Discovering Available Drinks Through Natural, Robot-Led Human-Robot Interaction Between a Waiter and a Bartender

Tirza Soute
Universiteit van Amsterdam
Informatics Institute
BSc Artificial Intelligence Thesis

Discovering Available Drinks Through Natural, Robot-Led, Human-Robot Interaction Between a Waiter and a Bartender

Tirza F.E. Soute
10761977

Bachelor thesis
Credits: 18 EC

Bachelor Opleiding Kunstmatige Intelligentie

University of Amsterdam
Faculty of Science
Science Park 904
1098 XH Amsterdam

Supervisor
dhr. dr. A. Visser

Informatics Institute
Faculty of Science
University of Amsterdam
Science Park 904
1098 XH Amsterdam

June 30th, 2017

Abstract

This research focuses on natural, robot-led, human-robot interaction that enables a robot to discover what drinks a barman can prepare through continuous application of speech recognition, understanding and generation. Speech was recognised using Google Cloud's speech to text API, understood by matching either the object or main verb of a sentence against a list of key words and, finally, generated using templates with variable parts. The difficulty lies in the large quantity of key words, as they are based on the properties of the ordered drinks. The results show that having the aforementioned interaction works well to some extent, i.e. the naturalness of the interaction was ranked 5.5 on average. Furthermore, the obtained precision when identifying the unavailable drinks was 0.625 and the obtained recall was 1, resulting in an F_1 measure of 0.769.

Contents

1 Introduction

Each year, the RoboCup[1] attempts to promote robotics and artificial intelligence research by organising a competition that offers difficult challenges. This year, it will be held in Nagoya, Japan, and will consist of several leagues, of which one of them is the RoboCup@Home league. The objective of this league is to develop service and assistive robot technology for future personal domestic applications. Within the RoboCup@Home league, there are various platform leagues, including the Social Standard Platform League (Visser, 2017). This league focuses on human-robot interaction, natural language processing, people detection and recognition, reactive behaviours and safe outdoor navigation and mapping. All of the techniques that result from this can be applied to many different domains.

In one of the challenges of the RoboCup@Home Social Standard Platform League, a Softbank Robotics Pepper robot[2], which is depicted in Figure 1, is a waiter at a cocktail party. The robot is obliged to complete several tasks, such as entering the arena, responding to getting called by customers, placing the orders that were made by the customers at the bar, realising what ordered drinks are unavailable and providing anyone whose drink is unavailable with three alternative drinks. All of these tasks show the necessity of the aforementioned focuses of the league.

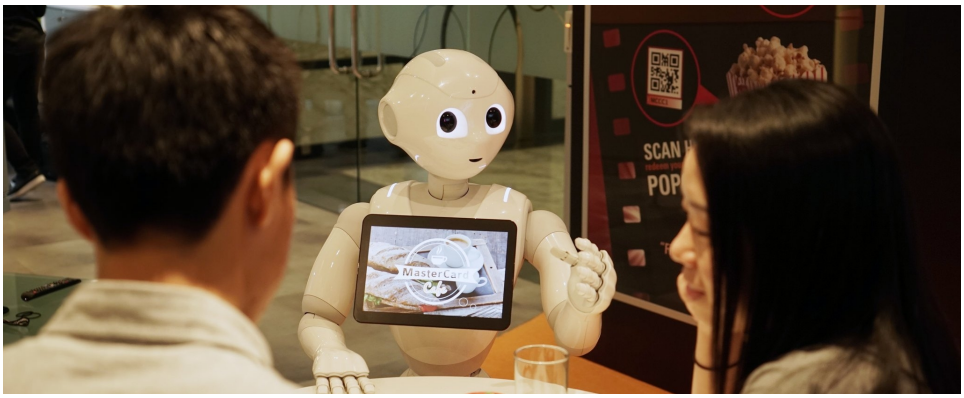

Figure 1: A Pepper robot operating as a waiter at a Pizza Hut restaurant in Singapore[3].

This research will focus on the task of realising what ordered drinks are unavailable, which will be achieved by conversing with the bartender and continuously applying speech recognition, understanding and generation. The aim of this research is to discover how to have a natural conversation between a robot and a human and apply this to the scenario of a cocktail party. Thus, the research ques-

[1] http://www.robocup.org/

[2] https://www.ald.softbankrobotics.com/en/cool-robots/pepper

tion that follows from this is: 'To what extent is it possible for a Pepper robot to have a natural conversation with a bartender to discover what drinks it can offer?'. The null hypothesis to this research question is that it is not possible for a Pepper robot to have a natural conversation with a bartender. Subsequently, several steps will need to be taken before the research question can be answered. Firstly, the robot needs to know what drinks there are in general and what their properties are before it is able to flag which are available and unavailable. Secondly, in order to have the conversation, it is crucial that the robot is capable of understanding what the bartender says, and, lastly, the robot needs to be able to generate language so that it direct the conversation by asking the bartender questions. The sub-questions that follow from these steps are:

1. 'How can the Pepper robot track what drinks the bartender can make?'
2. 'How can the Pepper robot understand what the bartender says?'
3. 'How can the Pepper robot generate language to direct a conversation with the bartender?'

Although this research focuses on the domain of drinks at cocktail parties, it could also be expanded to different domains, such as hospitals, banks or hotels. To summarise, the importance of this research is that if it is possible to have a good discussion about drinks, then it is possible to have a good discussion about anything and this knowledge can be applied to any robotic context. For example, robots, e.g. the Pepper robot, are often used in a service context, such as in mental health care for elderly people, to assist, guide, provide therapy, educate and enable communication (Shibata, 2011). In such a context, human-robot interaction is especially important, because it ensures that people are more prone to trust the robot and that they thus have a better experience. Furthermore, although there have been many developments in fields of artificial intelligence, such as natural language processing and human-robot interaction, there has not been developed a robot that is intelligent with whom people can interact long-term yet. For example, it is uninteresting for people to interact with a robot that behaves based on hand-coded rules, which demonstrates the difficulty of developing a robot that excels at human-robot interaction (Taniguchi et al., 2016).

One of the interesting aspects of this research is that it is the robot who leads the conversation, while this is normally done by the person with whom the robot interacts (Mavridis, 2015). The difficulty of the domain in this scenario lies in the amount of properties of drinks, e.g. the large amount of ingredients. Furthermore, the majority of related research applies natural language understanding by matching words against either regular expression or key words, which are often represented in a knowledge base (Glas et al., 2016; Avilés et al., 2010; Lemaignan et al., 2012; Padmakumar et al., 2017). Since there is a vast amount of drink properties, the matching has to be done against an extremely large amount of key words.

Moreover, natural language is often generated using hand-scripted sentences (Glas et al., 2016).

This thesis is structured as follows: firstly, in section 2, the theoretical foundation of this research will be given, followed by a description of related research in section 3. Furthermore, the approach that was taken will be explained in section 4, after which the method that was used to evaluate this approach will be described in section 5. Moreover, the results that were obtained will be shown in section 6 and, finally, the conclusion, discussion and suggestions for future work will be given in section 7.

2 Theoretical foundation

Before there will be given a description of related research and the approach that was taken for this research will be explained, a theoretical foundation will be provided in this section. Firstly, a dialogue model is a model that is used to structure a conversation and could be described as a directed graph, where nodes represent situations and edges define expectation-action pairs so that the corresponding action is performed if an expectation is met. An example of a dialogue model can be seen in Figure 2, which is a dialogue model based on a finite state machine. This model demonstrates that from the start symbol s there are three different sub-dialogues that can be chosen, which all lead to the final sub-dialogue Rn and end in the finishing state fs. The importance of a dialogue model is that it describes a set of expected situations and defines the interaction protocol, therefore regulating the human-robot interaction.

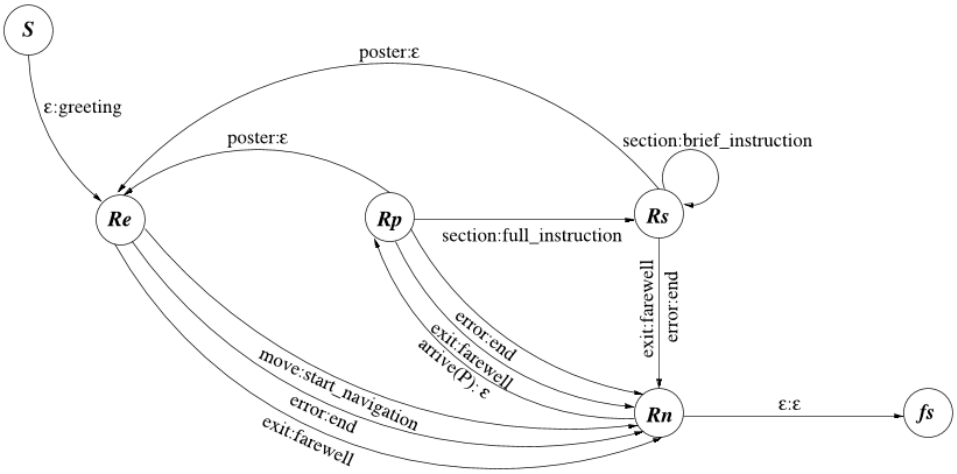

Figure 2: An example of a dialogue model that was used by Avilés et al. (2010).

Secondly, beliefs, desires and intentions are often used to create decision-making

models of artificial agents (Georgeff et al., 1999; Lee and Son, 2008). Belief represents whatever an agent holds about its environment and itself (O'Hare and Jennings, 1996), while intention and desire both represent a state that the agent wishes to produce. However, the difference is that only intentions cause an agent to act in order to fulfil its wish. For this research, each beliefs, desires and intentions are important, because the knowledge that the robot obtains from the bartender are its beliefs and, furthermore, the robot's intentions cause the robot to ask the bartender questions, while the robot's desires represent the other questions that it wants to ask the bartender.

Moreover, to understand natural language, there is often made use of a lexicon, parser and grammar, which decompose sentences into their internal representation. The structure of these components is that the parser decomposes the sentence into lexical entries using the provided grammar, which defines the rules of the decomposition. There are many different grammars, e.g. a Combinatory Categorial Grammar consists of, firstly, a categorial lexicon that associates each word with a syntactic and semantic category and, secondly, the corresponding combinatory rules that allow functions and arguments to be combined (Jurafsky, 2000, p.451). One of the two types of categories are 'arguments', which have simple categories, while the second type of category, 'functors', are verbs and determiners. For instance, a determiner could be seen as a function that applies to a noun (N) on its right to produce a noun phrase (NP). Such complex categories are built using the X/Y and $X\backslash Y$ operators, meaning that X/Y is a function from Y to X, i.e. something that combines with a Y on its right to produce an X such that a determiner receives the category NP/N. Accordingly, $X\backslash Y$ is a function from Y to X such that something combines with a Y on its left to produce an X. Furthermore, an example of a parsing algorithm is the CYK algorithm, which is a bottom-up parsing algorithm for context-free grammars that uses a dynamic programming table to efficiently store its intermediate results. It determines whether it is possible to generate a string from a provided context-free grammar and does so if it is (Jurafsky, 2000, p.449-451).

Finally, one of the standing difficulties in natural language understanding is that robots are often in need of explicit world knowledge in order to understand what the 'underlying meaning' of something is. This is referred to as 'symbol grounding', which is related to the problem of how symbols i.e. words, get their meanings and thus what the meaning of the word itself is.

3 Related research

Related research demonstrated task selection according to a dialogue model (Avilés et al., 2010). The speech recognition module was based on general acoustic-phonetic models and the interpretation module depended on regular expressions,

namely by comparing a recognised sentence with a set of equivalent regular expressions, which were defined for each expectation. If no match was found, the user was requested for another attempt. The researchers asserted that this approach showed promising results in human-robot interaction, although they did indicate that the naturalness of the conversation should be improved. As was explained before, this research will focus on having a natural conversation.

Lemaignan et al. (2012) grounded natural language using Google's speech recognition API, which converts speech into text, and by grammatically structuring this text using a heuristics-based parser. The resulting atoms were resolved with the help of a knowledge base to ground concepts, such as objects. This knowledge base was constructed by showing the robot certain objects or by pointing at them, which requires extensive training time. Therefore, this approach is not applicable to this research. However, this approach could be taken in future research. Moreover, the content of the resolved phrase was analysed to determine the intention of an utterance (Lemaignan et al., 2017). The intention could either be to inform, to inquire or to express desire. During the final step, the statement was built into a Resource Description Framework (RDF) statement. The researchers claimed that this approach made human-robot interaction more natural.

Later work by Petrick and Foster (2013) demonstrated the use of an iCat as a bartender. In order to recognise speech, there was made use of a Microsoft Kinect and the associated Microsoft Speech API, which provides a list of speech recognition hypotheses and associated confidence scores. Firstly, recognised speech was processed to extract the underlying meaning and, secondly, the recognised hypotheses were parsed using a bi-directional, bilingual OpenCCG grammar to create symbolic representations of the speech. To illustrate, a bi-directional grammar is a grammar that both includes language understanding and generation, and, furthermore, OpenCCG is an open source natural language processing library, which provides parsing and realisation services based on Combinatory Categorial Grammars [4]. Similarly to the research by Avilés et al. (2010), which made use of expected regular expressions, the speech recognition grammar covered expected user utterances in a bartending scenario, therefore constraining the recognition task and producing more reliable results.

Glas et al. (2016) developed a human-like android, Erica, which is depicted in Figure 3, that is capable of conversational interaction. The speech synthesis module determined the intonation of a sentence by grammar. However, manual specification of pitch, speed and intensity was possible. The speech recognition module assumed wireless, handheld microphones and employed the Deep Neural Network version of an open-source large-vocabulary speech recognition system. Dialogue was controlled using a state-transition model, which compared speech recognition

[4] `http://openccg.sourceforge.net/`

Figure 3: The human-like android Erica[5].

results with a list of keywords using statistical matching. If a match occured, an internal state machine was updated and an utterance was generated for the android to speak. When no match was found, the android asked for clarification. The utterance content and transition rules were scripted by hand and the researchers claimed that this mechanism, although simple, could have multiple-turn conversations and incorporate history, which, they asserted, was enough to successfully address people's questions and ask questions back.

Before a robot can understand natural language, it needs to know when someone is speaking and for how long so that it can determine when to stop listening. To achieve this, Perera et al. (2017) used NAOqi[6], which is the API of the Pepper robot. This API offers the option to retrieve the signal energy of the robot's microphones and to detect when any of the robot's sensors are touched. The average signal energy was measured over one second using moving windows of 0.2 seconds when the robot's hand was touched. Firstly, a baseline was measured shortly to determine the average noise level in the environment and, secondly, the average signal energy over one second was measured until the baseline was reached again. This method to record the sound was used in combination with IBM Watson's Speech to Text service, which returns a transcription of the audio, to recognise speech.

The integration of learning a dialogue strategy using reinforcement learning and a semantic parser for robust natural language understanding, using only natural dialogue interaction for supervision was demonstrated by Padmakumar et al. (2017). Semantically parsed sentences were obtained through probabilistic CKY-parsing with a Combinatory Categorical Grammar and meanings associated with lexical entries. In addition, a Partially Observable Markov Decision Process modelled dialogue and learned a policy and, finally, The Hidden State Information model tracked the belief state as the dialogue progressed. Experiments demonstrated that learning both a dialogue strategy using reinforcement learning and a semantic parser for robust natural language understanding improved dialogue performance over learning either of these components alone. It was not possible to apply this

[6] `http://doc.aldebaran.com/2-1/naoqi`

approach to this research, because it requires a corpus, which was not available.

4 Approach

This section will explain the approach that was taken to answer the sub-questions of the research question and thus the research question. Firstly, the method that was used to track what the bartender can make will be explained in section 4.1. Secondly, the approach that was taken to understand natural language will be described in section 4.2 and, furthermore, the used method to update the list of available drinks will be described in section 4.3. Finally, the method that was used to generate natural language will be explained in section 4.4.

4.1 Database

It is crucial that the robot knows what drinks exist and what some of their properties are in order to flag what drinks the bartender can and cannot make. Therefore, a database of 3634 drinks was created using the Absolut Drinks Database[7], which is a free Application Programming Interface (API) with unlimited calls. The web pages of the API are formatted in JSON and provide information about drinks and their properties.

{cosmopolitan: [cosmopolitan,
pink,
alcoholic,
non-carbonated,
cold,
ice cubes, 2 parts absolut citron, 1 part lime juice, (...),
fresh, fruity, sweet,
citrus press, freezer, jigger, strainer, twist knife, boston shaker,
fill, add, shake, strain, chill, garnish]}

Figure 4: An example of the content of the database, which shows that the name of the drink was the key in the database and that the list of properties was its value. The list consists of: [name, colour, is_alcoholic, is_carbonated, is_hot, ingredients, tastes, tools, actions]. To decrease the line width in the example, long properties were shortened using '(...)'.

There are many properties in the Absolut Drinks Database, though not all of them are important. Therefore, the most relevant properties were selected, namely the name of the drink, its colour, whether it is alcoholic, whether it is carbonated or not, whether it is hot or cold, its ingredients, its tastes, the required tools, and the actions the bartender has to do in order to make it. This information was saved in

[7] `http://addb.absolutdrinks.com`

a list and added to a Python dictionary, which held the drink name as the key and the list of the properties of the drink as its value, such that {drink name: [name, colour, is_alcoholic, is_carbonated, is_hot, ingredients, tastes, tools, actions]}. This dictionary was saved in a Pickle file and formed the database, of which an example entry is given in Figure 4.

4.2 Natural language understanding

This section will describe the approach that was taken to understand natural language, starting with how to obtain a written format of spoken language in section 4.2.1 and continuing with how to understand this written format in section 4.2.2.

4.2.1 Speech recognition

Understanding natural language requires a written format of the sentence that was spoken, which was obtained using the Google Cloud speech to text API[8], which takes an audio file as input and returns a transcription of the spoken sentences in the audio file as output using CLDNN-HMM, which combines a Convolutional Neural Network (CNN), a Deep Neural Network (DNN), Long Short-Term Memory (LSTM) and Hidden Markov Models (HMM) (Chan et al., 2016). This combination was made, because CNNs and LSTMs have shown improvements over DNNs on speech recognition tasks and because CNNs, LSTMs and DNNs are complementary in modelling capabilities (Sainath et al., 2015). In particular, CNNs perform well at reducing frequency variations, while LSTMs are good at temporal modelling and DNNs are appropriate for mapping features to a more separable space. Firstly, the input signal was passed through two convolutional layers to reduce frequency variance, of which the first layer had a 9x9 frequency-time filter and the second layer a 4x3 filter. On the first layer, there was done non-overlapping max pooling. The second, i.e. last, layer of the CNN was large and thus a linear layer was added to reduce the feature dimension before it was passed to two LSTM layers, which modelled the signal in time. The output of the LSTM was passed to fully connected DNN layers, which produced a higher-order feature representation that was more easily separable into different classes. Furthermore, HMMs were used to model time series data as they are often used in speech recognition systems, e.g. the Google Cloud text to speech API (Ghahramani, 2001). They represent probability distributions over sequences of observations.

Before selecting Google Cloud's speech to text API, IBM Watson's API[9] was analysed as well. However, this returned transcriptions significantly slower than Google's API, e.g. where IBM took 7.18 seconds to transcribe a sentence, Google took 2.87 seconds to transcribe it. Since the research focuses on having a natural

[8] https://cloud.google.com/speech/

[9] https://www.ibm.com/watson/developercloud/doc/speech-to-text/index.html

conversation, it is desirable to receive a transcribed sentence as quickly as possible. Furthermore, the results of Google's API were more accurate than IBM's API and it was thus preferable to use Google's speech to text API.

Naturally, the robot needs to know when the bartender is speaking and, therefore, the bartender can indicate that he wants to speak by touching and holding the back of the robot's left hand, which is similar to the method that was used by Perera et al. (2017), as they also used the robot's hand sensor to determine when to start listening. However, the difference is that Perera et al. (2017) used signal energy to determine when to stop listening, while the robot in this research stopped listening once the bartender let go of the robot's hand. In order to indicate to the bartender when the robot was listening, the blue LEDs in its eyes would rotate. Furthermore, the audio file that was recorded during that time was automatically saved on the robot and thus needed to be sent to the program in order for it to be transcribed by the Google Cloud speech to text API. Copying the audio file required the password of the robot to be typed into the terminal, which would decrease the naturalness of the conversation. Therefore, this was done automatically using the Python library `pexpect`, which waits for the password to be requested and then enters it. The outline of the approach that was taken to obtain a written format of a spoken sentence is given in Figure 5.

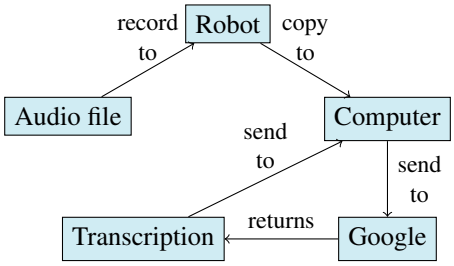

Figure 5: The outline of the method that was used to obtain a written format of a spoken sentence.

If a value error occurred during the transcription of the spoken sentence, the speech to text API did not hear any words. In this case, the robot communicated that it did not hear what the bartender said and the bartender had the opportunity to answer the same question again. Once a transcription of the spoken sentence had been obtained, the robot started trying to understand it. Moreover, it is possible that the bartender did not hear the robot and he could thus ask the robot to repeat the question.

4.2.2 Speech understanding

There are several steps that were taken in order to understand a written sentence. Firstly, the sentence was parsed using the Stanford Dependency Parser[10], and the main verb was extracted using the parsed sentence and NLTK's[11] `pos_tag` method (Bird, 2006), which processes a sequence of words and attaches a part-of-speech tag to each word. If the tag starts with 'VB', it is a verb. This method was thus used to extract the verbs of a sentence. However, since the main verb was the only required verb for the program, each extracted verb was analysed to detect if it was auxiliary using the parsed sentence. If it was auxiliary, the parser labelled the verb as 'aux' and it was thus not added to the list of verbs. Furthermore, the `pos_tag` function occasionally incorrectly tagged words as verbs and, therefore, the parsed sentence was used to filter out any incorrectly tagged verbs by analysing the parsed element in which the tagged verb occurs. If 'VB' was not in the parsed element in which the tagged verb occurred, then it was deleted from the list of verbs. Finally, some verbs consist of two parts, e.g. 'top up', and the preposition would be parsed as 'compound:prt'. Therefore, the parsed sentence was analysed for such occurrences, in which case it was added to the verb in the list.

During the second step, the type of the given answer was analysed. The types were categorised into 'empty' and 'non-empty' answers: an 'empty' answer is an answer such as "Yes" or "No, I don't", while a 'non-empty' answer is an answer such as "I don't have any lemons". The module used the main verb, object and, optionally, the negation to understand a written sentence. If the bartender gave an empty answer, then the main verb and object of the question were used to understand the sentence instead of those of the answer. However, the negation of the answer was always used.

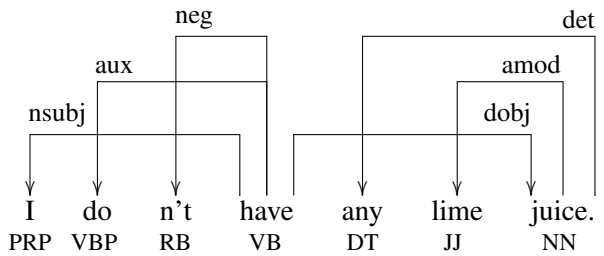

Figure 6: A visualisation of the output of the Stanford Dependency Parser.

The third step was to analyse the sentence itself, which required identifying the object of the sentence and, optionally, the negation. The parser labelled the object as 'dobj' and the negation as 'neg', as can be seen in Figure 6. Each found object was added to a list of objects and, similarly, if a negation was detected then 'not'

[10] `https://nlp.stanford.edu/software/lex-parser.shtml`
[11] `http://www.nltk.org/`

was added to a list of negations. However, if no negation occurred in the written sentence, 'None' was added to the list of negations instead. The parser only labelled the head of the object as the object, such that if the object was 'boston shaker', then the parser would identify 'shaker' as the object. Therefore, this was corrected by inspecting whether any words were linked to the object, as they would have been labelled as 'compound' or 'amod', e.g. such as in Figure 6. The term 'amod' represented an adjectival modifier, which is an adjectival phrase that serves to modify the meaning of the noun phrase and 'compound' was a compound modifier, which is any noun that serves to modify the head noun (De Marneffe and Manning, 2008). A noun phrase is a phrase that has a noun as its head word or that performs the grammatical function of a noun, such that, for example, in the sentence 'The dog has a bone.', the phrases 'the dog' and 'a bone' are noun phrases.

It is possible that the bartender gave an answer such as 'I have lemons and peaches', in which case the sentence would be parsed as having 'two objects', namely 'lemons' and 'peaches'. However, there would only be one main verb: 'have', which would therefore have to be copied so that the length of the list of verbs was equal to the length of the list of objects. Due to the fact that there were three different lists, namely the verbs, objects and negations, the length of the longest list was obtained and the elements of the other lists were repeated until they were the same length.

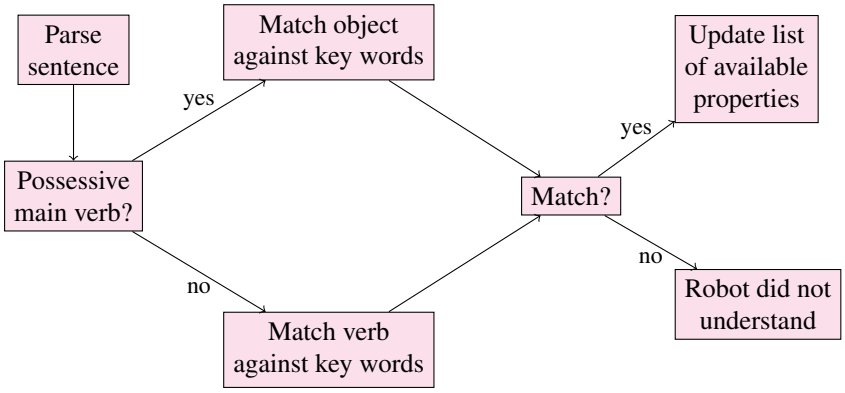

Figure 7: The outline of the approach that was taken to understand a sentence that is written in text format.

Using the obtained features, namely the verb, object and negation, the program could understand the sentence by matching either the main verb or object against a list of key words, which depended on whether the main verb was 'possessive', e.g. if it was a verb such as 'have' or 'own'. If it was, the object was matched against the key words, while the main verb was matched if it was not. If no match was found, the robot did not understand what the bartender said. However, if a match was found, then the robot could update the list of available drink properties or remove a drink from the list of available drinks. The outline of the approach to

understand a written sentence is illustrated in Figure 7.

4.3 Updating the available drinks

This section will explain how the list of available drinks was updated. Firstly, in section 4.3.1, the method that was used to generate key words will be explained and, secondly, the approach that was taken to generate the 'unflagged properties lists' will be described in section 4.3.2.

4.3.1 Key word generation

Using the database of drinks, the key words were generated by creating a dictionary of dictionaries. For each drink in the database, its properties were used to generate a list of key words using NLTK's WordNet Interface[12], which is a large lexical database of English where words are grouped into synonyms (Miller, 1995). The key words thus consisted of the drink properties and their synonyms. However, it was not necessary to generate synonyms for all of the properties, e.g. the bartender would likely not use a different word than 'green' to describe a green drink. Therefore, no synonyms were generated for the colour of the drink and whether the drink was alcoholic, carbonated and hot. Accordingly, synonyms were generated for the ingredients of the drink, its tastes, the necessary tools and the actions the bartender needs to do to make the drink. For each drink in the database, a dictionary of synonyms was created such that the key was the synonym and its value was the word the synonym was generated from, i.e. {synonym: word the synonym was generated from}. Furthermore, there was always an entry where a word referred back to itself such that {word: word}.

Due to the nature of a Python dictionary, keys could be overwritten if they occur more than once. For example, according to WordNet, 'stir' is a synonym of 'shake', but 'shake' is an action that often occurs in the list of properties. Therefore, the synonym was not saved in the dictionary if there was already an entry in the dictionary of the word that the synonym was generated from, e.g. if {stir: stir} was already in the database, then {stir: shake} was not added. After the synonym dictionary was created, it was saved in another dictionary that held the name of the drink as key and the created synonym dictionary as value. This final, large dictionary was saved in a Pickle file and used to generate the key words at the start of the program by merging the synonym dictionaries of the ordered drinks. Since the synonyms were of a dictionary type, any object or verb that one was trying to match could be retrieved using Python's `get` function.

[12] `http://www.nltk.org/howto/wordnet.html`

4.3.2 Unflagged properties list generation

In addition to the key words, another list was generated using the database to flag what properties the robot had been informed of by the bartender. This list will be referred to as the 'unflagged properties list' and was constructed similarly to the key words, namely by generating the list for every single drink in the database and then saving it in a dictionary in a Pickle file using the name of the drink as key and the unflagged property list as value, such that {drink name: [unflagged properties list]}. As a result of this, each unflagged properties list could easily be obtained at the start of the program using Python's `get` function and the list of ordered drinks. Each unflagged properties list was constructed as follows: firstly, the properties of a drink from the database were obtained. Secondly, since the properties that indicated the drink's colour, whether it was alcoholic, carbonated and hot always consisted of a single word, they could therefore be appended to the unflagged properties list without modification using the following format: [property_value: None // property_type_name]. To illustrate, the property 'green' of the property type 'colour' would look as follows: [green: None // colour].

However, the properties ingredients, tastes, tools and actions did require modification. Firstly, the ingredients were stripped of their quantities such that only the actual ingredients were left. Secondly, since the aforementioned properties were joined by commas into a single string such that the ingredients were, for example, 'ice cubes, 2 parts absolut vodka, 3 parts tomato juice, 1 pinch ground black pepper, (...)', they were split on commas so that each individual element could be added to the unflagged properties using the aforementioned format.

As was explained previously, the unflagged properties lists were used to flag what properties were available. If a match was found in the key words between a word and a property, then the unflagged properties list of each drink was analysed for an occurrence of that word, because not every drink had the same properties and thus not every list needed to be updated. If the word did occur, then the list entry was updated such that 'None' was replaced by 'True' or the entry was deleted, according to whether there was made use of a negation or not. The entry was only deleted if a negation occurred, because this implied that the property and thus the ordered drink were unavailable.

4.4 Natural language generation

This section will explain the approach that was taken to generate natural language. Starting with a description of the method that was used to generate questions in section 4.4.1, followed by the dialogue model in section 4.4.2.

4.4.1 Question generation

In order to generate natural language, there templates that contained variable parts were used, which can be seen in Figure 8. Although the templates were hand-scripted, the variable parts ensured that the questions that were asked were not always the exact same. The templates were used in combination with the unflagged properties lists, namely by randomly selecting the unflagged properties list of an ordered drink and then continuing to randomly select an index of this list. As was explained before, the robot had not been informed by the bartender about a property yet if its status was 'None'. Therefore, a question could be asked about this property to determine whether this property was available or if the bartender was capable of it. If the program was close to finishing, there would be a lot of properties that have 'True' instead of 'None' as a status and, therefore, randomly finding a None status was attempted five times. If no None status had been found after five tries, the list of unflagged properties was searched for the first occurrence of a None status and this property was used to generate a question.

1. Do you have any {colour} drinks?
2. Do you have any {alcoholic} drinks?
3. Do you have any {carbonated} drinks?
4. Do you have any {hot} drinks?
5. Do you have any {taste} drinks?
6. Do you have any {ingredient}?
7. Do you have a(n) {tool}?
8. Can you {action} drinks?

Figure 8: The templates that were used to generate questions that the robot asked the bartender.

Using the found property, a question could be generated by filling in the variable part of the corresponding template. Since the entry in the unflagged properties list looked as follows: [property_value: status // property_type_name], the name of the property type could be used to retrieve the correct template by searching for an occurrence of the type name. Moreover, the tool question template required an analysis to determine whether the tool started with a vowel, in which case the determiner 'an' had to be used instead of 'a'. New questions were generated until there was no longer any occurrence of 'None' in the unflagged properties lists or until there were no more unflagged properties lists, which implied that none of the ordered drinks were available.

4.4.2 Dialogue model

The robot followed a certain dialogue model in order to direct a conversation. Firstly, it would ask the bartender a question, after which it would obtain an answer

from the barman. There were several things that could happen after the robot had obtained the bartender's answer: firstly, the robot could have not heard what the bartender said, or, secondly, it could have not understood or, finally, it could have understood. If one of the first two options occurred, i.e. the robot did not hear or it did not understand, then the robot would keep attempting to obtain an answer until it understood what the bartender said. The robot always indicated what option occurred by saying sentences such as, 'Sorry, I don't understand. Could you use different words?', 'Sorry, what did you say?' or 'Okay, I understand.'. Several of such sentences were constructed for each of the three options so that the robot did not always say the exact same thing and the conversation would thus feel more natural. All of these sentences were saved in a text file, which was processed at the start of the program such that each option was saved under a distinctive name in a dictionary with the corresponding sentences in a list as its value. Identically to the method that was used to choose a question, a sentence was chosen by a random generator and this sentence was spoken by the robot. The approach that was used to regulate the dialogue can be seen in Figure 9.

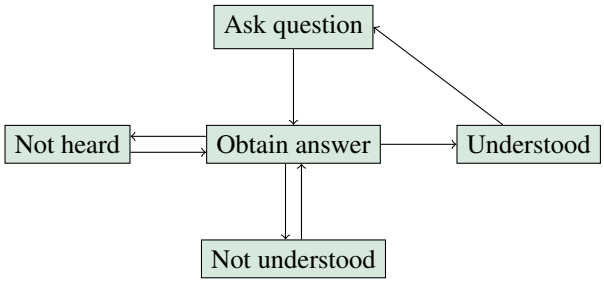

Figure 9: An illustration of the dialogue model that was used to direct a conversation.

5 Evaluation

There are two separate types of results that are obtained once the program finishes and each one has to be evaluated individually in a separate way. Firstly, it is important that the natural language understanding component works well and, therefore, the experiments were conducted using different bartenders, which ensures that the program is indifferent to, for example, variations in tone of voice and accent. The accuracy of the program, which represents whether the unavailable drinks have been correctly identified, will be evaluated using precision, recall and the F_1 measure. Precision (1) reflects how relevant the returned set is, while recall (2) measures how many relevant objects were recognised. Finally, the F_1 measure (3) rewards strong recall and precision, while punishing each deviation towards neglecting one. Furthermore, a false positive represents a drink that the robot identified as available even though it was not, while a true positive symbolises a drink

that was correctly identified as available. Moreover, a false negative represents a drink that was incorrectly identified as unavailable, while a true negative symbolises a drink that was correctly identified as unavailable.

$$\text{precision} = \frac{\text{true positives}}{\text{true positives + false positives}} \tag{1}$$

$$\text{recall} = \frac{\text{true positives}}{\text{true positives + false negatives}} \tag{2}$$

$$f_1 = 2 * \frac{\text{recall * precision}}{\text{recall + precision}} \tag{3}$$

Although these measures work well to evaluate whether the correct drinks were identified, they cannot be used to evaluate the naturalness of the conversation. Hence, this was evaluated by the bartenders using a survey form. This form asked to rate the naturalness of the conversation of a scale from one to ten, whether the correct drinks were identified, whether the bartender thought anything was lacking that would have made the conversation more natural and, finally, whether there was anything in particular that had made the conversation feel natural. The evaluation form can be seen in Appendix A.

6 Results

Figure 10: Two images of the evaluation procedure where a bartender had a conversation with the Pepper robot.

Similarly to the amount of subjects that evaluated the developed program by Avilés et al. (2010), ten different people tested the program that was developed in this research by filling the role of the bartender and answering the questions of their waiter, i.e. the Pepper robot. Some photographs of the setup of the evaluation method can be seen in Figure 10, in which two different bartenders are having a conversation with the robot. As was explained before, each bartender evaluated the

naturalness of the conversation and analysed whether the unavailable drinks were correctly identified.

On the evaluation form that was provided, the bartenders were asked to rank the naturalness of the conversation of a scale from one to ten. The results of this can be seen in Figure 11, from which the average of the given ranks was calculated to be 5.5. Moreover, they were asked what they thought was lacking in naturalness or what made the conversation feel unnatural. Many of them found the questions that the robot asked too repetitive and that the robot did not move its head enough or make enough eye contact. Furthermore, they indicated that there was not enough variety in the responses that the robot gave and that the overall experiment, as well as the processing time, took too long. In addition, a few bartenders thought that the robot lacked intonation, which made it difficult to understand what it was saying at times, and that having to hold the robot's hand to give an answer felt unnatural. Finally, one bartender would have appreciated an indication so that they knew that the robot had heard them and was processing their answer, which corresponds to the processing time being too long.

Rank	1	2	3	4	5	6	7	8	9	10
Frequency	0	0	1	1	0	2	4	1	0	0

Figure 11: The ranks that were given to the naturalness of the conversation on a scale from one to ten.

The bartenders were also asked if there was anything in particular that had made the conversation feel natural. Firstly, many bartenders indicated that the variation in responses and questions made the conversation feel more natural. However, there was not enough variation to truly make it feel natural. Secondly, they appreciated the hand and arm movements of the robot and found that, generally, the robot understood them well. Furthermore, they appreciated the possibility to ask the robot to repeat the question, as well as the natural structure of the robot's sentences and the robot's indication when it did not understand a response. Although some bartenders thought that the robot's intonation was difficult to understand, others indicated that they perceived a good intonation. Finally, it was indicated that the robot communicated well overall and that the robot's method to ask questions was natural.

Moreover, the bartenders analysed the correctness of the program, i.e. whether the unavailable drinks were correctly identified. The program worked correctly 4 out of 10 times, thus resulting in an accuracy of 40%. Furthermore, the precision was 0.625 (4), while the recall was 1 (5), which results in an F_1 measure of 0.769 (6).

$$\text{precision} = \frac{20}{20 + 12} \tag{4}$$

$$\text{recall} = \frac{20}{20 + 0} \tag{5}$$

$$f_1 = 2 * \frac{0.625 * 1}{0.625 + 1} \tag{6}$$

7 Conclusion

This section will start by discussing the obtained results and providing suggestions for future research in section 7.1 and close with the conclusion of this research in section 7.2.

7.1 Discussion and future work

The program was evaluated by ten different test subjects and it is therefore possible that results would have been different if it had been evaluated by more people. Furthermore, the program was evaluated by both first-year and third-year Artificial Intelligence bachelor students and it was noticeable that the first-year students were more fascinated by the program. This could be due to the fact that they are less accustomed to robots, causing them to rank the naturalness of the conversation slightly higher than the third-year students. Therefore, it would have been interesting to have subjects of different backgrounds, e.g. different bachelors, and inspect how they would rank the naturalness. In addition, it would have been interesting to have test subjects of other ages, e.g. older people that are not used to robots, because they might have found the conversation less or more natural than the current test subjects did on average.

Furthermore, the accuracy of the program was lower than expected, due to the fact that an action was always set to possible in the unflagged properties list if the given answer was 'No, I can't', because the negation was not detected in this case. This was caused by 'can' being the main verb in the answer, but the parser labelling the main verb of the answer as 'ca'. Since the negation is only detected if it is linked to the main verb, this resulted in no negation being found.

The program that was developed by Avilés et al. (2010) was evaluated using a form as well and each test subject ranked the naturalness of the conversation of a scale from one to four. By converting this into a scale from one to ten, it is possible to compare the average naturalness of both researches. The average naturalness by Avilés et al. (2010) was 8.5, while the average naturalness of this research was 5.5. This research thus produced less natural conversations. Moreover, the android ERICA possesses many human-like, and thus often perceived as natural, capabilities, such as facial expressively and a highly expressive speech synthesizer (Glas et al., 2016). Many test subject indicated that incorporating these elements would have improved the naturalness of the conversation with the Pepper robot. Thus, it can

thus be concluded that the developed program obtained less naturalness than the research by Glas et al. (2016). Padmakumar et al. (2017) developed a program that could be requested to perform two commands, namely navigation and delivery. It asked a full command and continued to ask for parameters until it understood the command. Due to the fact that it is only possible to ask for two commands, one could argue that this is less natural than having the possibility to converse with a robot about drinks.

The aforementioned discussion points can each be incorporated into future research. Furthermore, another suggestion is to integrate a face detection method into the program, because the fact that the Pepper robot did not make eye contact with the bartender made the conversation feel less natural. Moreover, there could be scripted more responses and there could be more variety in the questions that the robot asks. In addition, the program took very long to complete, both because of the processing time and because of the amount of questions that had to be asked. One method to reduce the amount of questions is by asking the questions not completely randomly. For example, by asking information about properties that occur least often in all of the drinks first so that these drinks can be eliminated if they are not available. Furthermore, the method that was used to answer, i.e. holding the robot's hand, made the conversation feel less natural and this could be improved by attaching an external microphone close to the bartender's mouth so that spikes in audio can be used to determine when to listen instead. Finally, a corpus that is based on conversations between bartenders and waiters could be used to train the parser so that sentences would be parsed more accurately.

7.2 Conclusion

The question that was researched was to what extent it is possible for a Pepper robot to have a natural conversation with a bartender to discover what drinks it can offer. Due to the fact that several steps needed to be taken before it would be possible to answer this research question, it was divided into three sub-questions. The first sub-question asked how to track what drinks the bartender can make, which was done by generating a database of drinks that included information about their properties. Using this database, it was possible to flag what properties the bartender possessed or was capable of.

In order to answer the second sub-question, the Pepper robot needed to be able to understand what the bartender says, which was achieved using the Google Cloud speech to text API, which returns a transcription of an audio file. From this transcription, the object, main verb and negation were obtained and matched against key words so that it was possible to flag the properties of the drinks and thus the drinks themselves.

The final sub-question was asked how the Pepper robot could generate language

to direct a conversation with the bartender. This was accomplished using question templates and hand-scripted sentences, which were used to clarify whether the robot had heard and understood the bartender.

Since these sub-questions have been answered, it is possible to answer the research question, because they define the steps that were taken. The null-hypothesis to the research question was that it is not possible to have a natural conversation such as described in the research question. However, the obtained results, which were described in section 6, i.e. the ranked naturalness of 5.5 on average, the precision of 0.625 and the recall of 1, suggest that it is possible. The conclusion that can be drawn is thus that, to some extent, it is possible for a Pepper robot to have a natural conversation with a bartender to discover what drinks it can offer. However, there is still room for improvement, which was discussed in section 7.1.

Furthermore, in the introduction, the claim that the developed program would be applicable to different contexts was made. Since the properties that questions are generated about are variable, this claim can be realised. In addition, all other elements in the program are variable as well. For example, it is possible to generate a new database and base the key words on that database instead or use different question templates and so forth.

To summarise, although the obtained accuracy was low, it is possible to improve it using the information that was given in section 7.1. If these improvements are incorporated, then it will be possible to use several elements of the program in the RoboCup@Home Cocktail Party Challenge. For example, the method to receive a transcription of an audio file, i.e. using the Python library `pexpect` and the Google Cloud speech to text API, could be employed, as well as the method to understand sentences, namely by using the object, main verb and negation of a sentence. If the proposed improvements will be incorporated, the UvA@Home team will likely perform well at the cocktail party challenge.

8 References

Avilés, H., Alvarado-González, M., Esther, V., Rascón, C., Meza, I. V., and Pineda, L. (2010). Development of a tour–guide robot using dialogue models and a cognitive architecture. In *Advances in Artificial Intelligence – IberoAmerican Society of Artificial Intelligence 2010: 12th Ibero-American Conference on AI, Bahía Blanca, Argentina, November 1-5, 2010. Proceedings*, volume 6433, pages 512–521, Berlin, Heidelberg. Springer Berlin Heidelberg.

Bird, S. (2006). Nltk: The natural language toolkit. In *Proceedings of the International Committee on Computation Linguistics and Association for Computational Linguistics on Interactive Presentation Sessions*, International Committee on Computation Linguistics and Association for Computational Linguistics '06,

pages 69–72, Stroudsburg, PA, USA. Association for Computational Linguistics.

Chan, W., Jaitly, N., Le, Q. V., and Vinyals, O. (2016). Listen, attend and spell: A neural network for large vocabulary conversational speech recognition. In *International Conference on Acoustics, Speech and Signal Processing*, page 2.

De Marneffe, M.-C. and Manning, C. D. (2008). Stanford typed dependencies manual. Technical report, Stanford University.

Georgeff, M., Pell, B., Pollack, M., Tambe, M., and Wooldridge, M. (1999). The belief-desire-intention model of agency. In Müller, J. P., Rao, A. S., and Singh, M. P., editors, *Intelligent Agents V: Agents Theories, Architectures, and Languages: 5th International Workshop, ATAL'98 Paris, France, July 4–7, 1998 Proceedings*, volume 1555, pages 1–10, Berlin, Heidelberg. Springer Berlin Heidelberg.

Ghahramani, Z. (2001). An introduction to hidden markov models and bayesian networks. *International Journal of Pattern Recognition and Artificial Intelligence*, 15(01):9–42.

Glas, D. F., Minato, T., Ishi, C. T., Kawahara, T., and Ishiguro, H. (2016). Erica: The erato intelligent conversational android. In *2016 25th Institute of Electrical and Electronics Engineers International Symposium on Robot and Human Interactive Communication (RO-MAN)*, pages 22–29.

Jurafsky, D. (2000). *Speech and language processing: an introduction to natural language processing, computational linguistics and speech recognition*. Prentice Hall.

Lee, S. and Son, Y. J. (2008). Integrated human decision making model under belief-desire-intention framework for crowd simulation. In *2008 Winter Simulation Conference*, pages 886–894.

Lemaignan, S., Ros, R., Sisbot, E. A., Alami, R., and Beetz, M. (2012). Grounding the interaction: Anchoring situated discourse in everyday human-robot interaction. *International Journal of Social Robotics*, 4(2):181–199.

Lemaignan, S., Warnier, M., Sisbot, E. A., Clodic, A., and Alami, R. (2017). Artificial cognition for social human–robot interaction: An implementation. *Artificial Intelligence*, 247:45 – 69. Special Issue on {AI} and Robotics.

Mavridis, N. (2015). A review of verbal and non-verbal human–robot interactive communication. *Robotics and Autonomous Systems*, 63:22 – 35.

Miller, G. A. (1995). Wordnet: A lexical database for English. *Communications of the Association for Computing Machinery*, 38(11):39–41.

O'Hare, G. M. and Jennings, N. (1996). *Foundations of distributed artificial intelligence*, volume 9. John Wiley & Sons, New York, NY, USA.

Padmakumar, A., Thomason, J., and Mooney, R. J. (2017). Integrated learning of dialog strategies and semantic parsing. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics (EACL 2017)*, pages 547–557, Valencia, Spain.

Perera, V., Pereira, T., Connell, J., and Veloso, M. M. (2017). Setting up pepper for autonomous navigation and personalized interaction with users. *Computing Research Repository*, 1704.

Petrick, R. P. and Foster, M. E. (2013). Planning for social interaction in a robot bartender domain. In *International Conference on Automated Planning and Scheduling*.

Sainath, T. N., Vinyals, O., Senior, A., and Sak, H. (2015). Convolutional, long short-term memory, fully connected deep neural networks. In *2015 Institute of Electrical and Electronics Engineers International Conference on Acoustics, Speech and Signal Processing*, pages 4580–4584.

Shibata, T. (2011). *Importance of Physical Interaction between Human and Robot for Therapy*, pages 437–447. Springer Berlin Heidelberg, Berlin, Heidelberg.

Taniguchi, T., Nagai, T., Nakamura, T., Iwahashi, N., Ogata, T., and Asoh, H. (2016). Symbol emergence in robotics: a survey. *Advanced Robotics*, 30(11-12):706–728.

Visser, A. (2017). A new RoboCup@Home challenge. *Benelux AI Newsletter*, 31(1):2–6.

A Evaluation form

In Figure 12, the form that was used to evaluate the program is shown.

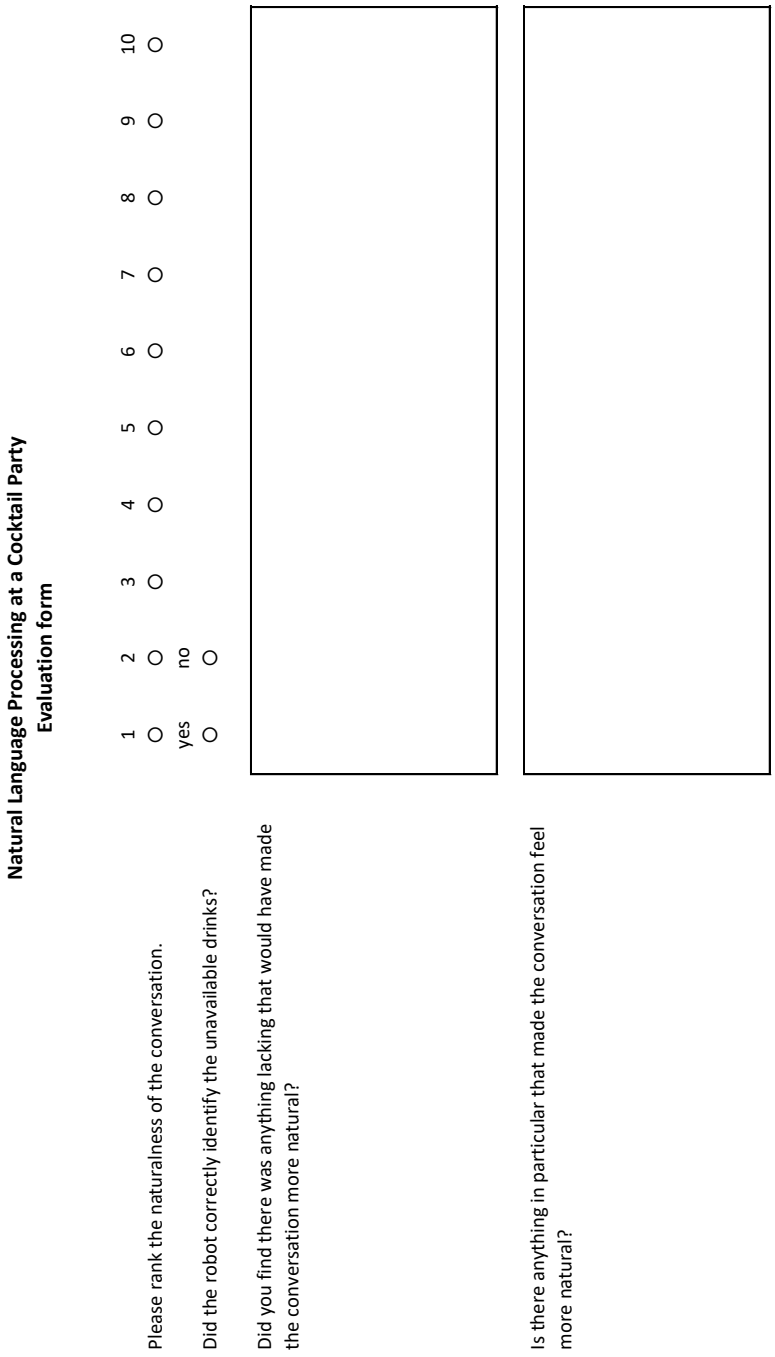

Figure 12: The form that was used to evaluate the naturalness of the conversation and the accuracy of the drinks identification.