Towards a Self-Learning Assistive Vocal Interface: Vocabulary and Grammar Learning

Janneke van de Loo[1], **Jort F. Gemmeke**[2], **Guy De Pauw**[1]
Joris Driesen[2], **Hugo Van hamme**[2], **Walter Daelemans**[1]
[1]CLiPS - Computational Linguistics, University of Antwerp, Antwerp, Belgium
[2]ESAT - PSI Speech Group, KU Leuven, Leuven, Belgium
janneke.vandeloo@ua.ac.be, jort.gemmeke@esat.kuleuven.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. This 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.

1 Introduction

Voice control of devices we use in our daily lives is still perceived as a luxury, since often cheaper and more straightforward alternatives are available, such as pushing a button or using remote controls. But what if pushing buttons is not trivial? 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 (Noyes and Frankish, 1992). By regaining the ability to control more 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 still lacks robustness to speaking style, regional accents and noise, so that users are typically forced to adhere to a restrictive grammar and vocabulary in order to successfully *command and control* a device.

In this paper we describe 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 vocal interface 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. Users should formulate commands as they like, using the words and grammatical constructs they like and only addressing the functionality they are interested in.

We distinguish two separate modules that establish self-learning: The **word finding** module works on the acoustic level and attempts to automatically induce the vocabulary of the user during training, by associating recurring acoustic patterns (commands) with observed changes in the user's environment (control). The **grammar induction** module works alongside the word finding module to automatically detect the compositionality of the user's utterances, further enabling the user to freely express commands in their own words.

This paper presents a functional description of the ALADIN learning framework and describes feasibility experiments with the 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 briefly describe the *Patience* corpus used

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

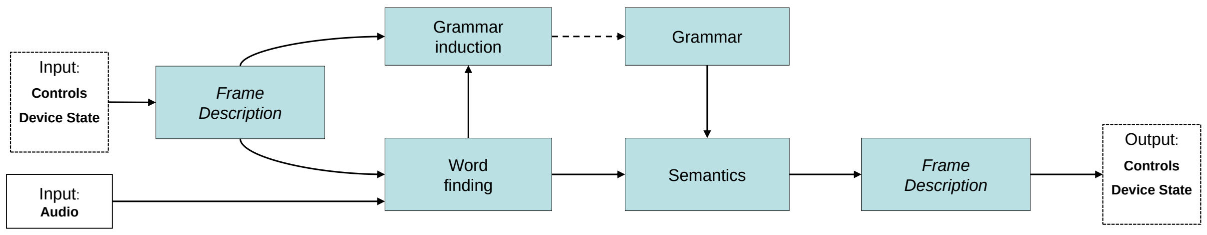


Figure 1: Schematic overview of the ALADIN framework.

in the feasibility experiments, as well as the experimental setup. In Section 4 we show and discuss our experimental results and we present our conclusions and thoughts on future work in Section 5.

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).

In order to provide a common framework for all possible actions we wish to distinguish, we adopt the use of *frames*, a data structure that encapsulates the control inputs and/or device states relevant to the execution of each action. Frames consist of one or multiple slots, which each can take a single value from a set of predefined values. In Section 2.1 we discuss the frame representation in detail.

During training, the *word finding* module builds acoustic representations of recurring acoustic patterns, given a (small) set of training commands, each described by a frame description and features extracted from the audio signal. Using the frame description, the module maps such acoustic representations to each slot-value pair in each frame. When using the framework for decoding spoken commands, the output of the module is a score for each slot-value pair in each frame, representing the probability that this slot-value pair was present in the spoken command.

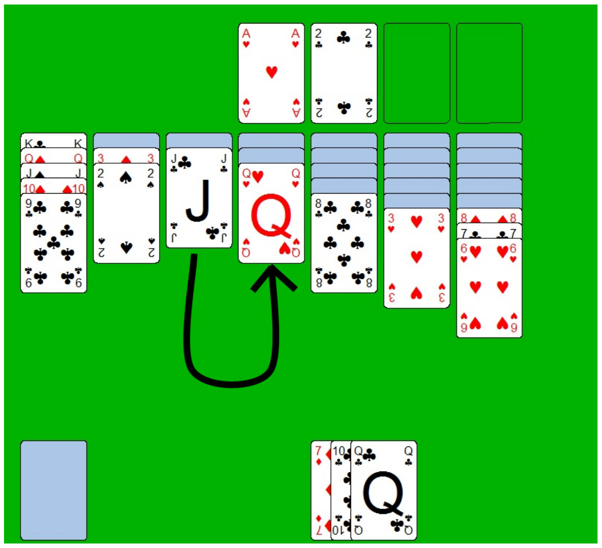
During training, the *grammar induction* module builds a model of the grammatical constructs employed by the user, using the frame description and the output of the word finding module. The output of the word finding module consists of estimates of the slot-value pair scores described above, based on the presence of automatically derived recurring acoustic patterns.

The *semantics* module, operational during decoding, processes the output of the word finding module to create a single frame description most likely to match the spoken command. This can then be converted to a control representation the target device can work with. The module can make use of a *grammar* module that describes which slot-value pair combinations (and sequences) are likely to occur for each frame. Such a grammar description should ideally be provided by the grammar induction module, but could optionally be hand-crafted.

2.1 Frame description

Each action that can be performed with a device is represented in the form of a *frame*. A frame is a data structure that represents the semantic concepts that are relevant to the execution of the action and which users of the command and control (henceforth C&C) application are likely to refer to in their commands. It usually contains one or multiple slots, each associated with a single value. The slots in an action frame represent relevant properties of the action. Such frame-based semantic representations have previously been successfully deployed in C&C applications and spoken dialog systems (Wang et al., 2005).

For our research, we distinguish three types of frames. The first, the *action frame*, is automatically generated during training by the device that is controlled with a conventional control method, such as button presses. Depending on the frame, more slots may be defined than are likely to be referred to in any single command. The second frame type, the *oracle*



Frame Slot	Value
`<from_suit>`	`c`
`<from_value>`	`11`
`<from_column>`	`3`
`<from_hand>`	`-`
`<to_suit>`	`h`
`<to_value>`	`12`
`<to_foundation>`	`-`
`<to_column>`	`4`

Figure 2: An example of a Patience move and the automatically generated `movecard` action frame. A card is defined as the combination of a `suit` - (h)earts, (d)iamonds, (c)lubs or (s)pades - and a `value`, from ace (1) to king (13). We also distinguish slots for the 'hand' at the bottom, the seven columns in the center of the playing field and the four foundation stacks at the top right.

action frame, is a manually constructed subset of the action frame based on a transcription of the spoken command. In this subset, only those slots that are referred to in the spoken command, are filled in. Finally, we define the *oracle command frame*, which is a version of the oracle action frame that can assign multiple values to each slot in order to deal with possible ambiguities in the spoken command.

We will illustrate these frame types 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.

For instance, if the move in Fig. 2 was associated with the spoken command "*put the jack of clubs on the red queen*", the oracle action frame of that particular move would only have the following slot values filled in: `<from_suit>=c`, `<from_value>=11`, `<to_suit>=h` and `<to_value>=12`, since the columns are not referred to in the spoken command. Also, since no slot was defined that is associated with the *color* of the card, the spoken command is ambiguous and during decoding, such a command might also be associated with a frame containing the slot-value pair `<to_suit>=d`. As a result, the oracle command frame will be constructed with `<to_suit>=h,d` rather than `<to_suit>=h`.

2.2 Word finding

The word finding module is tasked with creating acoustic representations of recurring acoustic patterns, guided by action frames. 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 (ASR), we only have knowledge of the slot-value pairs in the action frame, each of which may have been referred to in the utterance with one or multiple words, and in any order. To meet these requirements, we turn to a technique called non-negative matrix factorization (NMF).

2.2.1 Supervised NMF

NMF is an algorithm that factorizes a non-negative $M \times N$ matrix $\mathbf{V}$ into a non-negative $M \times R$ matrix $\mathbf{W}$ and a non-negative $R \times N$ matrix $\mathbf{H}$: $\mathbf{V} \approx \mathbf{W} \cdot \mathbf{H}$. In our approach, we construct the NMF problem as follows:

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

with the matrix $\mathbf{V_1}$ composed of N spoken commands, each represented by a vectorial representation of dimension M_1. The columns of $\mathbf{V_0}$ associate each spoken command with a label vector of dimension M_0 that represents the frequency with which a particular label occurred in that spoken command. After factorization, the matrix $\mathbf{W_1}$ contains R acoustic patterns of dimension M_1, and the matrix $\mathbf{H}$ indicates the weights with which these R acoustic patterns are linearly combined for each spoken command n, $1 \leq n \leq N$, to form the observed spoken commands in $\mathbf{V_1}$. The columns of the matrix $\mathbf{W_0}$ describe the mapping between the R acoustic patterns in $\mathbf{W_1}$ and the M_0 labels that can be associated with each spoken command. In addition to columns of $\mathbf{W_1}$ associated with labels, we use a number of so-called 'garbage columns' to capture acoustic representations not associated with labels, for example to capture less meaningful words such as 'please'.

To decode a spoken command (the 'testing' phase), we find a vector $\mathbf{h}$ for which holds: $\mathbf{v_1}^{tst} = \mathbf{W_1}\mathbf{h}^{tst}$, with $\mathbf{W_1}$ the matrix found during training. $\mathbf{v}_1^{tst}$ is the M_1 dimensional acoustic representation of the spoken command we wish to decode, 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 $\mathbf{v}_1^{tst}$. Finally, we calculate the label association with the spoken command $\mathbf{v}_1^{tst}$ using: $\mathbf{a} = \mathbf{W_0}\mathbf{h}^{tst}$, where $\mathbf{a}$ is a M_0 dimensional vector giving a score for each label.

For more details on how to carry out these factorizations, we refer the reader to Lee and Seung (1999). For a discussion on representing spoken commands of varying length as a M_1-dimensional vector, and the constraints under which it holds that the spoken command is the linear combination of R such vectors from $\mathbf{W_1}$, we refer the reader to (Van hamme, 2008; Driesen and Van hamme, 2012; Driesen et al., 2012) and the references therein.

2.2.2 Frame decoding

In our framework, we consider each unique slot-value pair of each frame (for example <to_suit>=h of the frame movecard) as a single label, making the total number of labels M_0 equal to the cumulative number of different values in all slots in all frames. 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 module is to automatically induce a 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 generated frame descriptions of the actions. Furthermore, the grammar may also serve as an additional aid during 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. It could for instance be a traditional context-free grammar, meaning that the contents of the frame description of the action are derived on the basis of a parse tree of the utterance. Unfortunately, context-free grammars have been proven to be very hard to automatically induce (de Marcken, 1999; Klein, 2005), particularly on the basis of limited training data.

Encouraging results have been reported in the unsupervised induction of sequence tags (Collobert et al., 2011). 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 (Ramshaw and Marcus, 1995). 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.

3 Experimental setup

The experiments described in this paper pertain to a vocal interface for the card game 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. Commands such as "*put the four of clubs on the five of hearts*" or "*put the three of hearts in column four*" are not replaceable by holistic commands, and identifying the individual components of the utterance and their interrelation is essential for the derivation of its meaning. This makes the Patience game a more interesting test case than domotica applications such as controlling lights, doors or a television, where the collection of unordered sets of keywords is usually sufficient to understand the commands.

In this section, we will describe the corpus collected to enable this case study, as well as the setup for exploratory experiments with the techniques outlined in Section 2.

3.1 Patience corpus

The Patience corpus consists of more than two thousand spoken commands in (Belgian) Dutch[2], transcribed and manually annotated with *concept tags*. Eight participants were asked to play Patience on a computer using spoken commands, which were subsequently executed by the experimenter. The participants were told to advance the game by using their own commands freely, in terms of vocabulary and grammatical constructs. The audio signals of the commands were recorded and the associated actions were stored in the form of action frames. There are two types of frames: a `movecard` frame, describing the movement of a card on the playing field (e.g. Fig. 2), and a `dealcard` frame that contains no frame slots, but simply triggers a new hand. Oracle action and command frames were derived on the basis of the automatically generated action frames and the manually annotated concept tags.

Each participant played in two separate sessions, with at least three weeks in between, so as to capture potential variation in command use over time. The participants' ages range between 22 and 73 and we balanced for gender and education level. We collected between 223 and 278 commands (in four to six games) per participant. The total number of collected commands is 2020, which means an average of 253 commands per participant and the average number of moves per game is 55. The total number of frame slot-value pairs is 63.

The experimental setup tries to mimic the ALADIN learning situation as much as possible. For each participant, a separate learning curve was made, 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 were used as a constant test set. The remaining utterances of the same participant were used as training material. The chronological order of the commands, as they were uttered by the participant, was preserved, in order to account for the development of the users' command structure and vocabulary use during the games. In each experiment, the first *k* utterances were 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.

3.2 Word finding

Spoken commands are represented by a Histogram of Acoustic Co-occurrence (HAC) features (Van hamme, 2008), constructed as follows: First, we extract mel-cepstral coefficients (MFCC) from audio signals sampled at 16kHz, framed using time windows of 25ms and shifted in increments of 10ms. From each of these frames, 13 cepstral coefficients, along with their first and second order differences are determined, yielding a 39 dimensional feature vector. Mean and variance normalization are applied on a per-utterance basis. Second, k-means clustering of 50000 randomly selected frames is used to create a Vector Quantization codebook with 200 codewords for each speaker, using k-means clustering. Finally, three sets of HAC features are constructed by counting the co-occurrences of the audio expressed as VQ codewords, with time lags of 2, 5 and 9 frames. The final feature dimension M_1 is thus M_1=3 $\times$ $200^2 = 120000$.

In these initial experiments, we use the oracle action frames to provide supervision. In the NMF learning framework, two acoustic representations were assigned to each label, with an additional 15 representations used as garbage columns. The total number of acoustic representations R is thus

[2]Note however that the ALADIN system is inherently language independent, which is why we present the examples in English.

$R = 2 \times 63 + 15 = 141$. For training, $\mathbf{W_1}$ is initialized randomly and $\mathbf{W_0}$ is initialized so that two columns are mainly associated with each label (i.e., a one in the corresponding label position and a small ($[0, 1e-5]$) random value for the other labels). The remaining 15 garbage columns are randomly initialized. Finally, the entries of $\mathbf{V_1}$ and $\mathbf{V_0}$ are scaled so their cumulative weight is equal. During training, the rows of $\mathbf{H}$ pertaining to non-garbage columns in $\mathbf{W_0}$ are initialized to be the same as $\mathbf{V_0}$, with a small ($[0, 1e-5]$) random value replacing values that are zero. The rows of $\mathbf{H}$ pertaining to garbage columns are initialized randomly. For the NMF factorization, we minimized the Kullback-Leibler divergence using 100 iterations of the procedure described in Lee and Seung (1999).

In these experiments, 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 `from` slots precede `to` slots, and that `_suit` slots precede `_value 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 labelvector-to-labelvector bigram transition matrix, and Viterbi decoding was used to generate a possible frame description for each grammar rule. For scoring, the most likely frame description was selected based on the most likely Viterbi path across grammar rules. Finally, we express results in terms of *slot-value accuracy*, which is the ratio of the number of slot-value pairs correctly selected, according to the oracle command frame, and the total number of slot-value pairs in the oracle command frame (expressed as a percentage).

3.3 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. 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 (Daelemans et al., 2010), although any type of data-driven tagger can be used.

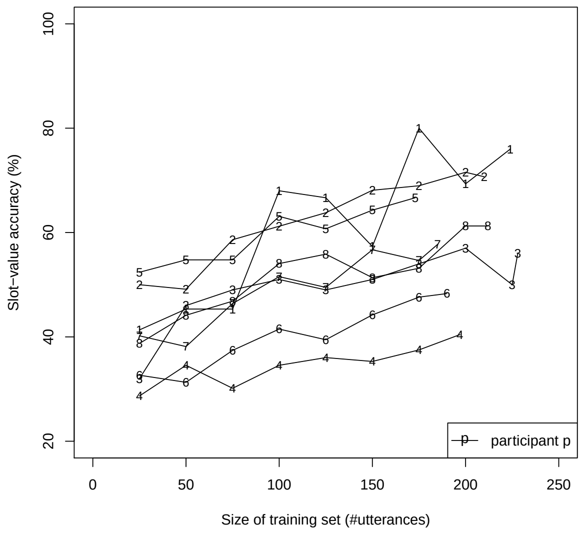
In the targeted ALADIN application, the number of utterances used to train the system should be as small as possible, i.e. the training phase should be as brief as possible in order to limit the amount of extraneous physical work or assistance needed for training by the physically impaired person. In order to get an idea of the minimal number of training utterances needed to enable successful concept tagging, we evaluated the supervised tagging performance with increasing amounts of training data, resulting in learning curves.

The metric used for the evaluation of the concept tagger is the micro-averaged F-score of the predicted `I` chunks: the harmonic mean of the precision and recall of the chunks with `I` labels (i.e. referring to slots in de frame description). This means that the concept tags as well as the boundaries of the predicted chunks are included in the evaluation. 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 compare our results against a baseline condition, in which only the focus word is used as a feature, in order to see the relative effect of the use of context information by the tagger.

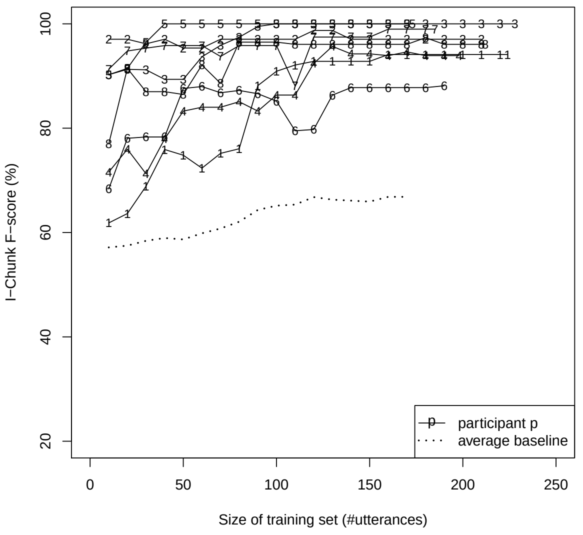
4 Results and discussion

4.1 Word finding

In Fig. 4a we can observe the results obtained with a learning framework that combines word finding with hand-crafted grammars. From these results, we can observe that the slot-value accuracy obtained after using all available training material, varies between 40.4% for speaker 4 and 76.0% 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 we do not yet observe that the accuracy levels off with increasing training data, indicates that the results are likely to further improve with more training data.



(a) Word finding results



(b) Grammar induction results

Figure 4: Learning curves viz. word finding accuracy (left) and grammar induction I chunk F-score (right).

This is also likely given the complexity of the learning task: In Patience, about half of the spoken commands pertains to `dealcard` frames, which means it is very likely that some slot-value pairs have never even 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. Given such data scarcity, the fact that we obtain accuracies up to 76% is encouraging.

Another observation that can be made is that for some speakers, such as speaker 1, there is a larger variation between consecutive training sizes - for example for speaker 1 the best accuracy is obtained for a training size of 175 spoken commands. There are several possible reasons. For one, even though the NMF learning problem is initialized using the constraints imposed by the frame labeling, the factorization process may not achieve the global optimal solution during training. This could be addressed by performing multiple experiments with different random initializations (Driesen et al., 2012).

Another issue is that the number of `dealcard` frames varies between speakers, due to the relatively small test set size of fifty spoken utterances. With the `dealcard` typically recognized correctly, this may account both for some of the differences between speakers, as well as for the variation between training sizes observed for some speakers: If the number of `movecard` frames in the test set is small, this makes the average accuracy more sensitive to errors on these frames. This issue could be addressed by an alternative evaluation scheme in which multiple occurrences of the same utterance are only counted once.

4.2 Grammar induction

Fig. 4b displays the learning curves for the supervised concept tagging experiments. There is a large amount of variation between the participants in accuracy using the first 100 training utterances. Six out of eight curves reach 95% or more with 130 training utterances, and level off after that. For two participants, the accuracies reach 100%, with training set sizes of 40 and 100 utterances respectively. The baseline accuracies, averaged across all participants, are also shown in Fig. 4b. These are significantly superseded by the individual learning curves with optimized features, showing that the use of context information is important to enable successful concept tagging on this dataset.

The fact that the tag accuracy for participant 6 remains relatively low (around 88%) is mainly due to a rather high level of inconsistency and ambiguity in the command structures that were used. One remarkable source of errors in this case is a structure repeatedly occurring in the test set and occurring only twice in the largest training set. It is a particularly difficult one: a structure in which multiple cards are specified to be moved (in one pile), such as in "*the black three, the red four and the black five to the red six*". In such cases, only the highest card of the moved pile (*black five* in the example) should be labeled with `I_FS` and `I_FV` tags (since only that card is represented in the action frame) and the lower cards should be tagged with `O` tags.

The commands given by participants 2 and 5 were structurally very consistent throughout the games, resulting in very fast learning. Participant 5's learning curve reaches a tag accuracy of 100% using as little as forty training utterances, underlining the learnability of this task in optimal conditions. Participant 3's curve reaches 100% accuracy, but has a dip at the beginning of the curve. This is due to the fact that in the utterance numbers 20-50, the suit specification was often dropped (e.g. "*the three on the four*"), whereas in the utterances before and after that, the suit specification was often included.

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 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.

Although the word finding experiments use the oracle action frames rather than the automatically generated frames as supervision information, the preliminary experiments shown in this work are promising enough to have confidence that even with this additional source of uncertainty, the goal of a self-learning vocal interface is feasible. The concept tagging experiments show that this type of representation is learnable in a supervised way with a high degree of accuracy on the basis of a relatively limited amount of data.

Future experiments will 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.

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 include almost all variations, while a speaker-specific grammar will typically be more restrictive. Another practical approach to improve the user experience is to have the ALADIN system produce an ordered set of several possible frame descriptions, based on the knowledge of the playing field and the rules of the game. Preliminary experiments revealed that even with a small ordered set of only five frame candidates, the slot-value accuracy of the Patience word finding experiments increased by 10% to 20% absolute. 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 `from` slot or in a `to` slot.

While the self-learning modules have not yet been integrated and while there is still ample room for improvement within each module individually, the results of the feasibility experiments described in this paper are encouraging. The insights gained from these experiments form a solid basis for further experimentation and will serve to further streamline the development of a language independent, self-learning command & control vocal interface for people with a physical impairment.

Acknowledgments

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

References

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

W. Daelemans, J. Zavrel, A. van den Bosch, and K. Van der Sloot. 2010. MBT: Memory-based tagger, version 3.2, reference guide. Technical Report 10-04, University of Tilburg.

C. de Marcken. 1999. On the unsupervised induction of phrase-structure grammars. In S. Armstrong, K. Church, P. Isabelle, S. Manzi, E. Tzoukermann, and D. Yarowsky, editors, *Natural Language Processing Using Very Large Corpora*, volume 11 of *Text, Speech and Language Technology*, pages 191–208. Kluwer Academic Publishers.

J. Driesen and H. Van hamme. 2012. Fast word acquisition in an NMF-based learning framework. In *Proceedings of the 36th International Conference on Acoustics, Speech and Signal Processing*, Kyoto, Japan.

J. Driesen, J.F. Gemmeke, and H. Van hamme. 2012. Weakly supervised keyword learning using sparse representations of speech. In *Proceedings of the 36th International Conference on Acoustics, Speech and Signal Processing*, Kyoto, Japan.

D. Klein. 2005. *The Unsupervised Learning of Natural Language Structure*. Ph.D. thesis, Stanford University.

D.D. Lee and H.S. Seung. 1999. Learning the parts of objects by nonnegative matrix factorization. *Nature*, 401:788–791.

J. Noyes and C. Frankish. 1992. Speech recognition technology for individuals with disabilities. *Augmentative and Alternative Communication*, 8(4):297–303.

L.A. Ramshaw and M.P. Marcus. 1995. Text chunking using transformation-based learning. In *Proceedings of the Third ACL Workshop on Very Large Corpora*, pages 82–94, Cambridge, USA.

H. Van hamme. 2008. Hac-models: a novel approach to continuous speech recognition. In *Proceedings International Conference on Spoken Language Processing*, pages 2554–2557, Brisbane, Australia.

Y. Wang, L. Deng, and A. Acero. 2005. An introduction to statistical spoken language understanding. *IEEE Signal Processing Magazine*, 22(5):16–31.