected future use of the service, and so forth. For a better insight, the whole questionnaire is provided in the appendix of this thesis. In the following, the results of the evaluation are presented.

5.2 Results

In the process, 29 test subjects were evaluated. Originally, a number of 60-70 participants was aimed for, but due to updates and errors of the Bing Speech API, which led to a breakdown of the systems, the evaluation was suspended for 10 days, resulting in lower number of participants. For testing the LUIS-based system, 16 people with an average age of 28.00 years were evaluated, of which 7 were female and 9 were male. The youngest participant was 19 years old, while the oldest was 51 years old. The native language of 81.3 % of the subjects was german. The group rated their English language skills with a median value of 4.00 (five-point likert-scale, 1 represents “novice”, 5 represents “expert”). Furthermore, the group were beginners with SDSs with a rated median value of 2.00, also relying on a five-point likert-scale. For testing the grammar-based approach, 13 people with an average age of 28.30 years were evaluated, of which 5 were female and 8 were male. There, the youngest participant was 20 years old, while the oldest was 38 years old. The native language of 92.3 % of this group’s subjects was german. The group rated their English language skills also with a median value of 4.00, while the participants also stated to have no experience with SDSs with a rated median value of 1.00.

Both groups rated their experience in workout training as intermediate, while nine test subjects were members of a fitness studio. 21 of the participants used a laptop for the evaluation, while 6 used a PC. 2 Persons interacted with the bots via smartphone.
Table 5.1: Results of the evaluation of the LUIS-based and grammar-based bot, given by the mean values (mean) for the metrics DCR, TSR, and by the mean values (mean) and the standard deviation (STD) for the metrics ADL, #Reprompts, and #Confirmations.

<table>
<thead>
<tr>
<th>System</th>
<th>DCR mean</th>
<th>TSR mean</th>
<th>ADL mean</th>
<th>STD</th>
<th>#Reprompts mean</th>
<th>STD</th>
<th>#Confirmations mean</th>
<th>STD</th>
</tr>
</thead>
<tbody>
<tr>
<td>LUIS</td>
<td>0.78</td>
<td>0.48</td>
<td>9.98</td>
<td>4.55</td>
<td>4.02</td>
<td>3.69</td>
<td>1.98</td>
<td>0.90</td>
</tr>
<tr>
<td>Grammar</td>
<td>0.85</td>
<td>0.60</td>
<td>11.86</td>
<td>4.20</td>
<td>4.30</td>
<td>3.78</td>
<td>4.00</td>
<td>0.00</td>
</tr>
</tbody>
</table>

5.2.1 Objective Metrics

In Figure 5.1, the overall evaluation results of the objective metrics DCR, TSR, ADL, #Reprompts, and #Confirmations for both systems are depicted. For visualisation reasons the error bars are omitted from the diagram. The DCR value of the grammar-based system is 85%, which is 7% higher than the value of the LUIS-based approach. With a value of 60%, the TSR is 12% above the LUIS-based value. The ADL of the LUIS-based bot is 1.88 turns lower than the ADL of the grammar-based bot, while the difference between the mean of the number of reprompts per dialogue between both systems is 0.28. The average of the number of confirmations per dialogue is 4.00 for the grammar-based approach, as all mandatory entities need to be affirmed for dialogue completion. By contrast, the LUIS-based approach has a #Confirmations value of 1.98. A complete overview over the mean values of the results and their respective standard deviations is provided in table 5.1.
In the following, the DCR and the TSR of the four individual dialogs are observed. The values for the DCR are visualized in figure 5.2. The difference between the two systems, regarding DCR starts with the value of the grammar-based approach being 9% higher than the LUIS-based approach for the first dialogue. In the second dialogue this number increases first to 17%, but then decreases 2% for the third dialogue. In the final, fourth dialogue, the LUIS-based approach has a completion rate of 12% higher than the grammar-based bot. The absolute mean values of all dialogues for both systems can be observed in table 5.2. In figure 5.3, the success rates of both approaches is compared for each dialogue. The first and fourth dialogue result in a relatively high TSR of both systems. The grammar-based approach achieves a rate of 100% for the first and a rate of 85% for the second dialogue, while the LUIS-based system achieves a rate of 81%, respectively 75%. The TSRs of the second and third dialogue have a value of 19% for the LUIS-based bot. The values for the grammar-based bot are 31%, respectively 23%. A summary of these results can be found in table 5.3.
5.2 Results

Table 5.3: Results of the individual TSR of each dialogue given by the mean values

<table>
<thead>
<tr>
<th>Dialogue</th>
<th>LUIS mean</th>
<th>Grammar mean</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dialogue 1</td>
<td>0.81</td>
<td>1.00</td>
</tr>
<tr>
<td>Dialogue 2</td>
<td>0.19</td>
<td>0.31</td>
</tr>
<tr>
<td>Dialogue 3</td>
<td>0.19</td>
<td>0.23</td>
</tr>
<tr>
<td>Dialogue 4</td>
<td>0.75</td>
<td>0.85</td>
</tr>
</tbody>
</table>

Figure 5.3: Results of the TSR of the LUIS-based and the grammar-based system for each individual dialogue in percent.
Table 5.4: The user-chosen complexity for the fourth, individual creatable dialogue given by the median and the standard deviation. The complexity ranges from 1, which represents “one item given in one utterance after starting message”, to 4, representing “four items given in one utterance after starting message”.

<table>
<thead>
<tr>
<th>System</th>
<th>Chosen Complexity</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>median  STD</td>
</tr>
<tr>
<td>LUIS</td>
<td>1.00  0.85</td>
</tr>
<tr>
<td>Grammar</td>
<td>1.00  1.29</td>
</tr>
</tbody>
</table>

Another value of interest forms the chosen complexity of the fourth dialogue of the evaluation, where the user has no restrictions, regarding the flow of the dialogue. The values can be observed in table 5.4. Both systems have a median value of 1.00, which represents “one item given in one utterance after starting message”. The grammar-based approach has a standard deviation of 1.29, which is 0.44 higher than the value of the LUIS-based system. None of the differences between the two systems are significant by testing with the Mann-Whitney U test (Mann & Whitney, 1947). In the next section, the results of the subjective questionnaires are presented.

### 5.2.2 Subjective Metrics

In this section, the results of the questionnaire for the overall experience with the system and the significant results for the experience with the individual dialogues are provided. The full questionnaires for the evaluation of the individual dialogues can be found in the in appendix of this thesis. In table 5.5, the median values for the ratings on the seven-point likert-scale ranging from the integer values -3 to +3, and their respective standard deviations are depicted. Negative median values represent a disagreement with the statement of the questionnaire, while positive values represent a agreement with the statement.

Both system receive negative ratings for the overall user experience in terms of future use of the system, reliability, the enjoyment of the dialogues, the handling of the systems, the system’s way of expression, and the overall experience with the systems. Furthermore, the users strongly agree to prefer a human operator with a median value of 3.00 and prefer to use another source of information with a median value of 2.00 for the LUIS-based system and a 3.00 median value for the grammar-based approach. Besides, the users are relatively unimpressed with the systems’ performances and perceive this possibility for obtaining information rather not helpful. The user groups also agree that the systems are largely in control of the dialogue flow. The value of the median for rating the calls as helpful shows a large variance for both systems. In addition, the users show large variance in their level of knowledge about the systems’ possibilities. However, both systems receive rather positive ratings, concerning politeness.
Table 5.5: The results of the final questionnaire, which covers the overall interaction experience with both systems. The rating is relying on a seven-point likert-scale ranging from the integer values -3 to +3. For each statement of the questionnaire the median and the standard deviation is provided. Where indicated with *, the differences are significant with $\alpha < 0.05$ (applying the Mann-Whitney U test (Mann & Whitney, 1947)).

<table>
<thead>
<tr>
<th>Statement</th>
<th>LUIS median</th>
<th>STD</th>
<th>Grammar median</th>
<th>STD</th>
</tr>
</thead>
<tbody>
<tr>
<td>Your overall impression with the system was excellent</td>
<td>-1.50</td>
<td>1.83</td>
<td>-2.00</td>
<td>1.55</td>
</tr>
<tr>
<td>The system’s way of expression was unclear</td>
<td>-1.00</td>
<td>1.74</td>
<td>-2.00</td>
<td>1.61</td>
</tr>
<tr>
<td>The system reacted politely</td>
<td>1.00</td>
<td>1.96</td>
<td>2.00</td>
<td>1.16</td>
</tr>
<tr>
<td>You would have expected more help from the system</td>
<td>0.00</td>
<td>1.49</td>
<td>1.00</td>
<td>2.02</td>
</tr>
<tr>
<td>Misunderstandings could be cleared easily*</td>
<td>-2.00</td>
<td>1.86</td>
<td>0.00</td>
<td>1.61</td>
</tr>
<tr>
<td>The system controlled the flow of the dialogue</td>
<td>1.50</td>
<td>1.41</td>
<td>2.00</td>
<td>1.44</td>
</tr>
<tr>
<td>You were able to handle the system without any problems</td>
<td>-1.50</td>
<td>1.76</td>
<td>-1.00</td>
<td>1.68</td>
</tr>
<tr>
<td>Regarding the dialogues, you are unimpressed</td>
<td>1.00</td>
<td>1.44</td>
<td>1.00</td>
<td>1.49</td>
</tr>
<tr>
<td>You enjoyed the dialogues</td>
<td>-2.00</td>
<td>1.74</td>
<td>-2.00</td>
<td>1.70</td>
</tr>
<tr>
<td>You feel adequately informed about the system’s possibilities</td>
<td>0.50</td>
<td>2.05</td>
<td>0.00</td>
<td>1.55</td>
</tr>
<tr>
<td>The skype calls with the system were worthwhile</td>
<td>-0.50</td>
<td>1.90</td>
<td>0.00</td>
<td>1.27</td>
</tr>
<tr>
<td>You rate the system as reliable</td>
<td>-1.00</td>
<td>1.98</td>
<td>-1.00</td>
<td>1.44</td>
</tr>
<tr>
<td>You prefer to use another source of information</td>
<td>2.00</td>
<td>1.71</td>
<td>3.00</td>
<td>1.52</td>
</tr>
<tr>
<td>You perceived this possibility for obtaining information as helpful</td>
<td>-0.50</td>
<td>1.65</td>
<td>-1.00</td>
<td>1.85</td>
</tr>
<tr>
<td>The handling of the system was complicated</td>
<td>-0.50</td>
<td>2.10</td>
<td>-1.00</td>
<td>1.45</td>
</tr>
<tr>
<td>You prefer a human operator</td>
<td>3.00</td>
<td>1.39</td>
<td>3.00</td>
<td>1.25</td>
</tr>
<tr>
<td>In the future, you would use the system again</td>
<td>-1.50</td>
<td>1.93</td>
<td>-2.00</td>
<td>1.80</td>
</tr>
</tbody>
</table>
5 Evaluation and Results

As all ratings have large standard deviations and the values of the systems’ medians are comparable, only one statement shows significant differences between the two systems by applying the Mann-Whitney U test (Mann & Whitney, 1947). The users agree significantly that the grammar-based approach handles misunderstandings during the dialogue easier than the LUIS-based approach with a median value of 0.00 to -2.00.

Table 5.6: An overview of the significant results of the questionnaire for the individual dialogues with both systems. The differences are significant with $\alpha < 0.05$ (applying the Mann-Whitney U test (Mann & Whitney, 1947)). Where indicated with *, The differences are significant with $\alpha < 0.1$.

<table>
<thead>
<tr>
<th>Statement</th>
<th>LUIS median</th>
<th>STD</th>
<th>Grammar median</th>
<th>STD</th>
</tr>
</thead>
<tbody>
<tr>
<td>You felt well understood by the system (Dialogue 1)</td>
<td>-2.00</td>
<td>1.63</td>
<td>0.00</td>
<td>1.63</td>
</tr>
<tr>
<td>In your opinion, the system processed your specifications correctly (Dialogue 1)</td>
<td>1.00</td>
<td>1.77</td>
<td>2.00</td>
<td>1.03</td>
</tr>
<tr>
<td>The system’s behaviour was always as expected* (Dialogue 1)</td>
<td>-0.50</td>
<td>2.09</td>
<td>1.00</td>
<td>1.14</td>
</tr>
<tr>
<td>The system makes a lot of mistakes* (Dialogue 1)</td>
<td>0.50</td>
<td>1.98</td>
<td>-1.00</td>
<td>1.18</td>
</tr>
<tr>
<td>You perceived the dialogue as natural* (Dialogue 1)</td>
<td>-2.00</td>
<td>1.31</td>
<td>1.00</td>
<td>2.16</td>
</tr>
<tr>
<td>The course of the dialogue was clear (Dialogue 1)</td>
<td>0.50</td>
<td>1.59</td>
<td>2.00</td>
<td>1.36</td>
</tr>
<tr>
<td>You had to concentrate in order to understand what the system expected from you (Dialogue 3)</td>
<td>0.50</td>
<td>2.16</td>
<td>-2.00</td>
<td>1.25</td>
</tr>
<tr>
<td>You had to concentrate in order to understand what the system expected from you (Dialogue 4)</td>
<td>0.50</td>
<td>1.75</td>
<td>-2.00</td>
<td>1.55</td>
</tr>
<tr>
<td>The system utterances were too long* (Dialogue 2)</td>
<td>0.00</td>
<td>1.96</td>
<td>-2.00</td>
<td>1.77</td>
</tr>
<tr>
<td>The system utterances were too long (Dialogue 3)</td>
<td>0.50</td>
<td>1.67</td>
<td>-1.00</td>
<td>1.63</td>
</tr>
<tr>
<td>The system utterances were too long* (Dialogue 4)</td>
<td>0.00</td>
<td>1.62</td>
<td>-2.00</td>
<td>1.82</td>
</tr>
</tbody>
</table>

The significant and approximately significant differences between the two approaches on dialogue level is presented in table 5.6. Most of the significant effects of this evaluation occur in the first dialogue between the system and the user. There, the users feel better understood by the grammar-based system than by the LUIS-based system with a median value difference of 2.00 points. Moreover, they perceive the dialogue course of the grammar-based bot clearer and agree that the system processes their specifications more correctly. The dialogue with the grammar-based approach is also perceived more natural and the users perceive that the system behaves more in a way that they have expected. By contrast, the grammar-based approach is rated to make more mistakes than the LUIS-based system. In the dialogues three and four, the user state that they need less concentration in order to understand what the LUIS-based system expected from them than the grammar-based approach.
Furthermore, they perceive the system utterances of the LUIS-based approach to be longer. In the next section, the found results are discussed.

5.2.3 Discussion

The results show that almost no significant differences between the LUIS-based and the grammar-based system are found. With reference to the objective evaluation of the systems, the grammar-based approach yields a better performance, regarding DCR and TSR. However, the differences are insignificant, hence they could be coincidental. The number of reprompts and confirmations is almost identical for both system, when the difference of the number of confirmations is added to the ADL of the LUIS-based approach, as the grammar-based approach always requires 4 confirmations, contrary to compared system. The values for the DCR are acceptable, which shows that the system is stable. By contrast, the TSR is far from being acceptable.

A reason for the low percentage is that most of the users do not try to accomplish the given tasks, as they simply overlook or do not understand the instructions, provided in the online study. This is especially the case in the second, third and fourth dialogue, where the users often repeat the task of the first dialogue by providing similar dialogues. For this reason, it is impossible to receive the necessary data for the evaluation of the complex user input behaviour of both systems. The lack of correct task execution is also visible in table 5.4, which shows that the complexity of the last “free” dialogue is nearly always in the range of the complexity of the first dialogue task. An explanation for the bad task completion could be too vague or unclear instructions in the dialogue scenario description, which could result in overstrained participants. Another explanation, which is quite typically in online studies, is that the users simply ignored and overlooked the instructions of the dialogues following the first one, as they are similarly structured. Apart from the evaluation design, there may also exist another explanation for the low task success of the systems. Therefore, consider the screenshot of an example dialogue in figure 5.4, which demonstrates the unfortunately common mistakes of the ASR module during the evaluation process. In this particular example, the dialogue length is already 17 turns with the user only having specified two of the four necessary mandatory items, caused by great speech recognition problems. This problems could lead to mistrust of the user in the system, which delivers another possible explanation why the users mostly stick to one item per utterance and not dare to provide more complex input.

During the evaluation process, this problem often occurred with users, which used a laptop microphone. Unfortunately, 21 of the 29 test subjects used a laptop for the evaluation of the systems, while 2 subjects used their smartphone and 6 participants used their home computer with a headset microphone. As a correlation between bad ASR and user experience is common in the context of SDSs, the type of the used speech input device could provide an explanation for the disastrous user ratings of the subjective questionnaires. Therefore, the subjective ratings of five statements are examined again, with the used device (“Laptop”, or “No Laptop”) as dependent variable.
Figure 5.4: Dialogue from evaluation data in order to demonstrate the ASR mistakes. First utterance is from the user, while the response is the recognized text by the ASR module. "NULL" implies that the recording of the user input failed. Test subject used laptop microphone for speech input.
Table 5.7: Medians for five representative statements of the evaluation, depending on the used device type “Laptop”, or “No Laptop”. The rating is relying on a seven-point likert-scale ranging from the integer values -3 to +3.

<table>
<thead>
<tr>
<th>Statement</th>
<th>Dialogue 1</th>
<th>Dialogue 2</th>
<th>Dialogue 3</th>
<th>Dialogue 4</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>No Laptop</td>
<td>Laptop</td>
<td>No Laptop</td>
<td>Laptop</td>
</tr>
<tr>
<td>In your opinion, the system processed your specifications correctly</td>
<td>1.00</td>
<td>1.00</td>
<td>0.00</td>
<td>-1.00</td>
</tr>
<tr>
<td>The system makes a lot of mistakes</td>
<td>-2.00</td>
<td>-1.00</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>The system reacted too slowly</td>
<td>-2.50</td>
<td>2.00</td>
<td>-2.00</td>
<td>2.00</td>
</tr>
<tr>
<td>Overall, you are satisfied with the dialogue</td>
<td>1.00</td>
<td>0.00</td>
<td>1.50</td>
<td>-1.00</td>
</tr>
<tr>
<td>You perceived the dialogue as pleasant</td>
<td>0.00</td>
<td>0.00</td>
<td>0.50</td>
<td>-1.00</td>
</tr>
</tbody>
</table>

In table 5.7, the medians of five representative statements are presented, which are used to indicate the dependency of the systems on the speech input device. Contrastly to the participants using a laptop, the test subjects using no laptop observe that the systems process their specifications overall more correctly and make less mistakes. This difference is most obvious in the second dialogue with a margin of 3.00 points. This phenomena is mostly due to the quality of the input microphones of the laptops, in contrast to the better functioning headsets and smartphone inputs. Furthermore, the reaction time of the systems differs greatly in favor of the non-laptop users, which could be due to internet speed variations. In the first dialogue, this margin reaches its peak at a 4.50 points difference between the device types. As a result, the users are more satisfied with the dialogues, when not using the laptop’s microphone for speech input. This shows that the device type and hence the speech input device have a great impact on the perception of the systems’ functions. As 72.4 % of the subjects used laptops during the evaluation, this may primary be responsible for the bad user experience.

Aside from the incorrect task execution of most users and the speech recognition problems, which aroused from low-quality laptop speech input devices, also the domain itself could be a reason why there exist almost no significant differences between the use of the LUIS-based system and the grammar-based system. The use case scenario of this thesis seems to be less complex in order to use the advantages of LUIS, as the user only provides information to the system and never needs to request anything or needs to change the domain itself. Furthermore, the active learning property of LUIS is not included in the evaluation and hence has no effect on the results, concerning the LUIS-based system.

The significant difference between the systems, regarding the clarification of the misunderstandings probably results from the different applied confirmation strategies. As the grammar-based approach uses confirmations for every mandatory entity, this approach seems to better react to misunderstandings. The differences in the interaction behaviour of the two systems during the first dialogue may also be caused by the more strict dialogue structure of the grammar-based approach, which limits the possibility of misunderstandings during the dialogue.
5 Evaluation and Results

The differences in the statements, regarding the length of the system utterances only make sense, if the participants mean the reaction times of the systems. For both systems the same system utterances are created. In case that the user want to address the reaction time, the differences most possibly result from the duration of the connection to the LUIS server, as the LU of the grammar-based approach is implemented within the bot. However, these results are rather unreliable, caused by the previous mentioned problems. Furthermore, for more significant results, also the number of test subjects needs to be higher. Due to the problems with the Bing speech service, the required number of participants could not be reached.
6 Conclusion

In this work, the creation and evaluation of an SDS using cloud-based cognitive services for intention and speech recognition is presented. For intention recognition from spoken utterances, Microsoft's LUIS was applied. Hence, a LUIS model was built and trained with utterances from the fitness-domain. For providing speech recognition capability, the Bing Speech API was utilised. The Microsoft Bot Framework was used as interface for enabling input/output operations with LUIS, the Bing Speech API, and the DM module of the SDS. The DM was implemented as a frame-based dialogue manager and was embedded in the bot framework. For enabling natural speech input, the bot framework was extended with the Skype Calling API. This allowed the bot to be called worldwide from any device, using Skype.

For the evaluation of the cloud-based SDS, a mixed-initiative dialogue was designed in the fitness-domain. The purpose of the dialogue was to define an individual workout goal together with a user. In order to test the intention recognition approach, with regard to the system performance and user experience, depending on the complexity of the user input, a LUIS-based bot was evaluated against a grammar-based bot. This bot was created identically, with the exception of using a hand-crafted grammar, instead of LUIS for spoken LU. The evaluation was conducted in an online study, where the test subjects had to interact with one of the bots in four dialogues and answer a subjective questionnaire after each dialogue. The task of the users was to increase the complexity, i.e., the number of intention per utterance, with each dialogue.

The results of the evaluation showed almost no significant differences between the two approaches. Furthermore, a very low TSR and a negative rating on user experience for both systems was revealed. The poor results were explained with the lack of correct task execution by the participants and the unusable ASR performance of the Bing Speech API on laptops without using a headset. In addition, the degree of complexity of the designed dialogues was questioned, as they were deemed to simple for really examining the powers of an intention recognition system.

As the problems were mostly aroused by the uncontrollable variables of the evaluation setup and the complexity of the evaluation task itself, it is proposed to transfer the experimental setup from the web to the laboratory. This would allow to control the speech input device and to make sure that the task is understood and executed correctly by the user. Moreover, this would also decrease the complexity of the user's preparation for the study. For the conducted evaluation, the users had to have Skype installed on their devices and needed to add the bot their contact list. This often resulted in problematic situations, with the often inexperienced participants being already stressed before the start of the evaluation.
Another proposition is to avoid the usage of the here applied, preview cloud-based services for the evaluation. As these services are currently beta software versions, they are prone to errors and are often changed or updated by the developers, which makes them unreliable for testing. In order to fully examine the possibilities of LUIS, it is also proposed to transfer to a more complex user scenario. A multi-domain scenario would pose high demands on intention recognition services, as the user's intentions may vary from domain to domain and have to be detected correctly by the system in order to establish a reliable and trustworthy conversation.
A Appendix
Table A.1: The results of the questionnaire for the first dialogue. The rating is relying on a seven-point likert-scale ranging from the integer values -3 to +3. For each statement of the questionnaire the median and the standard deviation is provided. Where indicated with *, the differences are significant with $\alpha < 0.05$ (applying the Mann-Whitney U test (Mann & Whitney, 1947)).

<table>
<thead>
<tr>
<th>Statement</th>
<th>LUIS</th>
<th>Grammar</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>median</td>
<td>STD</td>
</tr>
<tr>
<td>You felt well understood by the system*</td>
<td>-2.00</td>
<td>1.63</td>
</tr>
<tr>
<td>You had to concentrate in order to understand what the system expected from you</td>
<td>-1.00</td>
<td>2.18</td>
</tr>
<tr>
<td>You knew at each point of the dialogue what the system expected from you</td>
<td>0.00</td>
<td>1.63</td>
</tr>
<tr>
<td>In your opinion, the system processed your specifications correctly*</td>
<td>1.00</td>
<td>1.77</td>
</tr>
<tr>
<td>The system’s behaviour was always as expected</td>
<td>-0.50</td>
<td>2.09</td>
</tr>
<tr>
<td>The system makes a lot of mistakes</td>
<td>0.50</td>
<td>1.98</td>
</tr>
<tr>
<td>The system reacted in the same way as humans do</td>
<td>-2.00</td>
<td>1.82</td>
</tr>
<tr>
<td>The system reacted flexibly</td>
<td>-1.00</td>
<td>1.75</td>
</tr>
<tr>
<td>You were able to control the dialogue in the desired way</td>
<td>-1.50</td>
<td>1.75</td>
</tr>
<tr>
<td>The system reacted too slowly</td>
<td>1.00</td>
<td>2.13</td>
</tr>
<tr>
<td>The system reacted in a friendly way</td>
<td>1.50</td>
<td>1.80</td>
</tr>
<tr>
<td>The system utterances were too long</td>
<td>1.00</td>
<td>1.86</td>
</tr>
<tr>
<td>You perceived the dialogue as natural</td>
<td>-2.00</td>
<td>1.31</td>
</tr>
<tr>
<td>The course of the dialogue was clear*</td>
<td>0.50</td>
<td>1.59</td>
</tr>
<tr>
<td>The dialogue was too long</td>
<td>-1.50</td>
<td>2.07</td>
</tr>
<tr>
<td>I sometimes wondered if I was using the right word</td>
<td>1.50</td>
<td>1.45</td>
</tr>
<tr>
<td>The course of the dialogue was smooth</td>
<td>0.00</td>
<td>1.67</td>
</tr>
<tr>
<td>Overall, you are satisfied with the dialogue</td>
<td>-0.50</td>
<td>2.18</td>
</tr>
<tr>
<td>You perceived the dialogue as pleasant</td>
<td>-0.50</td>
<td>1.92</td>
</tr>
<tr>
<td>During the dialogue, you felt relaxed</td>
<td>-1.00</td>
<td>1.36</td>
</tr>
</tbody>
</table>
Table A.2: The results of the questionnaire for the second dialogue. The rating is relying on a seven-point likert-scale ranging from the integer values -3 to +3. For each statement of the questionnaire the median and the standard deviation is provided. Where indicated with *, the differences are significant with $\alpha < 0.05$ (applying the Mann-Whitney U test (Mann & Whitney, 1947)).

<table>
<thead>
<tr>
<th>Statement</th>
<th>LUIS</th>
<th>Grammar</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>median</td>
<td>STD</td>
</tr>
<tr>
<td>You felt well understood by the system</td>
<td>-1.00</td>
<td>2.03</td>
</tr>
<tr>
<td>You had to concentrate in order to understand what the system expected</td>
<td>-1.00</td>
<td>1.97</td>
</tr>
<tr>
<td>from you</td>
<td></td>
<td></td>
</tr>
<tr>
<td>You knew at each point of the dialogue what the system expected from you</td>
<td>1.00</td>
<td>1.83</td>
</tr>
<tr>
<td>In your opinion, the system processed your specifications correctly</td>
<td>0.50</td>
<td>2.21</td>
</tr>
<tr>
<td>The system’s behaviour was always as expected</td>
<td>1.00</td>
<td>1.93</td>
</tr>
<tr>
<td>The system makes a lot of mistakes</td>
<td>1.00</td>
<td>2.16</td>
</tr>
<tr>
<td>The system reacted in the same way as humans do</td>
<td>-2.00</td>
<td>1.45</td>
</tr>
<tr>
<td>The system reacted flexibly</td>
<td>0.00</td>
<td>1.81</td>
</tr>
<tr>
<td>You were able to control the dialogue in the desired way</td>
<td>-1.00</td>
<td>1.73</td>
</tr>
<tr>
<td>The system reacted too slowly</td>
<td>1.00</td>
<td>2.00</td>
</tr>
<tr>
<td>The system reacted in a friendly way</td>
<td>0.50</td>
<td>1.79</td>
</tr>
<tr>
<td>The system utterances were too long</td>
<td>0.00</td>
<td>1.96</td>
</tr>
<tr>
<td>You perceived the dialogue as natural</td>
<td>-2.00</td>
<td>1.15</td>
</tr>
<tr>
<td>The course of the dialogue was clear</td>
<td>1.00</td>
<td>1.82</td>
</tr>
<tr>
<td>The dialogue was too long</td>
<td>-1.50</td>
<td>2.05</td>
</tr>
<tr>
<td>I sometimes wondered if I was using the right word</td>
<td>1.00</td>
<td>1.99</td>
</tr>
<tr>
<td>The course of the dialogue was smooth</td>
<td>-1.00</td>
<td>1.93</td>
</tr>
<tr>
<td>Overall, you are satisfied with the dialogue</td>
<td>-1.50</td>
<td>2.19</td>
</tr>
<tr>
<td>You perceived the dialogue as pleasant</td>
<td>-1.00</td>
<td>1.99</td>
</tr>
<tr>
<td>During the dialogue, you felt relaxed</td>
<td>-1.00</td>
<td>1.53</td>
</tr>
</tbody>
</table>
Table A.3: The results of the questionnaire for the third dialogue. The rating is relying on a seven-point likert-scale ranging from the integer values -3 to +3. For each statement of the questionnaire the median and the standard deviation is provided. Where indicated with *, the differences are significant with $\alpha < 0.05$ (applying the Mann-Whitney U test (Mann & Whitney, 1947)).

<table>
<thead>
<tr>
<th>Statement</th>
<th>LUIS median</th>
<th>LUIS STD</th>
<th>Grammar median</th>
<th>Grammar STD</th>
</tr>
</thead>
<tbody>
<tr>
<td>You felt well understood by the system</td>
<td>-1.00</td>
<td>2.09</td>
<td>-1.00</td>
<td>1.66</td>
</tr>
<tr>
<td>You had to concentrate in order to understand what the system expected from you*</td>
<td>0.50</td>
<td>2.16</td>
<td>-2.00</td>
<td>1.25</td>
</tr>
<tr>
<td>You knew at each point of the dialogue what the system expected from you</td>
<td>2.00</td>
<td>2.32</td>
<td>1.00</td>
<td>1.91</td>
</tr>
<tr>
<td>In your opinion, the system processed your specifications correctly</td>
<td>0.00</td>
<td>2.20</td>
<td>0.00</td>
<td>1.38</td>
</tr>
<tr>
<td>The system’s behaviour was always as expected</td>
<td>1.00</td>
<td>2.15</td>
<td>0.00</td>
<td>1.47</td>
</tr>
<tr>
<td>The system makes a lot of mistakes</td>
<td>0.00</td>
<td>1.78</td>
<td>0.00</td>
<td>1.92</td>
</tr>
<tr>
<td>The system reacted in the same way as humans do</td>
<td>-2.00</td>
<td>2.03</td>
<td>-2.00</td>
<td>1.50</td>
</tr>
<tr>
<td>The system reacted flexibly</td>
<td>0.00</td>
<td>1.78</td>
<td>-1.00</td>
<td>1.21</td>
</tr>
<tr>
<td>You were able to control the dialogue in the desired way</td>
<td>-1.00</td>
<td>2.03</td>
<td>-1.00</td>
<td>1.38</td>
</tr>
<tr>
<td>The system reacted too slowly</td>
<td>1.00</td>
<td>1.96</td>
<td>-1.00</td>
<td>1.89</td>
</tr>
<tr>
<td>The system reacted in a friendly way</td>
<td>0.50</td>
<td>2.35</td>
<td>1.00</td>
<td>1.45</td>
</tr>
<tr>
<td>The system utterances were too long*</td>
<td>0.50</td>
<td>1.67</td>
<td>-1.00</td>
<td>1.63</td>
</tr>
<tr>
<td>You perceived the dialogue as natural</td>
<td>-2.00</td>
<td>1.41</td>
<td>-1.00</td>
<td>1.17</td>
</tr>
<tr>
<td>The course of the dialogue was clear</td>
<td>2.00</td>
<td>2.02</td>
<td>1.00</td>
<td>1.59</td>
</tr>
<tr>
<td>The dialogue was too long</td>
<td>-0.5</td>
<td>1.96</td>
<td>-1.00</td>
<td>1.41</td>
</tr>
<tr>
<td>I sometimes wondered if I was using the right word</td>
<td>2.00</td>
<td>1.72</td>
<td>1.00</td>
<td>2.27</td>
</tr>
<tr>
<td>The course of the dialogue was smooth</td>
<td>-0.50</td>
<td>1.75</td>
<td>-1.00</td>
<td>1.03</td>
</tr>
<tr>
<td>Overall, you are satisfied with the dialogue</td>
<td>0.00</td>
<td>2.13</td>
<td>0.00</td>
<td>1.86</td>
</tr>
<tr>
<td>You perceived the dialogue as pleasant</td>
<td>-0.50</td>
<td>1.99</td>
<td>0.00</td>
<td>1.50</td>
</tr>
<tr>
<td>During the dialogue, you felt relaxed</td>
<td>-1.00</td>
<td>2.00</td>
<td>0.00</td>
<td>1.53</td>
</tr>
</tbody>
</table>
Table A.4: The results of the questionnaire for the fourth dialogue. The rating is relying on a seven-point likert-scale ranging from the integer values -3 to +3. For each statement of the questionnaire the median and the standard deviation is provided. Where indicated with *, the differences are significant with $\alpha < 0.05$ (applying the Mann-Whitney U test (Mann & Whitney, 1947)).

<table>
<thead>
<tr>
<th>Statement</th>
<th>LUIS median</th>
<th>LUIS STD</th>
<th>Grammar median</th>
<th>Grammar STD</th>
</tr>
</thead>
<tbody>
<tr>
<td>You felt well understood by the system</td>
<td>-1.50</td>
<td>2.17</td>
<td>-1.00</td>
<td>1.85</td>
</tr>
<tr>
<td>You had to concentrate in order to understand what the system expected from you</td>
<td>0.50</td>
<td>1.75</td>
<td>-2.00</td>
<td>1.55</td>
</tr>
<tr>
<td>You knew at each point of the dialogue what the system expected from you</td>
<td>1.00</td>
<td>2.03</td>
<td>2.00</td>
<td>1.70</td>
</tr>
<tr>
<td>In your opinion, the system processed your specifications correctly</td>
<td>0.00</td>
<td>2.37</td>
<td>-1.00</td>
<td>1.88</td>
</tr>
<tr>
<td>The system’s behaviour was always as expected</td>
<td>0.00</td>
<td>2.29</td>
<td>0.00</td>
<td>1.89</td>
</tr>
<tr>
<td>The system makes a lot of mistakes</td>
<td>0.00</td>
<td>1.97</td>
<td>0.00</td>
<td>1.82</td>
</tr>
<tr>
<td>The system reacted in the same way as humans do</td>
<td>-2.00</td>
<td>1.59</td>
<td>-2.00</td>
<td>1.33</td>
</tr>
<tr>
<td>The system reacted flexibly</td>
<td>-1.00</td>
<td>1.92</td>
<td>-1.00</td>
<td>1.68</td>
</tr>
<tr>
<td>You were able to control the dialogue in the desired way</td>
<td>-0.50</td>
<td>1.90</td>
<td>-2.00</td>
<td>2.02</td>
</tr>
<tr>
<td>The system reacted too slowly</td>
<td>1.00</td>
<td>1.93</td>
<td>-1.00</td>
<td>1.90</td>
</tr>
<tr>
<td>The system reacted in a friendly way</td>
<td>0.00</td>
<td>2.24</td>
<td>1.00</td>
<td>1.88</td>
</tr>
<tr>
<td>The system utterances were too long</td>
<td>0.00</td>
<td>1.62</td>
<td>-2.00</td>
<td>1.82</td>
</tr>
<tr>
<td>You perceived the dialogue as natural</td>
<td>-1.50</td>
<td>1.40</td>
<td>-2.00</td>
<td>1.33</td>
</tr>
<tr>
<td>The course of the dialogue was clear</td>
<td>1.00</td>
<td>2.06</td>
<td>1.00</td>
<td>2.05</td>
</tr>
<tr>
<td>The dialogue was too long</td>
<td>-1.50</td>
<td>2.19</td>
<td>-1.00</td>
<td>1.19</td>
</tr>
<tr>
<td>I sometimes wondered if I was using the right word</td>
<td>1.00</td>
<td>1.93</td>
<td>0.00</td>
<td>2.34</td>
</tr>
<tr>
<td>The course of the dialogue was smooth</td>
<td>-2.00</td>
<td>2.05</td>
<td>-1.00</td>
<td>1.41</td>
</tr>
<tr>
<td>Overall, you are satisfied with the dialogue</td>
<td>-1.00</td>
<td>2.19</td>
<td>-1.00</td>
<td>2.02</td>
</tr>
<tr>
<td>You perceived the dialogue as pleasant</td>
<td>-1.00</td>
<td>2.09</td>
<td>-1.00</td>
<td>1.82</td>
</tr>
<tr>
<td>During the dialogue, you felt relaxed</td>
<td>-1.50</td>
<td>1.95</td>
<td>0.00</td>
<td>1.73</td>
</tr>
</tbody>
</table>
Welcome to the Evaluation of the WorkoutBot!

This Bot is part of my master’s thesis in Dialogue Systems. It aims to help you defining your personal Workout goal with using natural language via a Skype call.

The evaluation will need 20-30 minutes of your time. Before you start the evaluation, please make sure that you have checked following things

- Have you installed Skype on your Laptop/PC/Smartphone?
- Are you in a calm location?
- Make sure that your skype audio and speech input settings are well functioning. Therefore, you should use skype’s own audio device test.
- Add the Skype Bot to your Contacts with this link: WorkoutBot!

The evaluation itself starts with some personal questions. After that, you interact with the system in four different scenarios. Please follow carefully the given instructions in these scenarios!

If the system doesn’t answer immediately, please wait a few moments and start talking after the bot has finished. Furthermore, always let the survey window open while calling, so you can always look at your task.

The final part of this evaluation constitutes a final survey regarding your overall impression with the system. For participating in this evaluation you will receive an Amazon giftcard as compensation for your help. Thank you for participating and let the evaluation begin!

There are 22 questions in this survey

User Background

1 [Acronym]

Acronym

(Second letter of your mother’s first name, last digit of her birthday, third letter of your father’s first name, last digit of his birthmonth last digit of your own birthday) *

Please write your answer here:
2 [age] What is your age? *
Please write your answer here:

3 [gender] You are... *
Please choose only one of the following:
○ Female
○ Male

4 [profession] What is your profession/field of study? *
Please write your answer here:

5 [experience_tech] Please answer this questions, regarding your experience with spoken dialogue technology. *
Please choose the appropriate response for each item:

What is your level of expertise with speech dialogue systems? (1: novice, 5: expert)
○ ○ ○ ○ ○

Are you using a lot of technology in your job respectively in your spare time? (1: strong disagree, 5: strong agree)
○ ○ ○ ○ ○

What is your level of experience spoken language apps like Apple Siri, Microsoft Cortana, or Google Speech a lot? (1: novice, 5: expert)
○ ○ ○ ○ ○

6 [Device] Which device do you do this evaluation on? *
Please choose only one of the following:
○ Laptop
○ PC
○ Smartphone
○ Tablet
○ Other
### 7 [experience_fitness]

Please choose the appropriate response for each item:

<table>
<thead>
<tr>
<th>Question</th>
<th>inexperienced</th>
<th>very experienced</th>
</tr>
</thead>
<tbody>
<tr>
<td>How experienced are you in strength training?</td>
<td></td>
<td></td>
</tr>
<tr>
<td>How experienced are you in planning a workout?</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### 8 [fitnessstudio] Are you a member of a fitness studio?

Please choose **only one** of the following:

- Yes
- No

### 9 [NatLang] Native Language *

Please choose **only one** of the following:

- German
- English
- Other

### 10 [LangSkills]

**How would you rate your English skills?**

(1: very bad, 5: very good) *

Please choose **only one** of the following:

- 1
- 2
- 3
- 4
- 5
**Trainings Scenario**

**Training phase**

You are using the Skype WorkoutBot in order to define your Workout goal. Do the following interaction steps!

Goal items you have to specify:

1. Click on the call button in the upper right corner in the Skype application!
2. After the welcome message from the system, Say to the system "I want to gain stamina".
3. Answer the next question regarding the training dates, "I want to train three days a week".
4. After, that you are asked, how long you want to train per day. Say, "I want to train twenty minutes".
5. The last question is about your fitness level. Say, "I am a beginner".
6. After the system has presented your goal, it quits the call automatically.

11 [DialogueDone1] Have you called yet? *

Please choose all that apply:

- [ ] Yes
Scenario 1

You are using the Skype WorkoutBot in order to define your Workout goal. Now you don’t have to follow a specific dialogue.

Your task this time is to give the system one item per question/exchange. The order of the items is your choice, but try to choose other goals than before!

Goal items you have to specify:

1. overall goal (e.g., become fitter, gain stamina, lose/gain weight, gain strength,...).
2. Dates
3. Duration per day
4. Fitness level

12 [Dialogue Done 1b] Have you called yet? *

Please choose all that apply:

☐ Yes
**EvaluationFirstDial**

**13 [P.851/1]Please choose, how much you agree or disagree with the following statements: * **

Please choose the appropriate response for each item:

<table>
<thead>
<tr>
<th>Statement</th>
<th>strongly disagree</th>
<th>strongly agree</th>
</tr>
</thead>
<tbody>
<tr>
<td>You felt well understood by the system.</td>
<td>○</td>
<td>○</td>
</tr>
<tr>
<td>You had to concentrate in order to understand what the system expected from you.</td>
<td>○</td>
<td>○</td>
</tr>
<tr>
<td>You knew at each point of the dialogue what the system expected from you.</td>
<td>○</td>
<td>○</td>
</tr>
<tr>
<td>In your opinion, the system processed your specifications correctly.</td>
<td>○</td>
<td>○</td>
</tr>
<tr>
<td>The system's behaviour was always as expected.</td>
<td>○</td>
<td>○</td>
</tr>
<tr>
<td>The system makes a lot of mistakes.</td>
<td>○</td>
<td>○</td>
</tr>
<tr>
<td>The system reacted in the same way as humans do.</td>
<td>○</td>
<td>○</td>
</tr>
<tr>
<td>The system reacted flexibly.</td>
<td>○</td>
<td>○</td>
</tr>
<tr>
<td>You were able to control the dialogue in the desired way.</td>
<td>○</td>
<td>○</td>
</tr>
<tr>
<td>The system reacted too slowly.</td>
<td>○</td>
<td>○</td>
</tr>
<tr>
<td>The system reacted in a friendly way.</td>
<td>○</td>
<td>○</td>
</tr>
<tr>
<td>The system utterances were too long.</td>
<td>○</td>
<td>○</td>
</tr>
<tr>
<td>You perceived the dialogue as natural.</td>
<td>○</td>
<td>○</td>
</tr>
<tr>
<td>The course of the dialogue was clear.</td>
<td>○</td>
<td>○</td>
</tr>
<tr>
<td>The dialogue was too long.</td>
<td>○</td>
<td>○</td>
</tr>
<tr>
<td>I sometimes wondered if I was using the right word.</td>
<td>○</td>
<td>○</td>
</tr>
<tr>
<td>The course of the dialogue was smooth.</td>
<td>○</td>
<td>○</td>
</tr>
<tr>
<td>Overall, you are satisfied with the dialogue.</td>
<td>○</td>
<td>○</td>
</tr>
<tr>
<td>You perceived the dialogue as pleasant.</td>
<td>○</td>
<td>○</td>
</tr>
<tr>
<td>During the dialogue, you felt relaxed.</td>
<td>○</td>
<td>○</td>
</tr>
</tbody>
</table>
Scenario 2

Your task this time is to give the system two items (For example, Overall Goal and FitnessLevel) after the welcome message. Again, the order of the items is your choice, but try to choose other goals than before!

Goal items you have to specify:

1. overall goal (e.g., become fitter, gain stamina, lose/gain weight, gain strength,...)
2. Dates
3. Duration per day
4. Fitness level

14 [DialogueDone2]Have you called yet? *

Please choose all that apply:

[ ] Yes
**EvaluationSecondDial**

15 [P.851/2] Please choose, how much you agree or disagree with the following statements: *

Please choose the appropriate response for each item:

<table>
<thead>
<tr>
<th>Statement</th>
<th>Strongly Disagree</th>
<th>Strongly Agree</th>
</tr>
</thead>
<tbody>
<tr>
<td>You felt well understood by the system.</td>
<td>☐</td>
<td>☐</td>
</tr>
<tr>
<td>You had to concentrate in order to understand what the system expected from you.</td>
<td>☐</td>
<td>☐</td>
</tr>
<tr>
<td>You knew at each point of the dialogue what the system expected from you.</td>
<td>☐</td>
<td>☐</td>
</tr>
<tr>
<td>In your opinion, the system processed your specifications correctly.</td>
<td>☐</td>
<td>☐</td>
</tr>
<tr>
<td>The system's behaviour was always as expected.</td>
<td>☐</td>
<td>☐</td>
</tr>
<tr>
<td>The system makes a lot of mistakes.</td>
<td>☐</td>
<td>☐</td>
</tr>
<tr>
<td>The system reacted in the same way as humans do.</td>
<td>☐</td>
<td>☐</td>
</tr>
<tr>
<td>The system reacted flexibly.</td>
<td>☐</td>
<td>☐</td>
</tr>
<tr>
<td>You were able to control the dialogue in the desired way.</td>
<td>☐</td>
<td>☐</td>
</tr>
<tr>
<td>The system reacted too slowly.</td>
<td>☐</td>
<td>☐</td>
</tr>
<tr>
<td>The system reacted in a friendly way.</td>
<td>☐</td>
<td>☐</td>
</tr>
<tr>
<td>The system utterances were too long.</td>
<td>☐</td>
<td>☐</td>
</tr>
<tr>
<td>You perceived the dialogue as natural.</td>
<td>☐</td>
<td>☐</td>
</tr>
<tr>
<td>The course of the dialogue was clear.</td>
<td>☐</td>
<td>☐</td>
</tr>
<tr>
<td>The dialogue was too long.</td>
<td>☐</td>
<td>☐</td>
</tr>
<tr>
<td>I sometimes wondered if I was using the right word.</td>
<td>☐</td>
<td>☐</td>
</tr>
<tr>
<td>The course of the dialogue was smooth.</td>
<td>☐</td>
<td>☐</td>
</tr>
<tr>
<td>Overall, you are satisfied with the dialogue.</td>
<td>☐</td>
<td>☐</td>
</tr>
<tr>
<td>You perceived the dialogue as pleasant.</td>
<td>☐</td>
<td>☐</td>
</tr>
<tr>
<td>During the dialogue, you felt relaxed.</td>
<td>☐</td>
<td>☐</td>
</tr>
</tbody>
</table>
Scenario 3

Your task this time is to give the system three items (E.g., Overall Goal, FitnessLevel, and Dates) after the welcome message. Again, the order of the items is your choice, but try to choose other goals than before!

Goal items you have to specify:

1. overall goal (e.g., become fitter, gain stamina, lose/gain weight, gain strength,...)
2. Dates
3. Duration per day
4. Fitness level

16 [DialogueDone3] Have you called yet? *

Please choose all that apply:

☐ Yes
17 [P.851/3] Please choose, how much you agree or disagree with the following statements: *

<table>
<thead>
<tr>
<th>Statement</th>
<th>strongly disagree</th>
<th>strongly agree</th>
</tr>
</thead>
<tbody>
<tr>
<td>You felt well understood by the system.</td>
<td>☐</td>
<td>☘</td>
</tr>
<tr>
<td>You had to concentrate in order to understand what the system expected</td>
<td>☐</td>
<td>☘</td>
</tr>
<tr>
<td>You knew at each point of the dialogue what the system expected from you.</td>
<td>☐</td>
<td>☘</td>
</tr>
<tr>
<td>In your opinion, the system processed your specifications correctly.</td>
<td>☐</td>
<td>☘</td>
</tr>
<tr>
<td>The system's behaviour was always as expected.</td>
<td>☐</td>
<td>☘</td>
</tr>
<tr>
<td>The system makes a lot of mistakes.</td>
<td>☐</td>
<td>☘</td>
</tr>
<tr>
<td>The system reacted in the same way as humans do.</td>
<td>☐</td>
<td>☘</td>
</tr>
<tr>
<td>The system reacted flexibly.</td>
<td>☐</td>
<td>☘</td>
</tr>
<tr>
<td>You were able to control the dialogue in the desired way.</td>
<td>☐</td>
<td>☘</td>
</tr>
<tr>
<td>The system reacted too slowly.</td>
<td>☐</td>
<td>☘</td>
</tr>
<tr>
<td>The system reacted in a friendly way.</td>
<td>☐</td>
<td>☘</td>
</tr>
<tr>
<td>The system utterances were too long.</td>
<td>☐</td>
<td>☘</td>
</tr>
<tr>
<td>You perceived the dialogue as natural</td>
<td>☐</td>
<td>☘</td>
</tr>
<tr>
<td>The course of the dialogue was clear.</td>
<td>☐</td>
<td>☘</td>
</tr>
<tr>
<td>The dialogue was too long.</td>
<td>☐</td>
<td>☘</td>
</tr>
<tr>
<td>I sometimes wondered if I was using the right word.</td>
<td>☐</td>
<td>☘</td>
</tr>
<tr>
<td>The course of the dialogue was smooth.</td>
<td>☐</td>
<td>☘</td>
</tr>
<tr>
<td>Overall, you are satisfied with the dialogue.</td>
<td>☐</td>
<td>☘</td>
</tr>
<tr>
<td>You perceived the dialogue as pleasant.</td>
<td>☐</td>
<td>☘</td>
</tr>
<tr>
<td>During the dialogue, you felt relaxed.</td>
<td>☐</td>
<td>☘</td>
</tr>
</tbody>
</table>
Scenario 4

Your task this time is to interact freely: You don`t have any restrictions and can give as many items as you like. Again, the order of the items is your choice, but try to choose other goals than before!

Goal items you have to specify:

1. overall goal (e.g., become fitter, gain stamina, lose/gain weight, gain strength,...)
2. Dates
3. Duration per day
4. Fitness level

18 [DialogueDone4]Have you called yet? *

Please choose all that apply:

- [ ] Yes
**EvaluationFourthDial**

19 [P.851/4] Please choose, how much you agree or disagree with the following statements: *

Please choose the appropriate response for each item:

<table>
<thead>
<tr>
<th>Statement</th>
<th>strongly disagree</th>
<th>strongly agree</th>
</tr>
</thead>
<tbody>
<tr>
<td>You felt well understood by the system.</td>
<td>○</td>
<td>○</td>
</tr>
<tr>
<td>You had to concentrate in order to understand what the system expected from you.</td>
<td>○</td>
<td>○</td>
</tr>
<tr>
<td>You knew at each point of the dialogue what the system expected from you.</td>
<td>○</td>
<td>○</td>
</tr>
<tr>
<td>In your opinion, the system processed your specifications correctly.</td>
<td>○</td>
<td>○</td>
</tr>
<tr>
<td>The system's behaviour was always as expected.</td>
<td>○</td>
<td>○</td>
</tr>
<tr>
<td>The system makes a lot of mistakes.</td>
<td>○</td>
<td>○</td>
</tr>
<tr>
<td>The system reacted in the same way as humans do.</td>
<td>○</td>
<td>○</td>
</tr>
<tr>
<td>The system reacted flexibly.</td>
<td>○</td>
<td>○</td>
</tr>
<tr>
<td>You were able to control the dialogue in the desired way.</td>
<td>○</td>
<td>○</td>
</tr>
<tr>
<td>The system reacted too slowly.</td>
<td>○</td>
<td>○</td>
</tr>
<tr>
<td>The system reacted in a friendly way.</td>
<td>○</td>
<td>○</td>
</tr>
<tr>
<td>The system utterances were too long.</td>
<td>○</td>
<td>○</td>
</tr>
<tr>
<td>You perceived the dialogue as natural.</td>
<td>○</td>
<td>○</td>
</tr>
<tr>
<td>The course of the dialogue was clear.</td>
<td>○</td>
<td>○</td>
</tr>
<tr>
<td>The dialogue was too long.</td>
<td>○</td>
<td>○</td>
</tr>
<tr>
<td>I sometimes wondered if I was using the right word.</td>
<td>○</td>
<td>○</td>
</tr>
<tr>
<td>The course of the dialogue was smooth.</td>
<td>○</td>
<td>○</td>
</tr>
<tr>
<td>Overall, you are satisfied with the dialogue.</td>
<td>○</td>
<td>○</td>
</tr>
<tr>
<td>You perceived the dialogue as pleasant.</td>
<td>○</td>
<td>○</td>
</tr>
<tr>
<td>During the dialogue, you felt relaxed.</td>
<td>○</td>
<td>○</td>
</tr>
</tbody>
</table>
FinalSurvey

This the final survey, regarding your impression with the system.

<table>
<thead>
<tr>
<th><strong>20 [Final] Please choose, how much you agree or disagree with the following statements:</strong> *</th>
<th><strong>strongly disagree</strong></th>
<th><strong>strongly agree</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>Your overall impression with the system was excellent.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>The system's way of expression was unclear.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>The system reacted politely.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>You would have expected more help from the system.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Misunderstandings could be cleared easily.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>The system controlled the flow of the dialogue.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>You were able to handle the system without any problems.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Regarding the dialogues, you are unimpressed</td>
<td></td>
<td></td>
</tr>
<tr>
<td>You enjoyed the dialogues.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>You feel adequately informed about the system's possibilities.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>The skype calls with the system were worthwhile.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>You rate the system as reliable.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>You prefer to use another source of information.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>You perceived this possibility for obtaining information as helpful.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>The handling of the system was complicated.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>You prefer a human operator.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>In the future, you would use the system again.</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Please delete now the Bot contact from your Skype contact list and please enter your email address and your name in the fields below in order to receive the 5€ Amazon gift-card!

21 [Get_Email] Your Email address *

Please write your answer here:

22 [Get_Name] Please enter your name *

Please write your answer here:
Thank you for participating!

01.01.1970 – 01:00

Submit your survey.
Thank you for completing this survey.
References


References


References


Erklärung

Ich erkläre, dass ich die Arbeit selbständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel verwendet habe.

Ulm, den .................................................................

Matthias Kraus