A Self-Learning Assistive Vocal Interface Based on Vocabulary Learning and Grammar Induction

Jort F. Gemmeke[1]*, Janneke van de Loo*[2]*, Guy De Pauw*[2]*,*
Joris Driesen[1]*, Hugo Van hamme*[1]*, Walter Daelemans*[2]

[1]ESAT - PSI Speech Group, KU Leuven, Leuven, Belgium
[2]CLiPS - Computational Linguistics, University of Antwerp, Antwerp, Belgium
jort.gemmeke@esat.kuleuven.be, janneke.vandeloo@ua.ac.be, guy.depauw@ua.ac.be,
joris.driesen@esat.kuleuven.be, hugo.vanhamme@esat.kuleuven.be, walter.daelemans@ua.ac.be

Abstract

This paper introduces research within the ALADIN project, which aims to develop an assistive vocal interface for people with a physical impairment. In contrast to existing approaches, the vocal interface is self-learning which means it can be used with any language, dialect, vocabulary and grammar. The paper describes the overall learning framework, and the two components that will provide vocabulary learning and grammar induction. In addition, the paper describes encouraging results of early implementations of these vocabulary and grammar learning components, applied to recorded sessions of a vocally guided card game, patience.

Index Terms: language acquisition, word finding, grammar induction, non-negative matrix factorization, concept tagging

1. Introduction

Physically impaired people with restricted (upper) limb motor control are permanently in the situation where voice control could significantly simplify some of the tasks they want to perform. By regaining the ability to control devices in the living environment, voice control could contribute to their independence of living and their quality of life. Unfortunately, the speech recognition technology employed for voice control typically forces users to adhere to a restrictive grammar and vocabulary in order to successfully *command and control* a device.

In this paper, we introduce research in the ALADIN project[1], which aims to develop an assistive vocal interface for people with a physical impairment. In contrast to existing vocal interfaces, the ALADIN system is self-learning: The interface should automatically *learn* what the user means with commands, which words are used and what the user's vocal characteristics are. In other words: the interface should adapt to the user instead of the other way around.

This paper presents a functional description of the ALADIN learning framework and describes feasibility experiments with self-learning word finding and grammar induction modules. In Section 2 we outline the overall learning framework, the knowledge representation that is used and the rationale behind the word finding and grammar induction modules. In Section 3 we describe the experimental setup, followed by a discussion of the experimental results in Section 4. We conclude with some thoughts on future work in Section 5.

[1]Adaptation and Learning for Assistive Domestic Vocal INterfaces. Project page:
www.esat.kuleuven.be/psi/spraak/projects/ALADIN

2. The ALADIN framework

The ALADIN learning framework consists of several modules, which are shown schematically in Fig. 1. On the left-hand side, the provided input is shown, which consists of a spoken utterance (command) coupled with a control input, such as the button press on a remote control or a mouse click, possibly augmented with the internal state of a device (for example the current volume of a television).

During training, the *word finding* module (Section 2.2) builds acoustic representations of recurring acoustic patterns on the basis of a (small) set of training commands. This process uses features extracted from the audio signal and is constrained by a semantic *frame description* of the action (Section 2.1). The *grammar induction* module (Section 2.3) uses the output of the word finding module and the constraints of the frame description of the action to build a user-specific grammar. During decoding, the semantics module combines the output of the newly trained word finding and grammar modules to map new commands to their associated controls.

The ultimate goal is to integrate the training and decoding phase. The user uses ALADIN to control a device, but if the action that was performed is incorrect, the user will instead use (non-speech) device controls to execute the desired action. The use of the device controls will have two additional effects: first, if possible the incorrect action performed by ALADIN will be undone, and second, the action indicated by the device controls will serve as supervision information to train or update the ALADIN system.

2.1. Frame Description

In order to provide a common framework for all possible actions we wish to distinguish, we adopt the use of *frames*, previously also successfully deployed in command and control (henceforth C&C) applications and spoken dialog systems [1]. Each action that can be performed with a device is represented in the form of a *frame*, a data structure that represents the semantic concepts that are relevant to the execution of the action and which users of the C&C application are likely to refer to in their commands. It usually contains one or multiple slots, each associated with a single value.

We illustrate our frame-based representation with an example from one the target applications in the ALADIN project: a voice-controlled version of the card game *patience*. In this game, one of the possible actions is moving a card in the playing field. This action is described by an action frame dubbed `movecard`, which contains slots specifying which card is moved and to which position it is moved. Fig. 2 shows an example of such a move, and the automatically generated action frame description of that move.

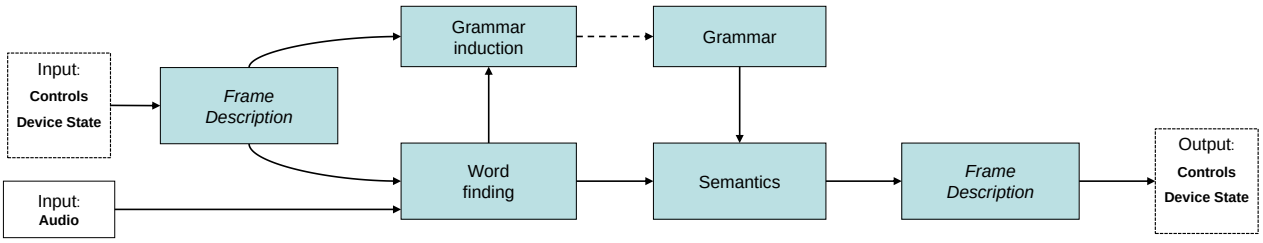
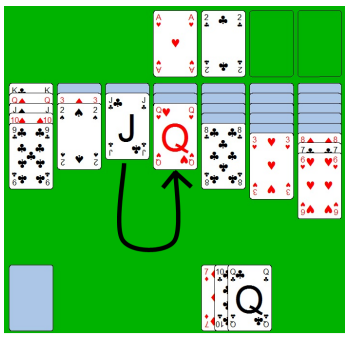


Figure 1: Schematic overview of the ALADIN framework.



Frame Slot	Value
<FS>	c
<FV>	11
<FC>	3
<FH>	–
<TS>	h
<TV>	12
<TF>	–
<TC>	4

Figure 2: An example of a patience move and the automatically generated `movecard` action frame. A move is defined as a combination of a from slot (<F?>) and a to slot (<T?>). A card is defined as the combination of a suit (<?S>) - (h)earts, (d)iamonds, (c)lubs or (s)pades - and a value (<?V>), from ace (1) to king (13). We also distinguish slots for the 'hand' at the bottom (<FH>), the seven columns (<?C>) in the playing field and the four foundation stacks at the top right (<TF>).

2.2. Word finding

The word finding module is tasked with creating acoustic representations of recurring acoustic patterns, guided by frame descriptions of utterances. As such, the learning task is only weakly supervised: rather than having knowledge of the sequence of words that were spoken, as common in Automatic Speech Recognition, we only have knowledge of the frame slot-value pairs that were referred to.

The word finding module employs supervised non-negative matrix factorization (NMF). For a detailed description we refer the reader to [2] and the references therein. In a nutshell, during training we find matrices $\mathbf{W}$ and $\mathbf{H}$ such that:

$$\mathbf{V} = \begin{bmatrix} \mathbf{V_0} \\ \mathbf{V_1} \end{bmatrix} \approx \begin{bmatrix} \mathbf{W_0} \\ \mathbf{W_1} \end{bmatrix} \mathbf{H} = \mathbf{WH} \qquad (1)$$

with the columns of the matrix $\mathbf{V_1}$ containing acoustic representations of the spoken commands. As in [2], the acoustic features are formed by first applying vector quantization (VQ) to MFCC features, and then constructing a histogram of the co-occurrences of these VQ labels. The columns of $\mathbf{V_0}$ associate each spoken command with a label vector that represents the frequency with which a particular label occurred in that spoken command. After training, the matrix $\mathbf{W_1}$ contains R acoustic patterns as its columns, and the matrix $\mathbf{H}$ indicates the weights with which these R acoustic patterns are linearly combined to form each spoken command. The columns of the matrix $\mathbf{W_0}$ describe the mapping between the R acoustic patterns in $\mathbf{W_1}$ and the labels associated with each spoken command.

To decode a spoken command, we find a vector $\mathbf{h}$ for which holds: $\mathbf{v_1}^{tst} = \mathbf{W_1 h}^{tst}$, with $\mathbf{W_1}$ the matrix found during training. $\mathbf{v}_1^{tst}$ is an acoustic representation of the spoken command, and $\mathbf{h}^{tst}$ is the R-dimensional vector that indicates which acoustic patterns in $\mathbf{W_1}$ need to be linearly combined to explain $\mathbf{v}_1^{tst}$. We calculate a score representing the label association with the spoken command using: $\mathbf{a} = \mathbf{W_0 h}^{tst}$.

In our framework, we consider each unique slot-value pair of each frame as a single label. This way, each frame description is uniquely mapped to a binary vector $\mathbf{v_1}$, and likewise, the decoded label vector $\mathbf{a}$ is uniquely mapped back to a frame description.

Put	*the*	*jack*	*of*	*clubs*	*on*	*the*	*queen*	*of*	*hearts*
O	O	I_FV	O	I_FS	O	O	I_TV	O	I_TS

Figure 3: Example of a command transcription, annotated with concept tags.

2.3. Grammar induction

The task of the grammar induction module is to automatically induce a user-specific grammar during the training phase, that detects the compositionality of the utterances and relates it to the associated meaning. In this case, the grammatical properties of the utterances are associated with action frames, containing slots and values. This grammar induction is performed on the basis of the output of the word finding module (hypothesized 'word' units, represented as acoustic patterns and possibly associated frame slot values) and the frame descriptions of the actions. The grammar aids the decoding process by providing information regarding the probability of specific frame slot sequences in the data.

There are different options with respect to the type of grammar that can be induced. Encouraging results have been reported in the unsupervised induction of sequence tags [3]. In the context of the ALADIN project, we therefore decided to adopt a *concept tagging* approach as a *shallow grammar* interface between utterance and meaning. In this vein, each command is segmented into chunks of words, which are tagged with the semantic concepts (i.e. frame slots) to which they refer.

We use a tagging framework which is based on so-called `IOB` tagging, commonly used in the context of phrase chunking tasks [4]. Words inside a chunk are labeled with a tag starting with `I` and words outside the chunks are labeled with an `O` tag, which means that they do not refer to any concept in the action frame. Fig. 3 illustrates the concept tagging approach for an example command for the action displayed in Fig. 2.

3. Experimental setup

The experiments described in this paper pertain to a vocal interface for the card game of patience. This presents an appropriate case study, since a C&C interface for this game needs to learn a non-trivial, but fairly restrictive vocabulary and grammar. In practice, most applications targeted in the ALADIN project will

have a lower complexity, for example in the case of domotica. We collected a corpus of more than two thousand spoken patience commands in (Belgian) Dutch[2], produced by eight participants during demonstration games. These command were manually transcribed and annotated with the aforementioned *concept tags* and associated with the automatically generated action frames. There are two types of frames: `movecard` frames (cf. Fig. 2) and a slotless `dealcard` frame that simply triggers a new hand. For more information on the constitution of this corpus and the frame representations, we kindly refer the reader to [5].

The experimental setup mimics the ALADIN learning situation as much as possible. The number of utterances used to train the system, should be as small as possible, i.e. the training phase should be as brief as possible in order to limit the amount of extraneous physical work or assistance needed for training by the physically impaired person. In order to get an idea of the minimal number of training utterances needed to enable successful processing, we evaluated the techniques with increasing amounts of training data, resulting in learning curves.

A separate learning curve is made for each participant individually, since the learning process in the targeted ALADIN application will be personalized as well. For each learning curve, the last fifty utterances of a participant are used as a constant test set, while the first *k* utterances are used as training data, *k* being an increasing number of slices of ten utterances for the grammar induction experiments and 25 utterances for the word finding experiments. The chronological order of the commands, as uttered by the participant, was preserved, in order to account for the development of the users' command structure and vocabulary use during the games.

3.1. Word finding

We use the automatically generated action frames to provide supervision. In the NMF learning framework, two acoustic representations were assigned to each label, with an additional 15 representations used to model filler words: $R = 2 \times 63 + 15 = 141$. Other experimental parameters were adopted from the setup described in [2].

For this experiment, frame decoding is guided by a hand-crafted grammar, rather than an automatically induced grammar. We defined 38 grammar rules corresponding to various possible slot sequences, under the assumption that `<F?>` slots come before `<T?>` slots, and that `<?S>` slots come before `<?V>` slots. These 38 rules also include various slot sequences in which the command was underspecified. A pilot experiment showed that this grammar covers 98% to 100% of the spoken commands, depending on the speaker.

The hand-crafted grammar was implemented as a slot/value-to-slot/value bigram transition matrix, and Viterbi decoding was used to generate a possible frame description for each grammar. For scoring, we generated an N-best list (N=5) of most likely frame descriptions. The use of an N-best list mimics a scenario in which the patience game software can select the most likely action, based on its knowledge of the playing field and the rules of the game.

3.2. Grammar induction

The exploratory experiments for the grammar induction module serve as a proof-of-the-principle experiment that showcases the *learnability* of the task in optimal conditions and focuses on the minimally required amount of training data needed to bootstrap successful concept tagging for this task. In these supervised learning experiments, the annotated corpus is used as training material for a data-driven tagger, which is subsequently used to tag previously unseen data. As our tagger of choice, we opted for MBT, the memory-based tagger [6], although any type of data-driven tagger can be used.

Feature selection was performed on the basis of a development set (last 25% of the training data) and establishes the best combination of disambiguation features, such as the number of (disambiguated) concept tags to the left, the tokens themselves (left/right/focus) and ambiguous tags (focus token and right context). We also compare our results against a baseline condition, a unigram tagger, in order to see the relative effect of the use of contextual information by the tagger.

4. Results and discussion

In Fig. 4 we can observe learning curves for the word finding and the grammar induction experiments. Experimental results for the former are expressed in terms of *slot-value recall*, which we calculate for each entry in the N-best list by dividing the number of slot-value pairs correctly selected, according to the reference frame description, by the total number of slot-value pairs in the reference frame description (expressed as a percentage). The reported slot-value recall is the highest recall of the entries in the N-best list.

The metric used for the evaluation of the concept tagger is the *I-chunk recall*, which is calculated by counting the number of correctly predicted I-chunks (i.e. chunks with I tags, referring to concepts) and dividing it by the total number of I-chunks given by the annotated transcription. This means that the concept tags themselves, as well as the borders of the predicted I-chunks are included in the evaluation.

4.1. Word finding

The graph on the left hand side of Fig. 4 displays the slot-value recall scores, obtained after using all available training material. These vary between 37.5% for speaker 4 and 85.3% for speaker 1. We can also observe that overall, the results for all speakers show a fairly linear increase in accuracy as more training material becomes available. The fact that the accuracy does not level off with increasing training data, indicates that the results are likely to improve with more training data.

This is also likely given the complexity of the learning task: In patience, about half of the spoken commands pertain to `dealcard` frames, which means it is very likely some slot-value pairs have never occurred in the training data, even after 200 spoken commands. We expect, however, that we need at least a few repetitions of each slot-value pair to build a robust acoustic representation: the accuracy of correctly detecting the `dealcard` frame, which has many repetitions in the training data, is close to 100% for all speakers.

Another issue that increases the complexity of the learning task is the fact that the automatically generated frame description used for supervision are often over-specified; for example, even if the spoken command only referred to `FS` and `FV` slots, the frame descriptions will also contain the `FC` slot. As a result, the supervision information contains noisy labels, which leads to an ambiguity that can only be resolved given enough training data. Given the complexity of the learning task, the fact that we obtain accuracies up to 85.3% is very encouraging.

4.2. Grammar induction

The right part of Fig. 4 displays the learning curves for the supervised concept tagging experiments. There is a lot of variation between the participants in accuracy using the first 100 training utterances. Six out of eight curves reach 95% or more

[2]The ALADIN system is expected to be language independent. For languages in which prosody is important, it is conceivable the acoustic features will need to be augmented with for example pitch information.

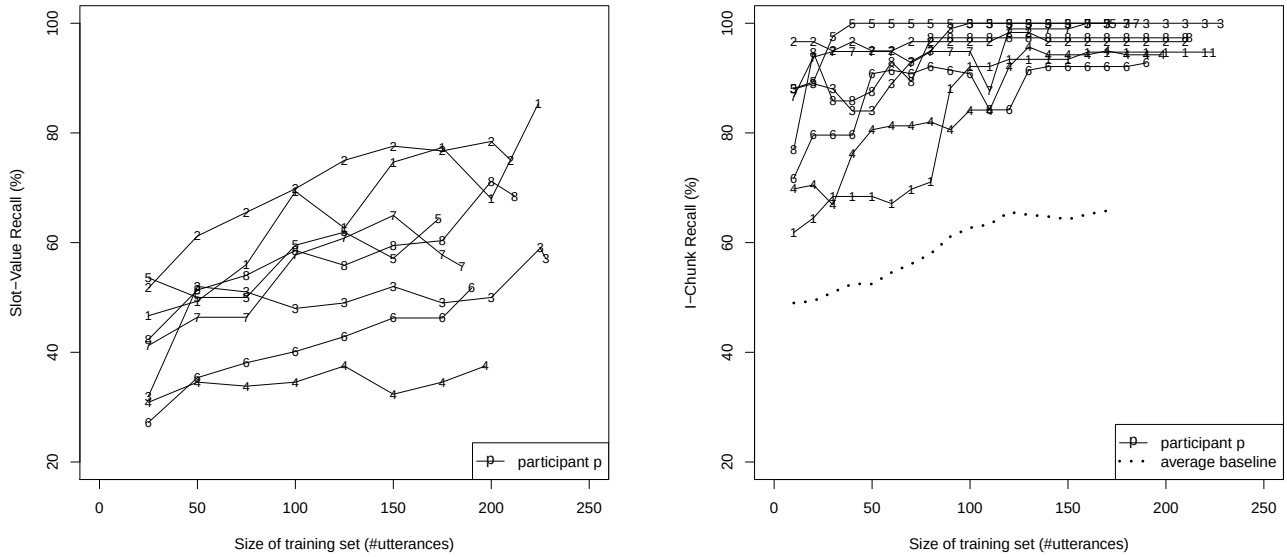


Figure 4: Learning curves viz. word finding (slot-value recall) and grammar induction (I-chunk recall).

I-chunk recall with 130 training utterances, and level off after that. For two participants, recall goes up to 100%, with training set sizes of respectively 40 and 100 utterances. The baseline accuracies, averaged over all participants, are also shown in Fig. 4. These are significantly superseded by the individual learning curves with optimized features, showing that the use of contextual information is essential for accurate tagging.

Error analysis shows that the observed differences in learning curves are related to the individual language usage. Participant 6's recall remains relatively low (around 92%) due to overly complex command structures throughout the experiments. The commands of participants 2 and 5 on other hand were structurally very consistent throughout the games, resulting in very fast learning. The learning curve of participant 5 reaches a recall of 100% using as few as forty training utterances, underlining the learnability of this task in optimal conditions.

5. Conclusions and future work

In this paper, we introduced a self-learning framework for a vocal interface that can be used with any language, dialect, vocabulary and grammar. In addition to a description of the overall learning framework and its internal knowledge representation, we described the two building blocks that will provide vocabulary learning and grammar induction. Our feasibility experiments show encouraging results, both for vocabulary learning and grammar induction, when applied to the very challenging task of a vocally guided card game, patience, with only limited training data.

We expect the word finding results to improve once speaker-specific grammars, provided by the grammar induction module, can be incorporated. The hand-crafted grammar employed in the word finding experiments, includes almost all variations, while a speaker-specific grammar will typically be more restrictive. Furthermore, we expect the number of repetitions needed for each slot-value pair to reduce substantially if we allow *sharing* of the acoustic representations between slots: For example, it is very likely that the user will refer to the suit of 'hearts' the same way, regardless of whether it occurs in a <FS> slot or in a <TS> slot.

Future experiments will also investigate how unsupervised learning techniques can be used to bootstrap concept tagging without using annotated and manually transcribed data. This will enable the output of the grammar module to replace the manually crafted grammar currently used by the word finding module. Since the learning curves for the word finding module still show significant room for improvement, more data will need to be collected to adequately investigate the interaction between the two modules and establish their integration.

6. Acknowledgments

This research was funded by IWT-SBO grant 100049 (ALADIN).

7. References

[1] Y. Wang and A. Acero, "Rapid development of spoken language understanding grammars," *Speech Communication*, vol. 48, no. 3-4, pp. 390–416, 2006.

[2] H. Van hamme, "Hac-models: a novel approach to continuous speech recognition," in *Proceedings INTERSPEECH*, 2008, pp. 2554–2557.

[3] R. Collobert, J. Weston, L. Bottou, M. Karlen, K. Kavukcuoglu, and P. Kuksa, "Natural language processing (almost) from scratch," *Journal of Machine Learning Research*, vol. 12, pp. 2461–2505, 2011.

[4] L. Ramshaw and M. Marcus, "Text chunking using transformation-based learning," in *Proceedings of the Third ACL Workshop on Very Large Corpora*, 1995, pp. 82–94.

[5] J. van de Loo, G. De Pauw, J. Gemmeke, P. Karsmakers, B. Van Den Broeck, W. Daelemans, and H. Van hamme, "Towards shallow grammar induction for an adaptive assistive vocal interface: a concept tagging approach," in *Proceedings NLP4ITA*, 2012, pp. 27–34.

[6] W. Daelemans, J. Zavrel, A. van den Bosch, and K. Van der Sloot, "MBT: Memory-based tagger, version 3.2, reference guide," University of Tilburg, Tech. Rep. 10-04, 2010.