Towards Shallow Grammar Induction for an Adaptive Assistive Vocal Interface: a Concept Tagging Approach

Janneke van de Loo[1], Guy De Pauw[1], Jort F. Gemmeke[2], Peter Karsmakers[3], Bert Van Den Broeck[3], Walter Daelemans[1], Hugo Van hamme[2]

[1]CLiPS - Computational Linguistics
University of Antwerp
Antwerp, Belgium
janneke.vandeloo@ua.ac.be
guy.depauw@ua.ac.be
walter.daelemans@ua.ac.be

[2]ESAT - PSI Speech Group
KU Leuven
Leuven, Belgium
jort.gemmeke@esat.kuleuven.be
hugo.vanhamme@esat.kuleuven.be

[3]MOBILAB
K.H. Kempen
Geel, Belgium
peter.karsmakers@khk.be
bert.van.den.broeck@khk.be

Abstract

This paper describes research within the ALADIN project, which aims to develop an adaptive, assistive vocal interface for people with a physical impairment. One of the components in this interface is a self-learning grammar module, which maps a user's utterance to its intended meaning. This paper describes a case study of the learnability of this task on the basis of a corpus of commands for the card game *patience*. The collection, transcription and annotation of this corpus is outlined in this paper, followed by results of preliminary experiments using a shallow concept-tagging approach. Encouraging results are observed during learning curve experiments, that gauge the minimal amount of training data needed to trigger accurate concept tagging of previously unseen utterances.

1. Introduction

Voice control of devices we use in our daily lives is still science fiction: we do not talk to elevators, fridges or heaters. The main reason for this poor market penetration is that often more straightforward alternatives are available, such as pushing a button or using remote controls. Furthermore, speech recognition 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 a commercial climate that focuses on the development of plug 'n play, user-friendly devices, users are loath to adapt to their equipment by reading manuals or documentation or by following training.

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 contributes to their independence of living, their security, their quality of life, their communicative abilities and their entertainment.

The ALADIN project[1] aims to develop a command and control interface for people with a physical impairment, using technology based on *learning* and *adaptation*: the interface should **learn** what the user means with commands, which words he/she uses and what his/her 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.

The language independent ALADIN system will contain two modules that reduce the amount of linguistic adaptation required from the user:

- The **word finding** module works on the acoustic level and attempts to automatically induce the vocabulary of the user during training, by associating acoustic patterns (command) 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 describes work on a self-learning grammar module for the ALADIN interface. A grammar module for a command & control interface enables a mapping between the structural, grammatical properties of a user's utterance and the semantic content of the utterance, i.e. the intended control. Traditionally, command & control interfaces may include a context-free grammar, as illustrated in Figure 1, for the task of operating a television set. The compositionality of possible commands are strictly defined in this grammar, as well as their association with the intended controls (indicated between square brackets).

The ALADIN grammar module, however, will attempt to automatically derive the compositionality of the commands, while keeping the training phase as brief as possible. In this paper, we will outline preparatory experiments towards achieving this goal: before attempting *unsupervised* (shallow) grammar induction of ASR output, this paper will first investigate the feasibility of the induction task itself. The grammar module is investigated in isolation and under ideal circumstances, i.e. using manually transcribed and annotated data. Section 2 will describe the task at hand and the annotated corpus developed to investigate the aforemen-

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

```
<sentence>          = <volume_command> | <channel_command>
<volume_command>    = (set | change) volume [VOL] to <number>
<channel_command>   = (select | change to) channel [CH] (<number> | <name>)
<number>            = one [1] | two [2] | three [3] | four [4] | five [5]
<name>              = BBC [4] | CNN [2] | EuroSports [1]
```

Figure 1: Context-free grammar for a television command & control interface.

tioned research goals. Section 3 outlines the envisioned approach, i.e. concept tagging. The paper concludes with a discussion of the results and pointers towards future research.

2. Patience Corpus

For many command & control (henceforth C&C) domotica tasks, a grammar is not a strictly necessary commodity. It is perfectly feasible to control your television set using holistic commands for which no compositionality as defined in a grammar (cf. Figure 1) is needed. Furthermore, in case of a command such as "*turn the TV a bit louder*", even the unordered collection of the keywords in the utterance and their associated meanings is usually sufficient to *understand* the utterance and trigger the intended control.

There are however plenty of C&C applications for which knowledge of the compositionality of the utterance is needed to determine its meaning, such as voice controlled GPS systems, controlling entertainment centers and various types of gaming applications. To study the expedience, as well as the feasibility of grammar induction for a manageable, yet non-trivial C&C task, we decided on a case study for the card game of "patience".

Patience (also know as "solitaire") is one of the most well-known single player card games. The playing field (cf. Figure 2) consists of seven columns, four *foundation stacks* (top) and the remainder of the deck, called the *hand* (bottom). The object of the game is to move all the cards from the hand and the seven columns to the foundation stack, through a series of manipulations, in which consecutive cards of alternating colors can be stacked on the columns and consecutive cards of the same suit are placed on the foundation stack.

This game presents an interesting 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. In this section, we will describe the collection and annotation of a modestly sized corpus of spoken commands for the card game of patience.

2.1. Data Collection

The patience corpus consists of more than two thousand spoken commands in (Belgian) Dutch[2], transcribed and annotated with *concept tags* (cf. Section 3). During data collection, eight participants were asked to play patience on a computer using spoken commands. These commands were subsequently executed by the experimenter. The participants were told to advance the game by using their own commands freely, both in terms of vocabulary and grammatical constructs. The audio signals of the commands were recorded and the associated actions, executed by the experimenter, were stored in the form of action frames (cf. Section 2.2).

During the patience games, the experimenter executing the commands was situated in a separate room, invisible to the participant, and the participant gave commands through a headset microphone. Half of the participants were given the impression that their commands were executed by a completely automated system (Wizard-of-Oz), while the other four participants were told in advance that the commands would be executed by a person.

Both setups approximate the ALADIN C&C situation in their own way: the Wizard-of-Oz setup accounts for effects of human-machine interaction, but inclines people to adapt their commands to what they think the computer program would *understand*, whereas the non-Wizard-of-Oz setup gives people more sense of freedom in making up their own commands, but might also yield commands that people would not use when talking to a computer. We decided to use both setups, in order to obtain a wide range of possible commands. A preliminary qualitative inspection of the corpus did not uncover significant grammatical differences between the two groups of participants, however.

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.

2.2. Action Frames

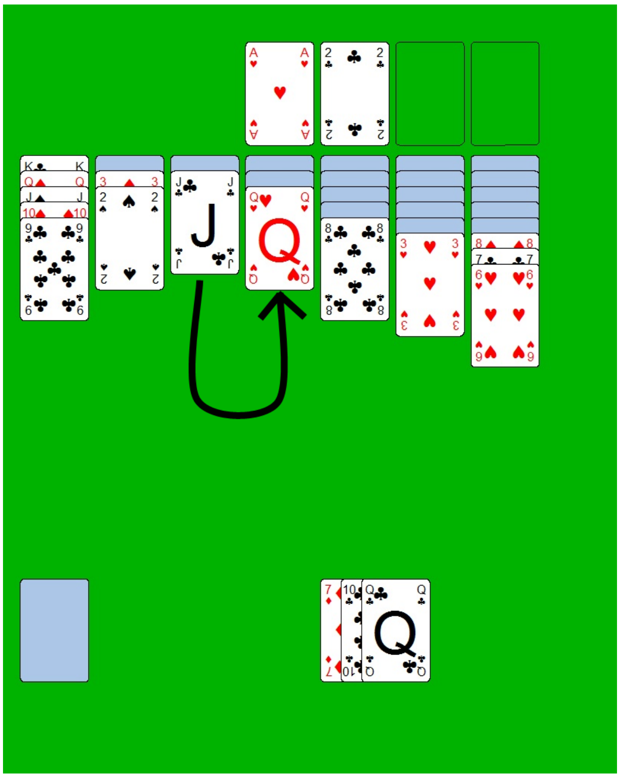
Each action in the C&C patience implementation was automatically stored in the form of an *action frame*. An action frame is a data structure that represents the semantic concepts that are relevant to the execution of the action and which users of the C&C application are likely to refer to in their commands. Such frame-based semantic representations have previously been successfully deployed in C&C applications and spoken dialog systems (Wang and Acero, 2005; Wang and Acero, 2006).

A frame usually contains one or multiple slots, associated with values. The slots in an action frame represent relevant properties of the action. The patience game has two

[2] Note however that the ALADIN system is inherently language independent.

Leg de klaveren boer op de harten vrouw

(Put the jack of clubs on the queen of hearts)



Frame Slot	Value
<from_suit>	c
<from_value>	11
<from_foundation>	–
<from_column>	3
<from_hand>	–
<to_suit>	h
<to_value>	12
<to_foundation>	–
<to_foundationempty>	–
<to_column>	4
<to_columnempty>	–

Figure 2: An example of a command, the associated action on the screen and the automatically generated movecard action frame.

types of action frames: dealcard and movecard. The former frame type does not have any slots: merely selecting this frame is sufficient for the execution of the action. The movecard frame on the other hand does have slots, specifying which card should be moved and to which position it should be moved. Figure 2 shows an example of a command, the action performed on the playing field and the frame description of that action.

Each card is defined as the combination of a suit[3] and a value[4]. The positions of the cards on the playing field are also represented by the frame description and different stacks are discerned: the hand, at the bottom, containing the visible cards which have not been played yet, the seven columns in the center of the playing field, and the four foundation stacks at the top right, where all cards should finally be moved to, ordered by suit.

Frame Slot	Value
<from_suit>	c
<from_value>	11
<to_suit>	h
<to_value>	12

Figure 3: Oracle Command Frame (movecard) for the utterance "*put the jack of clubs on the queen of hearts*".

The movecard frame has from slots, identifying the card (suit and value) that is moved and the position (the hand, a column or a foundation stack) from which it is moved, and to slots, identifying the card and position that it is moved to. If the card is moved to an empty column, the slot to_columnempty is filled with the value 1. Similarly, the slot to_foundationempty receives the value 1 when a card is moved to an empty foundation stack.

Note that the frame description in Figure 2 is over-specified with respect to the actual command. While the command may for instance only mention the cards involved in the move, column numbers are also specified in the frame description. This is due to the fact that the program generates the frame descriptions without any knowledge of the actual audio or its content. The final grammar module will therefore need to be able to not only identify the compositionality of the utterance, but also which subset of frame slots are actually mentioned by the user.

Oracle Command Frames

In the experiments we describe in this paper, we perform a reduction on the basis of *oracle command frames*. The slots of an oracle command frame typically constitute a subset of the slots of the (usually over-specified) action frame and represents the semantic concepts that are actually expressed in the command, i.e. only frame slots that the participant refers to in the command, are filled in. Figure 3 shows the oracle command frame corresponding to the command "*put the jack of clubs on the queen of hearts*".

For some commands in the patience corpus, multiple mappings to frame slot values are possible. For instance, if a participant says "*put the black king in column three*", the word "*black*" refers to two possible values for the frame slot from_suit, i.e. spades or clubs. Therefore, this command has two oracle command frames: one version with from_suit=spades and one version with from_suit=clubs. In such cases, multiple corresponding oracle command frames are added.

2.3. Transcription and Annotation

In the next phase, orthographic transcriptions of the audio commands were created manually. In addition, the transcriptions were manually annotated using a concept tagging approach. This means that each command is segmented into chunks of words, which are tagged with the semantic concepts to which they refer. The concepts are, in this case,

[3] hearts(h), diamonds(d), clubs(c) or spades(s).

[4] From ace (1) to king (13).

Tag	Corresponding Frame (Slot)
I_FS	movecard(from_suit)
I_FV	movecard(from_value)
I_FF	movecard(from_foundation)
I_FC	movecard(from_fieldcol)
I_FH	movecard(from_hand)
I_TS	movecard(target_suit)
I_TV	movecard(target_value)
I_TF	movecard(target_foundation)
I_TFE	movecard(target_foundationempty)
I_TC	movecard(target_fieldcol)
I_TCE	movecard(target_fieldcolempty)
I_DC	dealcard()
O	-

Table 1: The set of concept tags used for annotation.

Leg	**de**	**klaveren**	**boer**	**op**	**de**	**harten**	**vrouw**
Put	the	clubs	jack	on	the	hearts	queen
O	O	I_FS	I_FV	O	O	I_TS	I_TV

Figure 4: Example of a command transcription annotated with concept tags.

slots in the frame-based description of the associated action, or, if the associated action frame does not contain any slots, the complete action frame. Thus, in the context of the patience game, the set of concept tags consists of the slots in the movecard action frame, plus one concept tag for the dealcard frame.

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. In the traditional IOB tagging framework, words at the beginning of a chunk are labeled with a tag starting with B. However, these B tags are typically only useful to indicate chunk boundaries when multiple chunks of the same type are immediately adjacent to each other. This does not occur in our data, however, yielding the complete tag set, shown in Table 1. The annotation of the command of Figure 2 is illustrated in Figure 4.

A Look inside the Patience Corpus

In this subsection, we will highlight some typical features and idiosyncratic patterns that can be found in the patience corpus. Figure 5 shows the most frequently used movecard command structures. The most frequent movecard structure is a structure in which the suit and value of the from card and the to card are specified, as shown in Figure 5(a). There is a lot of lexical variation of the prepositions and the verbs which are used in this structure. In addition, the position of the verb may vary considerably. The most frequent positions are the first position (usually imperative (cf. Figure 5(a)) and the final position (in infinitive form (cf. Figure 5(b)), but it may also occur in the position following the I_FV element, as in "*de harten vijf mag op de klaveren zes*" (*the hearts five may [be put] on the clubs six*).

Most participants used a specific word or phrase to move a card to one of the foundation stacks, without specifying which stack. Two examples are shown in Figures 5(b) and 5(c). Some participants also used specific phrases to move a card to an empty foundation stack, such as in the commands shown in Figure 5(e→f). When moving a king to an empty column, most participants used the structure shown in Figure 5(d). One participant, however, used the word *"afleggen"* ("*lay down*") for this purpose, which other participants typically used to express <to_foundation>, as shown in Figure 5(c). This type of inter-speaking variation underlines the importance of the adaptability of the ALADIN approach: a flexible C&C interface should adapt to the idiosyncracies of specific users and should not predefine the vocabulary and grammar with which the device is to be manipulated.

The column numbers and foundation stack numbers were rarely specified in the commands. Some participants referred to column numbers when moving a king to an empty column. Figure 6(a) shows an example. There were also some participants, however, who referred to specific ranges of columns or foundation stacks, by using the words "*links*" (*left*) and "*rechts*" (*right*). An example is shown in Figure 6(b). In this case, the word "*links*" ambiguously refers to column numbers 1, 2 and 3.

Figure 6(b) also shows another phenomenon, which occurred frequently: the use of the word "*zwarte*" (*black*), referring to the suits clubs and spades (and, similarly, the word "*rode*" (*red*), referring to the suits hearts and diamonds). Both in the black/red situation and the left/right situation, one single word refers to a range of possible frame slot values. This means that the command expresses multiple options with respect to certain slot values: in case of the command in Figure 6(b), regarding the values of the slots <from_suit>, <from_column> and <to_suit>. As previously mentioned, this means that the command yields multiple oracle command frames (Section 2.2), in which all possible combinations of values within the specified ranges are represented.

Another interesting phenomenon occurred in some cases, when a pile of multiple cards was moved from one column to another. In such cases, some participants specified all cards to be moved - an example is shown in Figure 6(c) - or the first and the last card in the pile to be moved. As shown in Figure 6(c), only the highest card in the pile is labeled with the concept tags I_FS and I_FV; the other cards are not represented in the frame description (and do not need to be).

Especially during the first few games, many participants showed some development with respect to the command structures that were used. Participants tended to shorten their commands as the games progressed, by, for instance, leaving out determiners and verbs, and sometimes even the card suits. In addition, the command structures of some of the participants gradually became more stable over the course of the games. It seems that many participants needed some time to establish the command structures that worked best for them. This type of intra-speaker variation over time

(a)	**[leg*]**	**[de]**	**harten**	**vijf**	**op**	**[de]**	**klaveren**	**zes**	
	[put*]	[the]	hearts	five	on	[the]	clubs	six	
	[O*]	[O]	I_FS	I_FV	O	[O]	I_TS	I_TV	
(b)	**[de]**	**schoppen**	**drie**	**naar**	**boven**	**[plaatsen*]**			
	[the]	spades	three	to	top	[move*]			
	[O]	I_FS	I_FV	O	I_TF	[O*]			
(c)	**[de]**	**klaveren**	**twee**	**afleggen**	**[bovenaan]**				
	[the]	clubs	two	lay-down	[at-the-top]				
	[O]	I_FS	I_FV	I_TF	[I_TF]				
(d)	**[de]**	**harten**	**koning**	**naar**	**[de]**	**lege**	**plaats**		
	[the]	hearts	king	to	[the]	empty	space		
	[O]	I_FS	I_FV	O	[O]	I_TCE	I_TCE		
(e)	**[de]**	**ruiten**	**aas**	**naar**	**het**	**groene**	**vak**		
	[the]	diamonds	ace	to	the	green	field		
	[O]	I_FS	I_FV	O	O	I_TFE	I_TFE		
(f)	**[de]**	**klaveren**	**aas**	**op**	**een**	**leeg**	**vakje**	**bovenaan**	**[leggen*]**
	[the]	clubs	ace	on	an	empty	field	at-the-top	[put*]
	[O]	I_FS	I_FV	O	O	I_TFE	I_TFE	I_TFE	[O*]

Figure 5: The most frequently used movecard command structures, ranked according to frequency of occurrence. Optional words and tags are shown in [ ]. * indicates that the position of the verb varies.

(a)	**de**	**schoppen**	**koning**	**op**	**de**	**tweede**	**plaats**		
	the	spades	king	on	the	second	position		
	O	I_FS	I_FV	O	O	I_TC	I_TC		
(b)	**de**	**zwarte**	**vier**	**links**	**naar**	**de**	**zwarte**	**drie**	
	the	black	four	on-the-left	to	the	black	three	
	O	I_FS	I_FV	I_FC	O	O	I_TS	I_TV	
(c)	**leg**	**harten**	**vier**	**en**	**schoppen**	**drie**	**op**	**klaveren**	**vijf**
	put	hearts	four	and	spades	three	on	clubs	five
	O	I_FS	I_FV	O	O	O	O	I_TS	I_TV

Figure 6: Examples of more unusual command structures.

is again an important point of reference in the context of the ALADIN approach: the system should adapt over time to changes in the user's linguistic behavior.

3. Concept Tagging: Proof-of-the-Principle Experiments

The sequence tagging approach, illustrated in Figures 4, 5 and 6, presents a decidedly different type of representation, compared to traditional context-free grammar approaches (Figure 1), since no grammar in the traditional sense of the word is being produced. In that respect, our approach also differs from recent approaches in which context-free grammars constitute at least a part of the grammar framework, such as described in Starkie (2001) and in Wang and Acero (2005; 2006). The idea behind the sequence tagging approach is in fact more akin to that coined in Hahn et al. (2008), although this research effort does not directly refer to grammar induction as such.

In terms of grammars and parsing, we might dub our concept-tagging approach *shallow grammar induction*: similar to the technique of *shallow parsing* (Ramshaw and Marcus, 1995; Daelemans et al., 1999; Tjong Kim Sang, 2002), we speculate that we do not need to construct a complete parse tree to enable successful processing of the data, but rather that a shallow representation of the syntactic/semantic compositionality of the utterance can suffice.

Furthermore, the final grammar module in the ALADIN system will need to be able to automatically induce these concept tags. Whereas 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 (De Pauw, 2005), encouraging results have been reported in the unsupervised induction of sequence tags (Collobert et al., 2011). Furthermore, in contrast to traditional unsupervised grammar induction approaches that only work on the basis of raw data, we have additional pseudo-semantic information at our disposal in the form of action frames, that further help streamline the weakly supervised induction process.

In this section, we will describe the experimental setup for *supervised* concept tagging of the patience C&C task. These experiments serve as a proof-of-the-principle experiment that showcases the *learnability* of the task in optimal conditions, particularly in terms of the minimally required amount of training data needed to bootstrap successful concept tagging. In these 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

	Baseline		Optimized	
	Mean	SD	Mean	SD
Tag accuracy (%)	77.8	4.0	96.9	1.3
Chunk accuracy ($\mathbf{F}_{\beta=1}$)	73.8	3.4	96.5	1.3

Table 2: Ten-Fold Cross Validation: Experimental Results

tagger of choice, we opted for MBT, the memory-based tagger (Daelemans et al., 2010).

3.1. Ten-fold Cross Validation

We tested the overall generalization capability of the tagger on the patience data, by performing a ten-fold cross-validation experiment on the complete data set of 2020 utterances. Each utterance in the data set was randomly assigned to one of ten sub-samples. Ten experiments were performed, each time using a different sub-sample as the evaluation set, with the remaining nine folds as the training set, including one development set to perform feature optimization. This means that the system is being trained and evaluated on utterances from different users.

The metrics used for the evaluation of the concept tagging performance are the tag accuracy and the chunk accuracy. The tag accuracy is the ratio of the number of correctly predicted tags; the chunk accuracy is the F-score for correctly predicted chunks, which means that the concept tags, as well as the borders of the predicted chunks are included in the evaluation. The accuracies with an optimized set of features[5] were compared to the accuracies in a baseline condition, in which only the focus word was used as a feature (and thus no context information was used).

The mean tag and chunk accuracies in the ten-fold cross-validation experiments are shown in Table 2. The mean tag accuracy with the optimized feature set is 96.9%, and the mean chunk accuracy is a bit lower at 96.5%. In the baseline condition, the mean tag accuracy is 77.8% and the mean chunk accuracy is 73.8%. The relatively large gap between the tag and chunk accuracies in the baseline condition is probably caused by the lack of coherence in that condition. Since no context features were used for tagging, chunk accuracies are lower.

3.2. Learning Curves

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.

In the learning curve experiments, we tried 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 processes regarding 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. The feature set used by the tagger, was optimized in advance by means of a development set, consisting of the last 25% of the training data.

Figure 7 displays the learning curves for tag accuracies and chunk accuracies. There is a lot of variation between the participants in accuracy using the first 100 training utterances. For all participants, except participant 6, the tag accuracy reaches 95% or more with 130 training utterances, and levels off after that. The chunk accuracies tend to be slightly lower, but six out of eight curves still reach at least 95% chunk accuracy around 130 utterances. For two participants, the accuracies go up to 100%, with training set sizes of respectively 40 and 100 utterances. The baseline accuracies, averaged over all participants, are also shown in Figure 7. The maximum tag and chunk accuracies reached on average in the baseline condition, are 79.8% and 75.6% , respectively (using 170 training utterances). Most individual learning curves with optimized features are far above the average baseline curve, except the trailing curve of participant 1.

The sudden leap in this curve between 80 and 90 training utterances is due to the introduction of a new utterance for the `dealcard` move by the participant after the 80th utterance. After that, the participant kept using this new utterance (consisting of two previously unseen words), and in the test data, that same utterance occurred frequently as well. Until it was encountered in the training data, this utterance could not be successfully tagged with the appropriate concepts.

The fact that the tag accuracy for participant 6 remains relatively low (maximum around 92%), 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. This is particularly difficult to learn: a structure in which multiple cards to be moved (in one pile) are specified, such as in "*de rode twee, de zwarte drie, de rode vier en de zwarte vijf naar de rode zes*" (*the red two, 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 (*red two*, *black three* and *red four*) should be tagged with `O` tags. Many errors were made in the tagging of this type of structure. An example of ambiguity in the commands is the use of the phrase "*groene vak*" (*green*

[5] Feature selection was performed on the basis of a development set. MBT can use disambiguated tags (left context), words (left/right context) and ambiguous tags (for the focus word and right context) as features. Morphological features to disambiguate unknown words were not considered, since these will not be available to the final ALADIN system either.

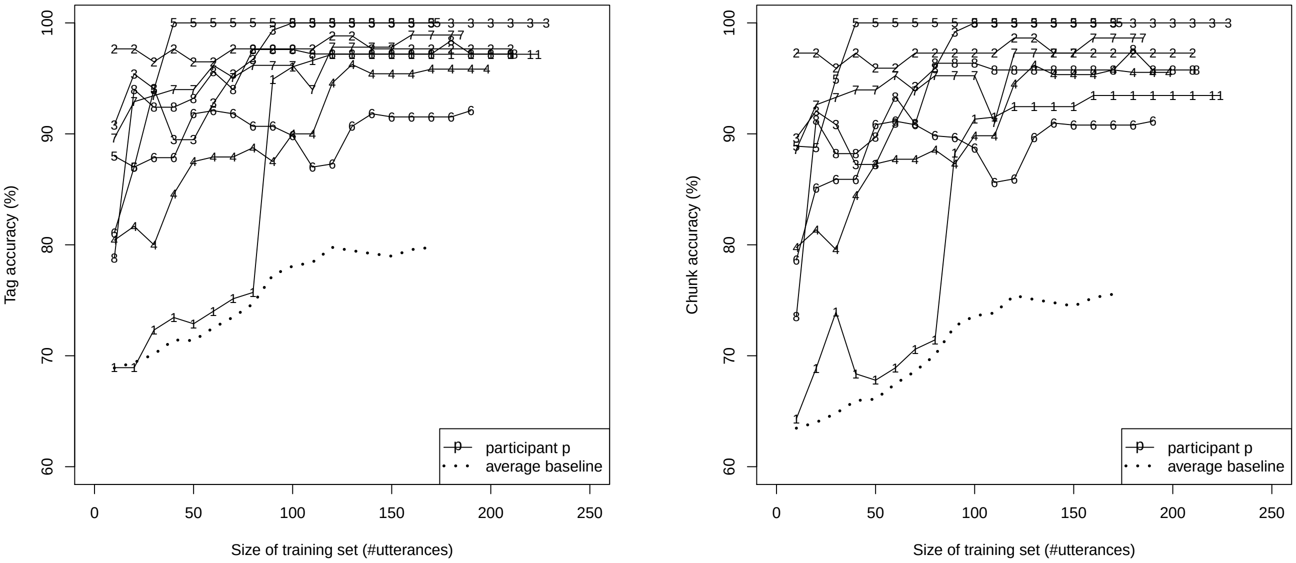


Figure 7: Learning curves viz. tag accuracy (left) and chunk accuracy (right). The solid curves show the accuracies per participant in the condition with the optimized feature set. The dotted curve shows the accuracies in the baseline condition, averaged over all participants.

field) for both an empty foundation stack and an empty column.

The commands of participants 2 and 5 were structurally very consistent throughout the games, resulting in very fast learning. The learning curve of participant 5 reaches a tag accuracy of 100% using as little as forty training utterances. The curve of participant 2 immediately starts with an extremely high accuracy of 97.7% using only ten training utterances. However, it does not reach 100%, mainly due to the presence of a restart confusing the tagger "*schoppen boer op schoppen... op... schoppen boer op harten vrouw*" (*clubs jack on clubs... on... clubs jack on hearts queen*) and one clear inconsistency: using the phrase "*naar boven*" (*up*) to move a king to an empty column, whereas this phrase was previously only used for moving a card to the foundation.

The curve of participant 3 does reach 100% accuracy, but has a remarkable dip in the beginning of the curve. This is due to the fact that in the utterance numbers 20 to 50, the specification of the suit was often dropped (e.g. "*de drie op de vier*" *the three on the four*), whereas in the utterances before and after that, the specification of the suit was often included, as well as in many of the test utterances.

3.3. Discussion

The learning curves in Figure 7 show that with around 130 training utterances, between 95% and 100% tag accuracy could be reached for all participants, except one. The chunk accuracies tend to be a bit lower, but six out of eight curves still reach between 95% and 100% chunk accuracy using around 130 training utterances. After 130 training utterances (in some cases even earlier) a plateau is usually reached, meaning that adding more utterances does not significantly improve the tagging performance any more. This implies that having a participant play around two games of patience and subsequently transcribing and annotating the utterances, would usually provide sufficient training material for training a memory-based concept tagger to tag new transcribed utterances of that same participant with reasonable accuracy. It seems that after about 100 to 130 utterances of training material, accurate execution of commands can indeed be expected.

The initial part of the learning curves, i.e. using small training sets, varies considerably among participants. In general, differences between participants regarding the individual learning curves can be attributed mainly to differences in the level of consistency and the level of ambiguity regarding the command structures and the words used. Disfluencies such as restarts have a negative effect on accuracy scores, especially those present in the test set.

The learning curves are all situated above the average baseline learning curve. This means, as expected, that in order to successfully attribute concept tags to words in patience commands, the use of the **context** of each word (not available to the unigram baseline tagger) is essential. Therefore, in the ALADIN application, a (shallow) grammar module is indeed needed in order to attribute a correct meaning to this type of commands.

The results of the ten-fold cross-validation experiments furthermore show that a memory-based concept tagger generalizes well over different sets of patience commands, with mean tag and chunk accuracies of 96.9% and 96.5%, respectively.

4. Conclusion & Future Work

This paper described a corpus of command & control utterances for the card game *patience*, as well as some preliminary experiments that gauge the feasibility of inducing this task automatically on the basis of training material. The patience corpus is a relevant case study for this type of research: while the language use is fairly constrained and the structural complexity is manageable, these utterances do require some kind of minimal detection of grammatical structure to trigger the intended controls.

The experimental results show that a *supervised* approach of concept tagging works very well for this task. The tenfold cross validation experiments show that state-of-the-art classification accuracy can be achieved on data spanning different users. The learning curve experiments performed for each user individually, mimic the intended training phase in the final ALADIN system. The experiments results show that after about 130 utterances have been processed by the system, a workable tag accuracy of 95% or more can be achieved. These results are encouraging and form a solid basis for further experimentation with unsupervised approaches and for the integration of the grammar module in a command & control domotica interface for people with a physical impairment.

In the final ALADIN system, the grammar induction module will work together with an acoustic word finding module that will identify which patterns in the acoustic signal correspond to which word candidates. These word candidates will need to trigger specific frame slots as well as their values (cf. bottom part of Figure 2). The grammar module will need to be able to deal with and help resolve ambiguities and inaccuracies of the word finding module, as it will not have access to the unambiguous identity of the words in the utterance. In our next set of experiments, we will further approach the setup of the ALADIN system, by training on indices of lattices, output by the speech recognizer, rather than on the idealized situation of using orthographically transcribed utterances.

One of the biggest challenges we have yet to tackle in this research effort is to move from the supervised approach to an *unsupervised* approach, where we will need to match tags to words (or word candidates) without reference to annotated training material. To this end, we will look into unsupervised part-of-speech tagging approaches and investigate if and how they can adapted to this particular task.

Acknowledgments

This research was funded by IWT-SBO grant 100049 (ALADIN). We would like to thank the participants in the *patience* recording sessions for their kind help.

5. References

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

Daelemans, W., Buchholz, S. & Veenstra, J. (1999). Memory-based shallow parsing. In M. Osborne & E. Tjong Kim Sang (Eds.), *Proceedings of the Conference on Computational Natural Language Learning (CoNLL-99)*. Bergen, Norway: pp. 53–60.

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

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

De Pauw, G. (2005). Agent-based unsupervised grammar induction. In M.P. Gleizes, G. Kaminka, A. Nowé, S. Ossowski, K. Tuyls & K. Verbeeck (Eds.), *Proceedings of the Third European Workshop on Multi-Agent Systems*. Brussels, Belgium, December, 2005: Koninklijke Vlaamse Academie van België voor Wetenschappen en Kunsten, pp. 114–125.

Hahn, S., Lehnen, P., Raymond, C. & Ney, H. (2008). A comparison of various methods for concept tagging for spoken language understanding. In Nicoletta . Calzolari, K. Choukri, B. Maegaard, J. Mariani, J. Odijk, S. Piperidis & D. Tapias (Eds.), *Proceedings of the Sixth International Conference on Language Resources and Evaluation (LREC'08)*. Marrakech, Morocco: European Language Resources Association (ELRA). http://www.lrec-conf.org/proceedings/lrec2008/.

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

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

Ramshaw, L.A. & Marcus, M.P. (1995). Text chunking using transformation-based learning. In *Proceedings of the Third ACL Workshop on Very Large Corpora*.

Starkie, B. (2001). Programming spoken dialogs using grammatical inference. In *Advances in Artificial Intelligence, 14th Australian Joint Conference on Artificial Intelligence*. Springer Verlag.

Tjong Kim Sang, E. (2002). Memory-based shallow parsing. *Journal of Machine Learning Research*, 2, pp. 559–594.

Wang, Y. & Acero, A. (2005). SGStudio: Rapid semantic grammar development for spoken language understanding. In *Proceedings of Ninth European Conference on Speech Communication and Technology*. Lisbon, Portugal: International Speech Communication Association.

Wang, Y. & Acero, A. (2006). Rapid development of spoken language understanding grammars. *Speech Communication*, 48(3-4), pp. 390–416.