Creativity Meets Automation: Combining Nonverbal Action Authoring with Rules and Machine Learning

Michael Kipp

DFKI, Stuhlsatzenhausweg 3, 66123 Saarbrücken, Germany
michael.kipp@dfki.de

Abstract. Providing virtual characters with natural gestures is a complex task. Even if the range of gestures is limited, deciding when to play which gesture may be considered both an engineering or an artistic task. We want to strike a balance by presenting a system where gesture selection and timing can be human authored in a script, leaving full artistic freedom to the author. However, to make authoring faster we offer a rule system that generates gestures on the basis of human authored rules. To push automation further, we show how machine learning can be utilized to suggest further rules on the basis of previously annotated scripts. Our system thus offers different degrees of automation for the author, allowing for creativity and automation to join forces.

1 Introduction

As virtual characters move toward real applications the need for tools becomes more pressing [1] [2]. Authoring tools do not only require intuitive user interfaces with a steep learning curve but also a certain amount of control to allow for rich design decisions on the part of the author.

Gesture generation is an area where automation is interesting because it is so tedious to do by hand. On the other hand, a high level of control is desirable since gestures are an integral part of what you could consider the virtual character's personality and gesture style can add a lot in terms of fun, interest and motivation to an application, even more so if multiple characters are involved. Producing interesting gestures may be more of an art than an engineering task. So, most of the time, authors have to hard-code them into the system. Although there are systems that offer rules to generate gestures [3][4], it often remains unclear how these rules are specified and how intuitive they are to use for non-technical authors. We present a framework which allows direct authoring of actions but also to define rules for automatic generation of actions and, finally, to let the system automatically learn new rules.

The gestures in our system consist of pre-fabricated keyframe animations. It appears that gesture generation calls for sophisticated skeleton-based procedural animation engines where gestural movement can be controlled and situationally adapted to a very fine degree [5] [6]. However, in many applications, even high-end games, procedural control of single bones is not done because it is too

J. Gratch et al. (Eds.): IVA 2006, LNAI 4133, pp. 230–242, 2006.

expensive in terms of development time and performance. Instead, most real-time applications rely on pre-fabricated keyframe animation and sophisticated motion blending techniques [7] [4] [8]. Gesture generation here consists of selecting a motion clip and synchronizing it with speech. Keyframe animation has the further advantage that motion capture can be used which yields a very high degree of authenticity.

This paper deals with a system where gestures can be added and generated on different levels from full authoring control to full automation (Figure 1). The COHIBIT system [12] allows the author to specify actions for two virtual characters in a screenplay-like document (level 1). In a next step, the author can write simple and intuitive gesture generation rules to exploit his or her expert knowledge for automation (level 2). Gestures generated from these rules blend seamlessly with the pre-authored ones, prioritizing the author's direct choices. In COHIBIT, before rules were introduced all actions were hand-coded (level 1) so that a large corpus of scripts with annotated actions existed. We decided to exploint this resource for machine learning. The system can learn new rules based on the annotated scripts and suggests the most appropriate ones to the author (level 3).

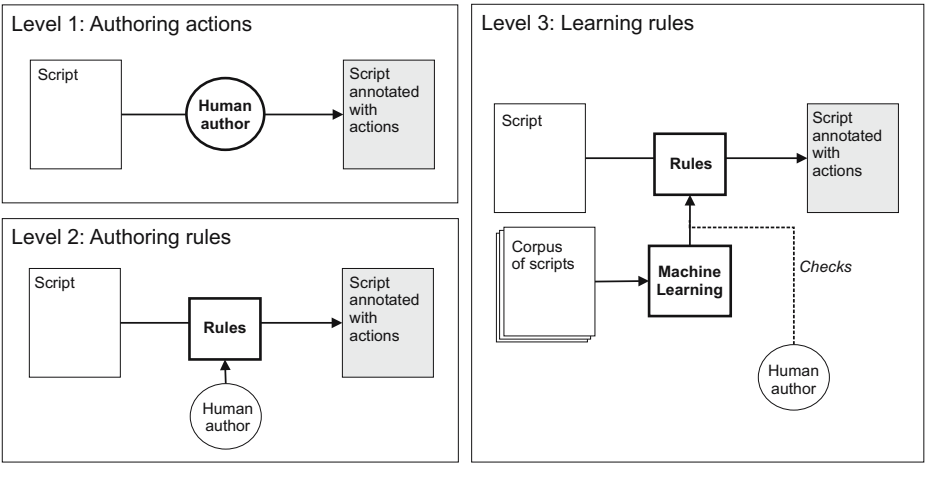

Fig. 1. Three different levels of control and automation when generating actions

So the rules of levels 2+3 automatically generate actions but many scripts already contain hard-coded actions from level 1. As we will show, all these actions can be blended by filtering and resolving conflicts. Thus, all three levels contribute to the final result: the gestural behavior of two virtual characters in the COHIBIT system which is fully implemented and has been running daily in a public exhibition space[1] since April 2006.

[1] http://vc.dfki.de

In the following sections the COHIBIT system will be presented first before proceding with how rules can be specified. Then, the automatic rule learning system will be explained. We conclude with a technical evalution and final remarks.

2 Related Work

Automating gesture generation for virtual characters is an interdisciplinary endeavor combining competences from artificial intelligence, computer animation and psychology. Cassell et al. [7] developed a rule-based system that generates audiovisual speech, intonation, facial expression, and gesture. The gesture stroke is synchronized with the accented syllable of the coexpressive word. Noma et al. [4] built the Virtual Presenter where gestures can be added to a text manually or with keyword-triggered rules. Animated gestures are synchronized with the following word. While the number of possible gestures is very small the authors focused on how to extract meaningful rules from the literature on public speaking. A more complex generation system is the Behavior Expression Animation Toolkit (BEAT) [3]. It gets plain text as input and first runs a linguistic analysis on it before generating intonation, facial animation, and gestures. Gestures are overgenerated using a knowledge base with handcrafted mappings and are then reduced by user-defined filters.

Hartmann et al. [5] achieve expressivity in gesture synthesis system by varying gesture frequency, movement amplitude and duration, fluidity, dynamic properties, and repetition. Noot/Ruttkay [9] are also deal with individual gesturing style. A style consists of meaning-to-gesture mappings, motion characteristics, and modality preferences. Combining style dictionaries yields mappings for new cultural groups or individuals.

Kopp et al. [10] [6] present a gesture animation system that makes use of neurophysiological research and generates iconic gestures from object descriptions and site plans when talking about spatial domains, e.g. giving directions. Iconics gestures resemble some semantic feature in the co-occurring speech.

In a different project [11] we have presented a system that generates gestures from statistical models of human speakers' behavior. This approach requires a lot of manual labour but yields character-specific result with the potential of imitating living people.

The approach presented in this paper resembles most closely the rule-based approaches to gesture generation. In this paper, we focus on how generation rules can be specified easily, how generated gestures can be combined with authored ones and how rules can be learned from a corpus of scenes where hand annotated actions are available in large quantities.

3 System Overview

The COHIBIT[2] system is a mixed-reality museum exhibit which features tangible interaction and two conversational virtual characters [12]. The visitor of the

[2] COnversational Helpers in an Immersive exhiBIt with a Tangible interface.

exhibit can assemble cars with real car pieces while life-size projected characters assist in the assembly and talk about various topics to convey educational content. The two virtual characters (one male, one female) give context-sensitive hints how to complete the construction, make personalized comments on the visitors' actions, encourage the visitors to continue playing, and provide additional background information about car technology and artificial intelligence. To enhance immersiveness of the exhibit, the characters must be as life-like as possible displaying a varied, yet consistent nonverbal behavior. This is achieved with a hybrid approach of authored actions and rule-based action generation.

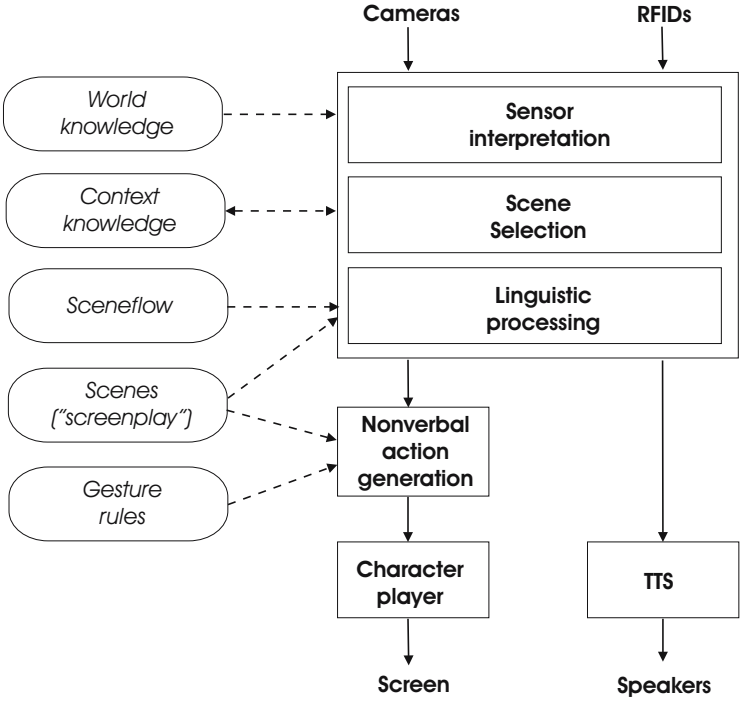

Fig. 2. Overview of the COHIBIT runtime system architecture

The COHIBIT runtime system is depicted in Figure 2. COHIBIT receives input from cameras and RFID tags hidden in the car pieces and the workbench. The signals are interpreted and transformed into events like user arrived, piece X placed on field Y, user departed etc. The scene selection is an extended, hierarchical state machine that uses the state diagram, called sceneflow[3], and the scenes[4] to select, adapt and play an appropriate scene (cf. [13][2]). The linguistic component transforms any generic context components that the author can use

[3] In the current system, the sceneflow encompasses 174 nodes, 29 supernodes and 266 transitions.

[4] The current system consists of 752 scenes.

(e.g. name of currently moved car piece, current time or weather conditions) into grammatically correct surface text. This component also generates a stemmed version of the text (e.g. "went" → "go") for later gesture rule application (see Section 4).

The final surface text is sent to the nonverbal action generator that uses rules to add actions to the already existing pre-authored ones and selects the best ones (see Section 4.2). Text and actions are then sent to text-to-speech (TTS) and animation engines which send the final output is to the output devices, i.e. speakers and screen.

The COHIBIT system's basic mechanism is to constantly recombine and select pre-authored scenes. A large corpus of 752 scenes has been written allowing a rich and varied interaction with hardly any repetitions.

3.1 Authoring Scenes and Actions

The corpus of scenes is represented in a single text document that can be written and extended by a naive user in any kind of text processing software.

Scenes are the smallest units to be played by the system. Within each scene, the author writes dialogue like in a screenplay or theatre script, simply by putting the speaker's initial up front and then type text. There are various commands at the speaker's disposal that can trigger nonverbal actions or query context information from the database. See the following example:

```
A: [bow] Welcome [B look@other] to the car construction [B nod] world!
B: [look@visitor] [happy] Could you do us a favor? Please
   take [GET current-piece def acc] from the table.
   [turn2other] What's the time?
A: [look@other] Well, [check_watch] it's around [GET time-fuzzy].
B: [nod] Thank you.
```

Nonverbal actions are specified within square brackets. If the action should not be performed by the current speaker, the speaker initial can be specified in front of the action name (for instance, "[B look@other]" in the above example).

The system has 28 different animations/actions for the two characters, including gestures, facial expressions and body movement. Some actions are only available for one of the characters and some actions are rarely used. Table 1 shows the most frequently used actions. The gesture actions were loosely named after the inventory defined by Kipp [11]. Actions are grouped into four *channels* which facilitates conflict resolution at the filter step (Section 4.2). The groups are: facial expression (F), gaze (Gz), manual gesture (G), and head movement (H).

Our total corpus consists of a script with 752 scenes. These scenes contain 1781 dialogue turns and 2786 single utterances. Within these scenes 1196 actions have already been authored. Using rules, many actions can automatically be generated, although potential conflicts beween these generated actions and the authored ones must then be resolved. How to do this will be shown in the next Section. The huge number of existing actions inspired us to implement automatic rule learning, using the existing scenes as a training corpus. This will be the topic of Section 5.

Table 1. Table of most frequently used actions for the two characters. The leftmost column shows the action channel: facial expression (F), gaze (Gz), manual gesture (G), and head movement (H). The numbers in the right half relate to corpus partitioning and learning evaluation as discussed in Section 5.

	gesture	*description*	*male character*				*female character*			
			pos.	*neg.*	R_{tr}	R_{te}	*pos.*	*neg.*	R_{tr}	R_{te}
G	cup	*show palm*	29	339	47.1	9.1	43	442	53.8	6.2
F	happy	*smile*	26	156	62.5	44.4	30	212	38.9	54.5
Gz	look@other	*look at other character*	132	748	74.7	55.8	127	721	76.3	56.0
Gz	look@visitor	*look at user*	86	370	94.1	93.9	68	287	97.5	96.2
H	nod	*head nod*	51	319	38.7	5.3	58	514	41.2	22.7
G	point@panel	*point at the display panel behind the characters*	12	140	28.6	0.0	—	—	—	—
G	progressive	*circular metaphoric gesture*	22	263	46.2	50.0	23	282	14.3	0.0
H	shake	*head shake*	13	195	62.5	0.0	19	313	27.3	14.3
G	so_what	*open arms, palm point up*	43	402	36.0	12.5	—	—	—	—
Gz	turn2other	*rotate torso toward other character*	—	—	—	—	20	175	16.7	14.3
Gz	turn2visitor	*rotate torso toward user*	24	79	78.6	66.7	24	122	64.3	33.3
G	walls	*both hands held parallel, palms facing each other*	—	—	—	—	25	341	42.1	28.1

4 Using Rules for Automatic Action Generation

To automate the tedious hand annotation of actions we introduced an intuitive rule mechanism based on keyword spotting. The user defines a set of rules that operate on the utterance level. For each utterance, all applicable rules can fire and generate actions which are stored together with any pre-authored actions that were already there for later conflict resolution.

4.1 Rule Syntax and Usage

Action rules are IF-THEN rules with a left hand side (LHS) consisting of conditions and a right hand side (RHS) consisting of effects. All conditions on the left hand side must be true for the rule to fire, i.e. they are connected by `AND` operators. To implement an `OR`, you write a new rule with equal effects but different conditions.

In gesture generation, for each utterance every rule is tested. If all conditions are true the rule fires. The author can use the predicates in Table 2 to specify the conditional side, predicates can be negated using "!" as a prefix.

On the right hand side the author specifies what happens if a rule fires. Two commands are at the author's disposal: `gen` and `gen_other`. The first generates a gesture for the character who utters the current utterances, the latter for the other character (our system consists only of two characters). The arguments for these commands specify the action name and the position. Position can be

Table 2. Table of predicates that can be used in the conditional part (LHS) of action generation rules

predicate	*description*
`says(‘‘foo baa’’)`	True if the string “foo baa” is contained in the utterance.
`says([you went])`	True if the word stems are found in the (stemmed) utterance. In the case of [you went] the system is looking for “you go”.
`speaker(X)`	True if the utterance is spoken by speaker X.
`begin_scene`	True if the utterance is the first utterance in the scene.
`begin_turn`	True if the utterance is the first utterance in the turn.
`question`	True if the utterance is a question, i.e. it ends with a “?”.
`command(C, A)`	True if the utterance contains the command C with arguments A.

`word`, `begin` or `end`. For word position the system remembers the match position on the conditional side. The action is then inserted before the respective word. This position can be modified by adding an offset like `+1` or `-2` behind the `word` keyword. Some sample rules[5] are:

```
says(‘‘of course’’) −− > gen(cup, begin)
says([develop]) −− > gen(progressive, word - 1)
speaker(Richie) & question −− > gen(cup, begin)
speaker(Tina) & command(picture) &
!command(picture, default.jpg) −− > gen(point@panel, word +1)
```

In our system, we use a set of 57 rules to automatically generate gestures. For the 752 scenes, these rules fire 2688 times.

4.2 Combining Rules and Pre-authored Actions

The rules are interpreted at runtime and are used to generate actions on the fly. However, scripts may also contain pre-authored gestures. The system must decide which actions and how many actions to actually use.

After generation the system has a text utterance annotated with a considerable amount of actions in-between words. Actions are selected by applying a number of constraints. First, at any one spot only *compatible* actions can be executed in parallel which is modelled using four channels: facial expression (F), gaze (Gz), manual gesture (G), and head movement (H). The constraint is that only actions from different channels can be performed in parallel. So a character can look at the user (gaze channel) and make a hand gesture (gesture channel) at the same time, whereas performing two hand gestures at the same time is not possible in our system.

Second, we model that human authored actions are preferred over automatically generated ones by assigning priority values: 2 for human authored actions,

[5] The samples are translated from German to English for better readability.

1 for generated ones. To filter out an action, priority is set to -1. We then apply constraints at three different levels to make the distribution of actions across a scene consistent. On the scene level, a constant action rate R must be observed where R can be specified by the developer. Action rate is measured by dividing number of actions by the number of utterances. On the turn level, no gesture or head move is allowed to occur twice in the same turn. On the utterance level, conflicts between simultaneous actions are resolved by selecting the action with the highest priority and actions with priority -1 are filtered out. The result is a sequence of actions, containing both human authored and automatically generated ones, where the amount of activity is controlled and repetitions and conflicts are filtered out.

5 Learning Rules from the Corpus

Our corpus has been extended over time and with the introduction of rules fewer and fewer actions have been manually annotated. We thus have a situation where some scenes are heavily annotated, some sparsely and some not at all. To obtain training and test material for machine learning we must first define criteria for finding suitable material.

5.1 Preparing the Corpus

We first defined a measure to select suitable training material. We did this using the action rate, setting a minimal threshold to 0.3. We obtained a total Corpus C of 334 scenes. In a second step we disjointly divided the corpus into training data C_{train} and test data C_{test}. In C_{train} we had to define positive and negative samples for each action. A positive sample for action A is an utterance where A occurs. Negative samples could theoretically be all utterances where A does not occur but we thought this might be too restrictive. Just because an action is not annotated it does not mean that it *should not* be there. However, we hypothesized that if within one scene the action A occurs, it might be that the user has intentionally put it there and nowhere else in the scene. So we define our negatives as all utterances u belonging to a scene S where A occurs in S but A does not occur in u. There is still doubt of whether these negatives could be too restrictive. However, we tried to balance the importance of positives vs. negatives with weights (Section 6). Table 1 shows the number of positive and negative utterance found for each speaker and action.

5.2 Learning

An important aspect of the learning task is that it is not a pure classification task that could be resolved with standard techniques like SVMs, n-grams, neural networks or ID3/4.5. Instead, we have to learn a set of conditions plus a position (in the simplest version). However, we can re-formulate the problem to map it to a classification problem but with the drawback of having fewer samples. For the sub-case where an action is generated at the very beginning of an utterance we

can directly apply a classification based approach (see clustering below). Moreover, since we have a hybrid approach where author and machine are supposed to cooperate we pursue the goal of keeping all generated rules human-readable.

We consider rule learning is a two-step process. First, a *rule generator* systematically generates a number of potentially interesting rules. Then, these rules are tested by the *rule appraisal* module against the positive and negative samples for matches and false positives. The best ones, according to a weighted measure, are selected. So the meat is obviously in the rule generator, whereas the appraisal module allows you to tweak your results using weight parameters. Note that one difficulty in rule learning in our case is that it is not only a question of whether a rule fires or not but also of where the action is placed.

The rule generator runs for one action at a time. For each positive utterance for this action the generator produces different answers to the question "what might have caused this action to be produced here?". We propose two mechanisms for genrating rules: one word based, one cluster based. The word based generation is very easy: each word in the positive utterance is seen as a potential trigger for the action. Let the utterance be $(w_0, ..., w_{i-1}, a_i, w_{i+1}, ..., w_n)$ where w_j are words and a_i is the action at position i. Then, for each w_j we generate a rule of the form:

$$\texttt{says}(w_j)\ \texttt{--> gen}(A, \texttt{ word - } (i-j+1))$$

In the cluster based approach we try to identify patterns of recurring *ordered sequences of words* in the positive samples of one action. This works only for begin and end type rules. In our corpus, we noted that many actions occur specifically at the beginning of utterances so we deemed it worth looking at this special case in detail. A word vector v_1 is an ordered partial vector of v_2 if the words in v_1 are all in v_2 and are ordered in the same way as the corresponding words in v_2. If you have two utterances u_1 and u_2 you can define the ordered word overlap $\hat{w}$ by

$$\hat{w}(u_1, u_2) := \max_{|v|}(\text{vector } v : v \text{ is ordered partial vector of } u_1 \text{ and } u_2)$$

The $\hat{w}$ function gives you an overlap vector v, the number of overlapping words is $|\hat{w}|$ which gives you a distance metric for cluster analysis. Cluster analysis makes it possible to obtain smaller patterns with bigger generality. We applied a simple nearest neighbor clustering algorithm to cluster similar patterns together. The $\hat{w}$ function is trivially expanded to compare an utterance with a set of utterances. We can improve the quality of the rules by using negation. We use the same clustering method for pattern identification to exclude false positives of a rule. The recognized patterns are added to the conditional side as negated conditions.

When learning rules it is important to ignore some words that can be considered "noise". Function words occur often but are also often relatively meaningless in terms of generation. But because of their frequency the learning algorithm often finds patterns in function words. Therefore, we have to ignore them. Which

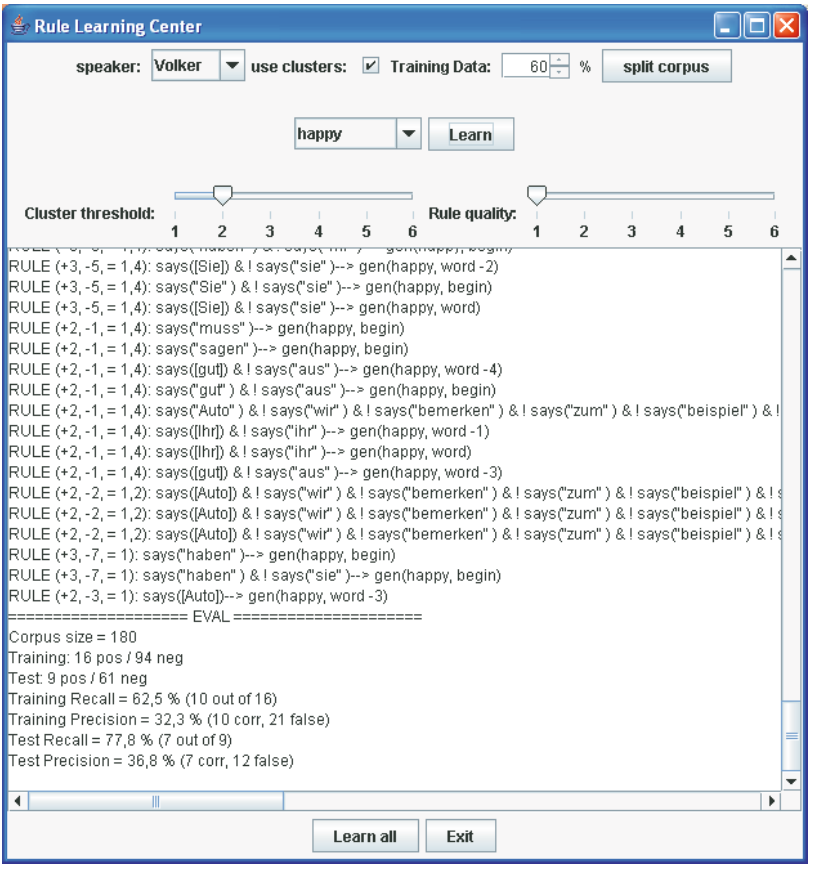

Fig. 3. Graphical user interface for gesture rule learning

words should be ignored is not always clear. Do you take out "and"? It might indicate metaphoric gestures that represent *sequence*. Clearer cases are "of", "by" or "to".

We implemented a graphical user interface (GUI) where the author can experiment with different parameters of rule learning (Figure 3). Our idea is that rule aquisition is not a fully automatic process but rather a source for inspiration for the author who may find some of the generated patterns appealing and illuminating while others might simply be unintuitive.

In the GUI the author can select the partitioning of test/training corpus, the speaker, the action and parameters for rule quality and clustering.

6 Evaluation

To really evaluate generated gestures one would have to set up an experiment where the results of different approaches are judged by independent coders. Then,

one can make a comparative analysis and draw final conclusions about each method. While we have not made such an extensive study yet, we did a quantitative analysis of the learning algorithm described above. For this, we partitioned the corpus into training (60%) and test (40%) and then computed recall R and precision P. Recall for an action A is defined as the number of utterances where an action was correctly generated divided by the total number of utterances. Precision is the number of times that rules fired correctly divided by the total number of times that rules fired. Note that we had a very strict measure for precision as every generated action that is at an incorrect position counts as a false positive.

In total, we achieved a recall of 56.9% and precision of 33.7% on the training data, and recall of 33.8%, precision of 13.6% on the test data for the male character. For the female character, we got recall/training of 47.2%, precision/training of 47.1%, recall/test of 32.6% and precision/test of 32.1% for the female character. See Table 1 for detailed results for every speaker and action.

Even though the results may look low one has to reflect what can be expected. Since the placement of actions is a rather arbitrary decision and a large number of different placements may all be correct, we cannot expect our system to predict precisely where an action must be placed. However, if sometimes the placement matches exactly what the human author has done it is an indication that it is a good rule. If it does not match the rule is not necessarily false. So recall/precision values must be looked at sceptically. We were actually quite satisfied with the quality of the output in terms of usability and human readability. The results vary largely across actions, from very good (look@visitor) to zero (nod, shake). These learned rules are meant to extend and complement existing rules, not to replace them altogether.

A more thorough evaluation should look not only at recall and precision but at the actions that the new, learned rules produce. However, judging these produced gestures is a non-trivial task. What are the evaluation criteria? Are we talking about general appropriateness of the gestures or about whether they reflect a certain style of the character? Quality of gestures is hard to grasp. Although first approaches exist (e.g. [5]) more research in terms of evaluation criteria is needed before tackling this difficult task.

7 Conclusions

We presented a system for authoring nonverbal actions in a character based system, for defining generation rules and, finally, for automatically aquiring rules by machine learning. Authoring actions directly is done in a screenplay-like style. Rule definition is done with simple IF-THEN rules. Rule learning relies on systematic rule candidate generation, based on word traversal and clustering, and rule evaluation. Learning was shown to have a very varied quality in terms of recall and precision but these measures are only very tentative indicators of quality. Hence, more qualitative evaluations remain to be done.

All three mechanism are used in a hybrid approach to nonverbal action creation that allows an arbitrary mixture of creative control and economic automa-

tion. This technique could also be used for recognizing other kinds of tags in a text, e.g. emotions or emotion eliciting conditions [14], dialogue acts [15], other multimodal actions like displaying pictures, changning light, audio clips.

Future work on this system includes more precise timing capabilities for the generated actions, extending learning to all types of predicates, and integrating more complex gesture types (hold gestures and multiple stroke gestures) to the repertoire.

Acknowledgements

This work is partially funded by the German Ministry for Education and Research (BMBF) as part of the VirtualHuman project under grant 01 IMB 01A.

References

1. Rist, T., André, E., Baldes, S., Gebhard, P., Klesen, M., Kipp, M., Rist, P., Schmitt, M.: A review of the development of embodied presentation agents and their application fields. In Prendinger, H., Ishizuka, M., eds.: Life-Like Characters – Tools, Affective Functions, and Applications. Springer, Heidelberg (2003) 377–404
2. Gebhard, P., Kipp, M., Klesen, M., Rist, T.: Authoring scenes for adaptive, interactive performances. In: Proceedings of the Second International Joint Conference on Autonomous Agents and Multiagent Systems. (2003) 725–732
3. Cassell, J., Vilhjálmsson, H., Bickmore, T.: BEAT: the Behavior Expression Animation Toolkit. In: Proceedings of SIGGRAPH 2001. (2001) 477–486
4. Noma, T., Zhao, L., Badler, N.: Design of a Virtual Human Presenter. IEEE Journal of Computer Graphics and Applications **20** (2000) 79–85
5. Hartmann, B., Mancini, M., Buisine, S., Pelachaud, C.: Design and evaluation of expressive gesture synthesis for embodied conversational agents. In: Proceedings of the fourth international joint conference on Autonomous agents and multiagent systems. ACM Press (2005)
6. Kopp, S., Tepper, P., Cassell, J.: Towards integrated microplanning of language and iconic gesture for multimodal output. In: Proc. Int'l Conf. Multimodal Interfaces 2004. (2004) 97–104
7. Cassell, J., Pelachaud, C., Badler, N., Steedman, M., Achorn, B., Becket, T., Douville, B., Prevost, S., Stone, M.: Animated Conversation: Rule-Based Generation of Facial Expression, Gesture & Spoken Intonation for Multiple Conversational Agents. In: Proceedings of SIGGRAPH '94. (1994) 413–420
8. André, E., Müller, J., Rist, T.: WIP/PPP: Automatic Generation of Personalized Multimedia Presentations. In: Proceedings of Multimedia 96, 4th ACM International Multimedia Conference. ACM Press, Boston, MA (1996) 407–408
9. Noot, H., Ruttkay, Z.: Gesture in style. In: Proc. Gesture Workshop 2003. Volume 2915 of LNAI., Berlin and Heidelberg, Germany, Springer-Verlag (2004) 324–337
10. Kopp, S., Sowa, T., Wachsmuth, I.: Imitation games with an artificial agent: from mimicking to understanding shape-related iconic gestures. In: Proc. Gesture Workshop 2003. Volume 2915 of LNCS., Berlin and Heidelberg, Germany, Springer (2004) 436–447
11. Kipp, M.: Gesture Generation by Imitation: From Human Behavior to Computer Character Animation. Dissertation.com, Boca Raton, Florida (2004)

12. Ndiaye, A., Gebhard, P., Kipp, M., Klesen, M., Schneider, M., Wahlster, W.: Ambient intelligence in edutainment: Tangible interaction with life-like exhibit guides. In: Proceedings of the first Confernce on INtelligent TEchnologies for interactive enterTAINment (INTETAIN), Berlin, Heidelberg, Springer (2005) 104–113
13. Klesen, M., Kipp, M., Gebhard, P., Rist, T.: Staging exhibitions: Methods and tools for modeling narrative structure to produce interactive performances with virtual actors. Virtual Reality. Special Issue on Storytelling in Virtual Environments **7** (2003) 17–29
14. Gebhard, P., Kipp, M., Klesen, M., Rist, T.: Adding the emotional dimension to scripting character dialogues. In: Intelligent Agents, 4th International Workshop, IVA 2003, Kloster Irsee, Germany, September 15-17, 2003, Proceedings. (2003) 48–56
15. Reithinger, N., Klesen, M.: Dialogue act classification using language models. In Kokkinakis, G., Fakotakis, N., Dermatas, E., eds.: Proceedings of the 5th European Conference on Speech Communication and Technology (Eurospeech 97). (1997) 2235–2238