Expectation-based Command Recognition Off the Shelf:
Publicly Reproducible Experiments with Speech Input

Dominik Ertl*  
Bosch Engineering GmbH  
Hüttenbrennergasse 5  
A-1030 Vienna, Austria  
dominik@ertl.in

Jürgen Falb, Hermann Kaindl, Roman Popp and David Raneburger  
Vienna University of Technology, Institute of Computer Technology  
Gusshausstrasse 27-29  
A-1040 Vienna, Austria  
{falb, kaindl, popp, raneburger}@ict.tuwien.ac.at

Abstract

When striving for a cheap implementation of command recognition for speech input today, you may resort to off-the-shelf tools. In contrast to specific research approaches, such tools by themselves do not take expectations for certain commands in a given situation into account. Such expectations will usually be available both in “intelligent” and more conventional programs using this speech interface, and they should be used to improve command recognition. We propose to make use of a set of expected commands at a given state of a dialogue and a list of ranked command hypotheses from basic speech recognition. We devised and implemented this with two approaches for speech input. One specializes a given grammar according to expected commands at each dialogue state, the other accepts the highest-ranked hypothesis for a command that fits the expected ones at a given state. The latter approach achieved a statistically significant improvement of the command success rate in an experiment, as compared to ignoring the expectations. Since everything is freely available, we made these experiments publicly reproducible.

1 Introduction

Command recognition for modalities such as speech is hard for machines. This is especially true for untrained users and an untrained recognition system. If one needs a quick and inexpensive solution, speech recognizers are available off the shelf these days, but they implement only bottom-up processes during recognition, since they have no information about expected commands available at a given state of the dialogue. Specific research approaches have already taken expectations in a given situation into account for additional top-down processing. How can this idea be made available when using off-the-shelf recognizers? In particular, can this be achieved as a “$1 Recognizer” for speech in the same spirit as for gestures in [17]?

Let us assume that a typical off-the-shelf recognizer provides a list of ranked command hypotheses from its bottom-up recognition processes, ranked according to some measure of how likely such a command was actually given by the user. Let us also assume that the application logic of the system to be guided by speech commands can deliver a set of commands expected at a given state, where the expected commands form a subset of the larger set of all possible commands. Both assumptions can typically be fulfilled today even when using off-the-shelf tools for implementing a command-guided system.

When given a list of ranked command hypotheses and a set of expected commands at some state of the dialogue, both should be used for expectation-based command recognition. First, we instantiated this approach by reducing the grammars for speech commands, so that only expected commands could be recognized at all at a given state of the dialogue. While this is the usual approach and it looked promising, this led to a high rate of false positives, especially since the grammars map unrelated input or even noise to one of the expected commands.

So, we investigated another instantiation of this approach, where the highest-ranked command hypothesis is accepted that matches the set of expected commands. Using this simple but effective technique, we achieved a statistically significant improvement of the command success rate in an experiment, where we compared it with simply accepting the highest-ranked command hypothesis without taking an expectation into account. In fact, this expectation-based recognition can never be worse than the one without using an expectation for recognizing a given command. In terms of false positives, we could not reject the hypothesis that these approaches are equally good.

As a running example and as the domain for our controlled experiments, we use a selected part of a speech user interface (UI) of a robot shopping application. Since everything...
is freely available, we made this experiment publicly reproducible.\footnote{Everything related to these experiments is available at http://www.ict.tuwien.ac.at/staff/kaindl/julius/}

The remainder of this paper is organized in the following manner. First, we review state-of-the-art approaches and provide background material. Then we present our approach to expectation-based command recognition and its two instantiations that we devised and implemented. Based on that, we report on experiments with this approach and their results. Finally, we discuss this approach more generally.

2 State of the Art

Cognitive scientists stated some time ago that expectations are ubiquitous and used them, for example, to achieve a better understanding of human reasoning \cite{5}. Multi-agent systems use expectations to reason about their interaction with other agents \cite{13}. Even long time ago, it was found that exploiting linguistic structures, which can be interpreted as expectations as well, can improve speech recognition \cite{8}. In particular, the idea of using dialogue contextual knowledge to improve speech recognition and understanding is certainly not new. We discuss here a selection of previous work that has introduced a variety of approaches for implementing this idea.

Fuse is an example of a situated spoken language understanding system that uses visual context to steer the interpretation of speech. Roy et al. \cite{11} show that with a given visual scene and a spoken description, the system finds the object in the scene that best fits the meaning of the description. To solve this task, Fuse performs speech recognition and visually grounded language understanding. Rather than treating these two problems separately, knowledge of the visual semantics of language and the specific contents of the visual scene are fused during speech processing. With Fuse one can solve a dedicated task, namely finding objects in a scene. We strived for using expectations more generally for improving the recognition of commands from speech input.

The approach of Young \cite{18} uses contextual knowledge to predictively generate expectations about the conceptual content that may be expressed in a system user’s next utterance. Young’s approach expands the expectations to constrain the possible words that may be matched from an incoming speech signal. It translates the predictions into a grammar to dictate legal word sequences and to constrain the recognition process. Such an approach, however, enhances the number of false positives when the grammars are smaller.

Fuegen et al. \cite{3} analyze the tight coupling of speech recognition and dialogue management with dialogue-context dependent grammar weighting. They restrict the search space of their grammar-based speech recognizer through the information given by the dialogue manager. According to the authors, the flexible context-free grammar implementation of their speech decoder Ibis allows weighting of specific rules at run-time to restrict the search space of the recognizer for the next decoding step. These rules are given by the dialogue manager depending on the current dialogue context. In the domain of household-robots they improved the Word Error Rate (WER) by 3.3\% at close talking distance and by 9.9\% at a longer talking distance to the robot.

Lison and Kruijff \cite{7} present salience-driven contextual priming of speech recognition for human-robot interaction (HRI). Priming means focusing the domain of words or word sequences that a system can expect next. We prime the dialogue between user and machine at design time with the help of a UI Behavior Model. Lison and Kruijff postulate that in HRI, speech recognition performance can be improved by exploiting knowledge about immediate physical situation and dialogue history. They dynamically perform updates of their statistical language model, and establish expectations about which words are most likely to be heard in a given context. In contrast, we claim that both in HRI and HCI modality input recognition performance can be improved by matching possible input with expected input. Up to now, salience-driven priming has been used in a dedicated cognitive architecture for mobile robots, whereas our expectation-based approach can be applied more generally and with off-the-shelf tools.

Weilhammer et al. \cite{15} present an approach for rapidly building language models for dialogue systems. They show that they can almost halve the WER by combining language models generated from a simple task grammar with a standard speech corpus and data collected from the Web using a sentence selection algorithm based on relative perplexity. As a result, they advocate the use of statistical language models (SLMs) in speech recognizers for dialogue systems.

Gabsdil and Lemon \cite{4} investigated the use of machine learners trained on a combination of acoustic confidence and pragmatic plausibility features that are computed from
dialogue context to predict the quality of incoming n-best recognition hypotheses to a spoken dialogue system. These predictions are then used to select a “best” hypothesis and to decide on appropriate system reactions. They claim that their approach is more generally applicable than preceding research, since they frame their methodology in the Information State Update (ISU) approach to dialogue management and, therefore, expect it to be applicable to a range of comparable multimodal dialogue systems that use speech input as well. This ISU approach collects information relevant to dialogue context in a central data structure from which it can be easily extracted.

Williams [16] presents another approach to make use of n-best lists for reducing dialogue understanding errors. This approach actually detects commonality over multiple n-best lists and assigns a joint likelihood using a dialogue model. An illustrative example for his approach is asking the same question twice and combining the two n-best lists of the answers. If different answers appear in the top positions of the two lists, but the answer on the second position is the same, then intuitively this second answer ought to be the candidate for the user’s intention. Williams shows that this approach finds the correct user goal more often than approaches that simply take the first answer of the n-best list.

We use a single n-best list of commands and match it with a list of expected commands in one of our approaches. In general, we strived for simple approaches to utilizing expectations for improving command recognition. Their implementation should not depend on the availability of complex custom-designed software. Therefore, we made use of an off-the-shelf speech recognition tool. Our approaches interface this tool in a simple way with finite-state machines often available for maintaining dialogue state and representing expectations on commands for these states.

For at least 30 years, academia and industry have the need to measure the performance of speech recognition systems. For example, Baker et al. [1] discuss the intentions for appropriately designed controlled experiments as necessary tools by which both research and commercial systems must assess the value of different algorithms and approaches. This facilitates also public reproducibility, and we present reproducible research results of controlled experiments. As pointed out in, e.g., Vandewalle et al. [12], reproducibility is an important factor when it comes to comparison or verification of experimental research results. However, it is still a challenging task to obtain reproducibility when rebuilding a dedicated architecture or algorithm and setting up an experiment that tests a hypothesis along the lines of the algorithm. Hence, we based our work on the insights of Vandewalle et al. and made everything about our experiments available for their potential reproduction.

3 Background for our Approach

The essential background for our approach is given by a simple bottom-up technique for command recognition (without taking expectations on commands into account), and UI Behavior Models (such as those for speech input) that carry expectations on commands.

3.1 Bottom-up Command Recognition

Bottom-up command recognition takes analogue modalities like speech input in the form of low-level (sub-symbolic) physical signals from humans, and interprets them into ranked command hypotheses. Such hypotheses are symbolic representations of what the toolkit implementing this approach believes to be the input. Every hypothesis within the ranked hypotheses has a score. The scale of the score is toolkit-dependent.

As a baseline of our speech input UI, we use this bottom-up recognition with a freely available off-the-shelf speech toolkit named Julius, which provides ranked command hypotheses for a given speech input. The bottom-up command recognition implemented there is configured with the complete grammar that includes all commands from all states of the dialogue. For a given speech input, it takes the command of the hypothesis with the highest score and does not consider other hypotheses.

However, this is not necessarily expected input for the machine at the given state of the dialogue. When using this straight-forward approach, the command recognition of analogue modalities is only dependent on the toolkit (and its configuration) and does not take expectations of the machine into account.

3.2 UI Behavior Model

Most UIs have implicitly or explicitly finite-state machines defining their behavior. Each state of such a machine corresponds to one state of the dialogue. It has a list of allowed user inputs (in our application commands), which lead to a state change. This state machine represents a UI Behavior Model and is usually executed by a dialogue manager. For implementing expectation-based command recognition according to our approach, we need this UI Behavior Model in explicit form with the possible commands specified for each state. In principle, such a UI Behavior Model can be created manually or generated automatically from higher-level specifications such as ConcurTaskTrees [9] or Discourse-based Communication Models [10].

Since the UI Behavior Model is modality-independent, we also need a modality-specific representation of the commands. In the case of a grammar-based speech recognizer, one needs a mapping from the recognizable user input to the modality-independent command representation. Either a UI designer creates such a mapping manually, or a semi-automatic generation process generates the mapping from higher-level interaction specifications [2].

For our running example from an implemented robot shopping application, we use the excerpt illustrated in Figure 1. This finite-state machine representing the UI Behavior Model only has three states (“PU” in the figure stands for Presentation Unit). The state Main Screen allows 14 different commands. In this state, the user can choose to manage her shopping list, and the state machine switches to its Shopping Edit state. Alternatively, the user can tell her semi-autonomous robot shopping cart where to move, and the state machine switches to its On The Move state. In the Shopping Edit state, 6 different commands are possible, and the user can only switch back to the state Main Screen. The same holds for state On The Move, where only 2 commands are possible. This makes a total of 22 commands possible in the dialogues defined through this UI Behavior Model. The whole command language is given in the appendix.

4 Expectation-based Command Recognition

We developed two different approaches for expectation-based command recognition from speech input. The first approach maps given expectations to the modality level (the grammars of a speech recognizer), according to the same basic idea as in [14, 18], to enhance the Command Success Rate by considering expectations on the modality level. In contrast, our second approach matches ranked command hypotheses resulting from the modality level with the expected commands defined for the given state. Both approaches are usable with any off-the-shelf modality toolkit that provides ranked hypotheses, since the toolkit is only loosely coupled with the state machine.

Both approaches are implemented using the speech toolkit and the UI Behavior Model sketched above. The latter provides the command expectations for the former.

In more detail, the first approach maps the UI Behavior Model, which represents the expectations on commands independently of a modality, to a modality-specific representation of the expectations. More specifically, it splits the complete speech input grammar into dynamically loadable partial grammars for the possible commands. For each state of the state machine representing the UI Behavior Model, there is a partial grammar, and for each transition, there is a possible command. So, the partial grammars reflect the states and transitions defined in the UI Behavior Model, and thus the expectations of the machine. We briefly denote this expectation-based approach as PG (for partial grammars).

Our second approach for expectation-based command recognition matches commands of the ranked hypotheses with the expected commands. Figure 2 illustrates this approach for command recognition with expectations on the modality-independent level in more detail. In the lower part of the figure, there is an off-the-shelf recognizer that provides ranked hypotheses for a user’s input. The commands corresponding to these ranked hypotheses are first translated into the representation used by the state machine and then matched against the expected commands defined there.
5 Experiments

To test our two expectation-based approaches against the baseline of our bottom-up command recognition from speech input, we performed controlled experiments. Since everything is freely available, we made these experiments publicly available. So, all the results can be reproduced with the recorded material and the test bench.

5.1 Setup of the Experiments

These experiments have been set up for speech input of our robot shopping application. We evaluated potential advantages for both, the approach with partial grammars through expectations and the approach with complete grammar matching expectations.

Recording Speech Input Data  According to this application, we recorded 22 different input commands from four participants (untrained users, \( n = 4 \) (3 male, 1 female), 10 times each. Moreover, we recorded 8 types of utterances that are likely to be said when using such a speech UI in a public area, like ehm, kchkch, yes, no, etc. In total, we have a database of 1,200 raw audio files for input, 880 for commands and 320 for other utterances.

The Given Speech Input Recognizer Toolkit  We used the Julius speech recognizer toolkit for our experiments. It is a robust speaker-independent, large-vocabulary speech recognition system and freely available for different operating systems, PC platforms and even embedded devices. Julius uses a context-free grammar with sentences. Hacioglu et al. [6] discuss that such grammars provide complete syntactic analysis across a sentence, considering all words. These grammars are supposed to work well for grammatical sentences (those covered by the grammar), but completely fail for sentences of ungrammatical construction. So, their use for applications with spoken natural language is limited.

We use voxforge-2010-03-02_16kHz_16bit_MFCC_O_D as acoustic model. This is a free speech corpus and acoustic model repository for open source speech recognition engines. An acoustic model contains statistical representations of sound that make up words in a language.

Finally, we configured the physical parameters of the recognizer (like sampling-rate, etc.) by following the recommendations in the Julius documentation.
Input Grammars  For our experiments we use an input grammar that contains all the commands from our UI behavior state machine. This grammar does not take the state of the dialogue into account and is used for the baseline approach as well as the approach with the complete grammar matching expectations. In contrast, we generated the specific grammars according to the UI Behavior Model for the partial grammar approach.

We ran the speech recordings through the Julius recognition system using the complete grammar and the partial grammars (all legal and non-legal sentences). We configured Julius to accept the complete list of audio files as input, which have been interpreted as speech input one after another. The recognized output was automatically parsed for recognized and not recognized commands as well as false positives. Each output for one given speech input contains an ordered list of hypotheses on the believed input command (that is part of the grammar) and a related score. In our experiments, we limited the number of command hypotheses to a maximum of 5, as the score typically degenerates for further hypotheses. The parsed output was subject for further statistical interpretation, which we present below.

Null Hypotheses  We defined two null hypotheses:

• NH1: There is no statistically significant difference (p-value=0.01) for the Command Success Rate between command recognition with expectations on the modality level (partial grammar, PG) and command recognition without expectations (complete grammar, CG).

• NH2: There is no statistically significant difference (p-value=0.01) for the Command Success Rate between command recognition with expectations on the modality-independent level (complete grammar matching expectations, CGExp) and command recognition without expectations (complete grammar, CG).

The metric Command Success Rate ($CSR$) can be easily calculated for each command as $CSR = \frac{\text{number of successfully recognized commands}}{\text{total number of commands}}$.

5.2 Results of the Experiments

In order to report on the results of these experiments, we first compare the $CSR$ scores of both expectation-based approaches with the baseline approach, which uses the complete grammar without expectations. Then we present data on false positives.

Table 1 compares the number of recognized commands for the expectation-based approach using partial grammars (PG), where the grammar is switched according to the expectations of the dialogue, with the baseline approach using the complete grammar (CG). We tested both approaches with the same set of 880 voice inputs (4 users repeated 22 commands 10 times), but for each partial grammar we tested only the commands possible with this partial grammar. Table 1 shows that using the expectations lead to 21 additionally recognized commands, which improves the $CSR$ score from 0.925 to 0.95.

<table>
<thead>
<tr>
<th>Recognized (PG)</th>
<th>no</th>
<th>yes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Recognized (CG)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>no</td>
<td>45</td>
<td>21</td>
</tr>
<tr>
<td>yes</td>
<td>0</td>
<td>814</td>
</tr>
</tbody>
</table>

Performing the same tests with the expectation-based approach using the complete grammar (CGExp), we can see in Table 2 that the number of recognized commands is higher (34 instead of 21) than with the PG approach. Our second expectation-based approach, therefore, has a $CSR$ score of 0.96.

Table 2. Number of recognized commands for the complete grammar matching expectations versus complete grammar without expectations (the baseline approach).

<table>
<thead>
<tr>
<th>Recognized (CGexp)</th>
<th>no</th>
<th>yes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Recognized (CG)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>no</td>
<td>32</td>
<td>34</td>
</tr>
<tr>
<td>yes</td>
<td>0</td>
<td>814</td>
</tr>
</tbody>
</table>

Both our approaches increased the $CSR$ scores as compared to the baseline approach with the complete grammar (CG). According to the McNemar test, however, the null hypothesis NH1 cannot be rejected, so there is no statistically significant difference of the CSR scores between the
PG and the baseline CG approaches. In contrast, the null hypothesis NH2 can be rejected according to this test. This means that the improvement of the CSR score through the CGExp approach over the baseline approach is statistically significant.

In addition to the CSR the number of false positives FP is also very important to determine the performance of the speech recognition approach. For expectation-based command recognition, we distinguish three categories of false positives:

- **FP type 1 (FP1)** arises when the user utters command A of the currently active grammar and the robot wrongly recognizes command B of this grammar.

- **FP type 2 (FP2)** arises when the user utters command A from a partial grammar not loaded at the moment and the robot wrongly recognizes command B of the currently active grammar. This FP type is (only) possible when dynamically loadable partial grammars are used as in our PG approach.

- **FP type 3 (FP3)** arises when the user utters a word/sentence that is in none of the grammars used and the robot wrongly recognizes command A which is in the currently active grammar.

FP2 and FP3 arise because a user utters commands/words (whether on purpose or not does not matter) that is not defined in the currently active grammar. It is subject of future research to investigate how likely this kind of utterance is compared to predefined input, in order to weigh the experiment results of the different types of false positives.

First, we measured false positives FP1 for the PG and CG-exp approach. In Table 3 we see that there are 5 FPs arising only in the PG approach and 4 FPs arising only in the CG baseline approach. Thus, there is almost no difference (only 1 FP). This difference is not statistically significant, as the results of the McNemar test show.

The CGExp approach slightly reduced the number of FP1 false positives (3 FPs arising only in this approach) as shown in Table 4. However, there is also no statistically significant difference to the baseline approach in this respect.

### Table 4. FP1 false positives for the CGExp versus the CG baseline approach.

<table>
<thead>
<tr>
<th>FP1 (CGexp)</th>
<th>no</th>
<th>yes</th>
</tr>
</thead>
<tbody>
<tr>
<td>no</td>
<td>861</td>
<td>3</td>
</tr>
<tr>
<td>yes</td>
<td>7</td>
<td>9</td>
</tr>
</tbody>
</table>

The situation changes completely when we also obey utterances, like yes or no, which are not defined in any of our grammars and result in FP3 false positives. In the PG approach, the utterances include also commands that would be valid in other partial grammars and in the complete grammar resulting in FP2 false positives. Table 5 shows that the number of false positives (FP2 + FP3) increases dramatically as the grammar becomes smaller. The smallest grammar in the PG approach is the on the move grammar, which contains only two commands. All utterances get mapped to these two commands and thus lead to the high number of 1102 false positives. In practice, this number may well decrease, if users utter only a small percentage of the tested utterances due to their knowledge of the context.

### Table 5. FP2 + FP3 false positives for the partial grammar approach (PG).

<table>
<thead>
<tr>
<th># Avail. Commands</th>
<th>PG: main screen</th>
<th>PG: shopping edit</th>
<th>PG: on the move</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>22</td>
<td>22</td>
<td>22</td>
</tr>
<tr>
<td>FP2 + FP3</td>
<td>165</td>
<td>946</td>
<td>1102</td>
</tr>
</tbody>
</table>

Regarding FP2 + FP3 false positives, the modality-independent expectation-based approach CGexp (where FP2 is not even possible) has a clear advantage over the PG approach. It has about the same low false positive rate as the baseline approach. Table 6 shows the results of a test performed by uttering 53 commands not specified in the grammar. 30 to 45 of the 53 commands are recognized as false positives (FP3). The remaining ones are not recognized at all, which is the desired result.

Overall, utterances not in the grammar may deteriorate the
6 Discussion

The results of the experiments show that both expectation-based approaches have increased the command success rate of an off-the-shelf speech recognizer, while only the complete grammar matching expectations approach achieved a statistically significant improvement. With a pure grammar-based speech recognizer, however, the false positive rate increases dramatically for degenerated partial grammars with a small (1 to 10) number of commands (more precisely for out-of-grammar false positives). This is an inherent property of grammar-based recognizers, as such grammars do not have an else-branch to ignore or reject illegal input. In contrast, the complete grammar matching expectations approach has the advantage that it can trigger clarification or helper dialogues, when it cannot recognize the input.

We aim at expectation-based UIs that can be built with off-the-shelf components and that have only low coupling between these components and the dialogue manager. Especially the complete grammar matching expectations approach does not even need partial grammars, just a list of ranked hypotheses to match with a set of expected commands. A tool can be simply and loosely connected with a given dialogue manager. Both of our techniques can even be used on different devices, where recognizers for speech with ranked hypotheses as output are available (e.g., Embedded Julius, PocketSphinx toolkit5, etc.).

There is space for further improvement of the command recognition by improving the CSR and decreasing the FP rates. For example, we could use the modality-dependent hypotheses scores to define a threshold where a hypothesis is not accepted if its score is below the threshold. Whenever no hypothesis is accepted, this should lead to an else-branch to ignore or reject illegal input. This would reduce the FP rate, especially for the partial grammars approach for implementing expectations.

Another approach to reduce the FP rate are filler/garbage models. Such models create a robust grammar that supports the rejection of out-of-grammar utterances and the recognition of utterances with pre- or post-ambles. These are words that are either uttered before or after a command. An example would be the utterance of "Can you GUIDE ME TO APPLES please", where the command GUIDE ME TO APPLES is surrounded by the words can you and please. The filler model approach presented by Yu et al. [19] is particularly interesting in the context of our work, because it is applicable to standard speech recognition engines. Therefore, it can be integrated with any of our approaches.

7 Conclusion

Command recognition from, e.g., speech input can be improved by including expectations. Evidence for this intuitively obvious and known result has, however, only rarely been reported in the literature. We provide another data point for such evidence through our approach to using information on expected commands at a given dialogue state for selecting a command from a list of ranked command hypotheses generated from speech input. In order to make the related experiments reproducible, we made them publicly available.

This approach is surprisingly simple but effective. In hindsight, it may even look trivial, but it still had to be found and evaluated. To our best knowledge, it is also new to employ expectations given from the application logic for selecting from a single n-best list of commands. An important property of this approach is its robustness against false positives as compared to approaches with restricted grammars.

We conjecture that this approach may also be more widely usable and useful, e.g., for gesture recognition, and we

Table 6. $FP_3$ false positives for the expectation-based approach (CGExp).

<table>
<thead>
<tr>
<th>CGExp: Main Screen</th>
<th>$FP_3$</th>
<th>not recognized</th>
</tr>
</thead>
<tbody>
<tr>
<td>CGExp: Shopping Edit</td>
<td>30</td>
<td>23</td>
</tr>
<tr>
<td>CGExp: On the Move</td>
<td>36</td>
<td>17</td>
</tr>
</tbody>
</table>

413
plan to investigate this. For achieving improvements, we made no attempt to improve the basic machinery for speech recognition, which is freely available. So, improvements can be achieved today with off-the-shelf tools. Also for including the expectations on commands, we did not have to invent anything new. Most user interfaces of today’s software have a finite-state machine for representing their behavior, explicitly or implicitly. We took an explicitly given state machine, where at each dialogue state the expected commands are defined. The only change necessary for the improvement was to connect both available machines in a simple way and with loose coupling.

8 Acknowledgment

Part of this research has been carried out in the CommRob project (http://www.commrob.eu), partially funded by the EU (contract number IST-045441 under the 6th framework programme).

References

A The Defined Speech Commands

In this appendix we present the defined speech commands, which we used in our experiments.

All commands: FOLLOW ME, GUIDE ME, GUIDE ME TO CASHIER, GUIDE ME TO NEXT ITEM ON SHOPPING LIST, GUIDE ME TO NEXT PRODUCT, MEET ME, MEET ME AT CASHIER, MEET ME AT NEXT ITEM ON SHOPPING LIST, MEET ME AT NEXT PRODUCT, STOP TROLLEY, CONTINUE, PAUSE, RESUME, RETURN TROLLEY, ACCEPT THAT, REJECT THAT, EDIT SHOPPING LIST, MANAGE SHOPPING LIST, SELECT, PRODUCT, REMOVE PRODUCT, SAVE SHOPPING LIST, FINISH SHOPPING LIST.

All these commands were uttered in the experiment. In addition, we used the following — not defined — speech utterances in the experiments: HELLO, GOOD BYE, YES, NO, EHM, *kchkch* (coughing).

Defined commands in the state MAINSCREEN: FOLLOW ME, GUIDE ME, GUIDE ME TO CASHIER, GUIDE ME TO NEXT ITEM ON SHOPPING LIST, GUIDE ME TO NEXT PRODUCT, MEET ME, MEET ME AT CASHIER, MEET ME AT NEXT ITEM ON SHOPPING LIST, MEET ME AT NEXT PRODUCT, CONTINUE, RESUME, RETURN TROLLEY, EDIT SHOPPING LIST, MANAGE SHOPPING LIST.

Defined commands in the state SHOPPING: ACCEPT THAT, REJECT THAT, SELECT PRODUCT, REMOVE PRODUCT, SAVE SHOPPING LIST, FINISH SHOPPING LIST.

Defined commands in the state ON THE MOVE: STOP TROLLEY, PAUSE.