Predicting when to laugh with structured classification
Bilal Piot, Olivier Pietquin, Matthieu Geist

To cite this version:
Bilal Piot, Olivier Pietquin, Matthieu Geist. Predicting when to laugh with structured classification. InterSpeech 2014, Sep 2014, Singapore, Singapore. pp.1786-1790. hal-01104739

HAL Id: hal-01104739
https://hal-supelec.archives-ouvertes.fr/hal-01104739
Submitted on 19 Jan 2015

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L’archive ouverte pluridisciplinaire HAL, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d’enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Distributed under a Creative Commons Attribution - NonCommercial - NoDerivatives| 4.0 International License
PREDICTING WHEN TO LAUGH WITH STRUCTURED CLASSIFICATION

Bilal Piot¹, Olivier Pietquin², Matthieu Geist¹

¹SUPELEC IMS-MaLiS research group and UMI 2958 (GeorgiaTech - CNRS)
²University Lille 1, LIFL (UMR 8022 CNRS/Lille 1) - SequeL team
¹firstname.lastname@supelec.fr, ²olivier.pietquin@univ-lille1.fr

ABSTRACT

Today, Embodied Conversational Agents (ECAs) are emerging as natural media to interact with machines. Applications are numerous and ECAs can reduce the technological gap between people by providing user-friendly interfaces. Yet, ECAs are still unable to produce social signals appropriately during their interaction with humans, which tends to make the interaction less instinctive. Especially, very little attention has been paid to the use of laughter in human-avator interactions despite the crucial role played by laughter in human-human interaction. In this paper, a method for predicting the most appropriate moment for laughing for an ECA is proposed. Imitation learning via a structured classification algorithm is used in this purpose and is shown to produce a behavior similar to humans’ on a practical application: the yes/no game.

Index Terms— Laughter; Imitation Learning; Structured Classification

1. INTRODUCTION

Building efficient and user-friendly human-machine interfaces is a key challenge for the future of computer science, enabling a large public to interact with complex systems and reducing the technological gap between people. In the last decade, Embodied Conversational Agents (ECAs) emerged as such interfaces. Yet, their behaviour is still perceived as quite unnatural to users. One of the reasons of this bad perception is the inability of ECAs to make a proper use of social signals, although there exists some research on this topic [1]. Among these signals, laughter is a prominent feature used by humans during interactions. Yet, very little attention has been paid to enable ECAs with laughter capabilities until recently [2].

Enabling ECAs with laughter capabilities is not only about being able to synthesize audio-visual laughter signals [3, 4]. It is also concerned by an appropriate management of laughter during the interaction. There is thus a need for a laughter-enabled interaction manager, able to decide when to laugh so that it is appropriate in the conversation. This being said, it remains uneasy to define what is an appropriate moment to laugh.

More formally, the task of the laughter-enabled interaction Manager (IM) is to take decisions about whether to laugh or not. These decisions have to be taken according to the interaction context which can be inferred from laughter, speech and smile detection modules (detecting social signals emitted by the users) implemented in the ECA but also by the task context (for example, if the human is playing a game with the ECA, what is the status of the game). Formally, the IM is thus a module implementing a mapping between contexts (or states noted \( s \in S \)) and decisions (or actions noted \( a \in A \)). Let’s call this mapping a policy, noted \( \pi(s) = a \). This mapping is quite difficult to learn from real data as the laughs are quite rare and very different from one user to another.

In this paper, we describe the research results for learning such a mapping from data, recorded during some human-human interactions, so as to implement, in the IM, a behavior similar to the one of a human. An imitation learning method is thus adopted. Especially, structured classification is investigated and proven to efficiently learn a behavior similar to human users where the similarity between human and algorithms is measured via a new criterion called Naturalness and defined in Sec. 5. In addition, we use a technique of boosting for the structured classification algorithm which makes it a non-parametric algorithm. This avoids the choice of meta-parameters. Finally, we test different imitation algorithms on data sets of real laughs in a natural interaction context which is the yes/no game described in Sec. 4.

2. IMITATION LEARNING

Describing the optimal behavior of the avatar is a very tricky task. It would require the perfect knowledge of rules prevailing to the generation of laughter by humans. Interpreting sources of laughter or predicting laughter from a cognitive or psychology perspective is non-trivial. Therefore, a data-driven method has been preferred here. Especially, learning by imitation seems the best suited framework to learn the IM policy. Indeed, humans are implementing such a policy and they can provide examples of natural behaviors.

Formally, in the learning by imitation framework, an artificial learning agent (here the IM) learns to behave optimally by observing some expert agent demonstrating the task. The expert is implementing an optimal policy noted \( \pi^* \) and the demonstrations provide a set of examples \( \{s_i, a_i = \pi^*(s_i)\}_{1 \leq i \leq N} \). The problem is thus to learn a policy \( \pi \) such that \( \forall s, \pi(s) \approx \pi^*(s) \) from the set of demonstrations.

One way to address the problem of imitation learning is to reduce it to a Multi-Class Classification (MCC) problem [5, 6, 7, 8]. The goal of MCC is, given a training set \( D = \{(s_i, a_i, \pi^*(s_i))\}_{1 \leq i \leq N} \) where \( S \) is a compact set of inputs (generally a compact set of \( \mathbb{R}^n \)) and \( A \) a finite set of labels, to find a decision rule \( \pi \in A^S \) that generalizes the relation between inputs and labels. More formally, it consists in finding a decision rule \( \pi \in H \), where \( H \subseteq \mathbb{R}^S \times A \) is called the hypothesis space, that tries to minimize the following empirical risk:

\[
T(\pi) = \frac{1}{N} \sum_{i=1}^{N} \mathbf{1}_{\{\pi(s_i) \neq a_i\}},
\]

where \( \mathbf{1}_{\{x = a\}} = 1 \) if \( a \neq b \) and 0 otherwise.

A large literature already exists about the MCC problem. Well known methods such as Classification Trees [9], K-Nearest Neighbors (KNN) [10] and Support Vector Machines (SVM) [11, 12] are widely used and statistically studied. In [5], the authors use an arti-
cial neural network to learn a driving policy for a robotic vehicle. Neural nets are also used in [1] to learn to play video games (although the method is more generic and could use other MCC methods). KNN’s where used in [2] in a similar application as the one described in this paper. In [3], structured classification [14] is used to learn a grasping control policy for a robotic arm.

3. STRUCTURED CLASSIFICATION FOR Imitation LEARNING

In [6], the authors use a large margin approach which allows adding some prior (or structure) via a margin function in the classification method. That is why it is considered as a structured classification method. The large margin approach is a score-based MCC where the decision rule $\pi \in A^5$ is obtained via a score function $q \in \mathbb{R}^{S \times A}$ such that $\forall s \in S, \pi(s) \in \text{argmax}_{a \in A} q(s,a)$. The large margin approach consists, given the training set $D$, in solving the following optimization problem:

$$q^* = \arg\min_{q \in \mathbb{R}^{S \times A}} J(q),$$

$$J(q) = \frac{1}{N} \sum_{i=1}^{N} \max_{a \in A} \{g(s_i,a) + l(s_i,a_i)\} - q(s_i,a_i),$$

where $l \in \mathbb{R}^{S \times A}$ is called the margin function. If it is zero, minimizing $J(q)$ attempts to find a score function $q^*$ for which the example labels are scored higher than all other labels. Choosing a nonzero margin function improves generalization [6]. Instead of requiring only that the demonstrated label is scored higher than all other labels, it requires it to be better than each label $a$ by an amount given by the margin function. Thus, the margin function allows deciding which samples are required to be well classified by putting an important margin on this particular example compared to the others. The policy outputted by this algorithm would be $\pi(s) \in \text{argmax}_{a \in A} q(s,a)$ where $q$ is the output of the minimization of $J(q)$. The advantages of this method are its simplicity and the possibility to change the margin that allows us to adapt to specific characteristics of the problem. In addition, in [14], the authors use a boosting technique to solve the optimization problem given by Eq. 1 which is advantageous. A boosting method is an interesting optimization technique: it minimizes directly the criterion in Eq. 1 without the step of choosing features. As presented in [15], a boosting algorithm is a projected sub-gradient descent [10] of a convex functional (here $J$ is convex relatively to the variable $q$) in a specific functions space (here $\mathbb{R}^{S \times A}$) which has to be a Hilbert space. Boosting algorithms use a projection step on a restriction set of functions when optimizing over functions space, because the functions representing the gradient are often computationally difficult to manipulate and do not generalize well to new inputs [15]. In boosting literature, the restriction set corresponds directly to the set of hypotheses generated by a weak learner. In our experiments, we choose as restriction set the set of classification trees with two classes.

4. EXPERIMENTAL SETUP

The yes/no game is one of the possible scenarios of an interaction between humans and avatars where laughter is involved. In this game, players must respond to questions without saying “yes” or “no”. The experiment scenario we present in this article is illustrated in Fig. 2. Two users are sitting on one side of a table while a virtual agent projected on a large screen is placed on the opposite side of the table. The users start to play the yes/no game, one asking questions (e.g., “what’s your nationality?”, “are you sure?”), this user is named $U_1$, and the other one answering trying to avoid to say “yes” or “no” (e.g., “I’m not sure” or “definitely”), this user is named $U_2$. The avatar, named $A$, participates to the interaction by laughing and asking questions. Of course, $U_1$ and $A$ try to make $U_2$ to say “yes” or “no” and thus try to induce a loss of self-control. At any point, laughter can occur for any participant. The avatar has to generate laughter at appropriate moments given its perception of the context.

As shown in Fig 3, detection of humans’ laughter is performed through body (Kinect and body markers), face (Kinect) and speech (head mounted microphones) analysis [17]. Several recognition algorithms are executed in real-time to determine users’ expressivity of motion.

In order to train our avatar by an imitation learning algorithm, several experiments are first recorded, where the avatar (symbolized by a screen in Fig. 2) is replaced by a human playing the role of the avatar (this is the expert we want to imitate). The same detection material as for the two other participants is used for the human playing the role of the avatar. Thanks to those recordings an expert data set $D = \{s_i,a_i = \pi^E(s_i)\}_{1 \leq i \leq N}$ is generated which is the input of an imitation learning algorithm. Indeed, for each user ($U_1$ and $U_2$), the recognition algorithms are able to extract each 0.5s a 4 features which are real values between 0 and 1. The 4 features are the probability of speech, the probability of laughter, the intensity of laughter and the probability of smile. Moreover, another feature, which represents the context of the game, is added by annotation of the recordings: 0 when the game is currently ongoing and 1 when it ends (that is when $U_2$ said “yes” or “no” or that some time-out occurred). Thus each 0.5s, we are provided 9 features (4 features for $U_1$, 4 features for $U_2$ and the context) that represents the state of the game $s_i$. Finally, by annotations of the recordings, we provide each 0.5s a binary information (1 or 0) giving the decision of the expert ($a_i$: laugh/no laugh) (so it is a 2 actions decision process). A sample $(s_i,a_i)$ where $a_i = 0$ corresponds to a no laugh sample and a sample $(s_i,a_i)$ where $a_i = 1$ corresponds to a laugh sample. In addition, we also collect, by annotations, the binary laugh/no laugh information for $U_1$ and $U_2$: $(a^{U_1}_i,a^{U_2}_i)_{1 \leq i \leq N}$. Now that we have the expert data set, it is possible to use it as an input to different imitation learning algorithms.

Fig. 1. Experimental setup.
In this section, we present the results obtained by applying different imitation learning algorithms to the expert data set. We use 4 different algorithms, 3 classical classification algorithms, which are KNN, Classification Tree and SVM, and the large-margin algorithm presented in Section 3. The KNN algorithm was previously used in [2] where we choose the following margin structure:

Sub-sample $D_p$ and each algorithm $\text{algo}^p$ we define the policy $\pi_p^\text{algo}(s_i) \in A^i$ learned on the remaining $P - 1$ sub-samples. In addition, we define, for each sub-sample $D_p$, the number of laugh samples $N_p^{\text{laugh}} = \sum_{j=1}^{N_p} 1_{(a_j = 1)}$ and the number of no laugh samples $N_p^{\text{no laugh}} = N_p - N_p^{\text{laugh}}$. Several quality evaluation criteria were used for each algorithm. The first criterion is the mean over the $P$ folds of the global classification rate:

$$\frac{1}{P} \sum_{p=1}^{P} \frac{1}{N_p} \sum_{j=1}^{N_p} 1_{\{\pi_p^\text{algo}(s_j) = a_j\}}.$$  

The second criterion is the mean over the $P$ folds of the classification rate on laugh samples:

$$\frac{1}{P} \sum_{p=1}^{P} \frac{1}{N_p^{\text{laugh}}} \sum_{j=1}^{N_p^{\text{laugh}}} 1_{\{\pi_p^\text{algo}(s_j) = a_j\}} 1_{\{a_j = 1\}}.$$  

The third criterion is the mean over the $P$ folds of the classification rate on no laugh samples:

$$\frac{1}{P} \sum_{p=1}^{P} \frac{1}{N_p^{\text{no laugh}}} \sum_{j=1}^{N_p^{\text{no laugh}}} 1_{\{\pi_p^\text{algo}(s_j) = a_j\}} 1_{\{a_j = 0\}}.$$  

We choose those different criteria in order to see the quality of each algorithm on the laugh samples and the no laugh samples because those two classes are not well balanced (basically there is 5 times more no laugh samples than laugh samples). In Table 1 we have the results of the different algorithms in terms of classification rates with $P = 5$.

<table>
<thead>
<tr>
<th>Algorithms</th>
<th>Global</th>
<th>laugh</th>
<th>no_laugh</th>
</tr>
</thead>
<tbody>
<tr>
<td>Large Margin</td>
<td>0.6871</td>
<td>0.3256</td>
<td>0.8081</td>
</tr>
<tr>
<td>SVM</td>
<td>0.6723</td>
<td>0.3106</td>
<td>0.7893</td>
</tr>
<tr>
<td>KNN</td>
<td>0.5440</td>
<td>0.6347</td>
<td>0.5173</td>
</tr>
<tr>
<td>Tree</td>
<td>0.5570</td>
<td>0.5106</td>
<td>0.5732</td>
</tr>
</tbody>
</table>

Table 1. Classification rates.

We have 18 minutes of recordings were collected in three sessions where the game was played several times (at least twice by recordings). This provided an expert data set.

Eighteen minutes of recordings were collected in three sessions where the game was played several times (at least twice by recordings). This provided an expert data set.

In P-fold cross-validation, the original data set $D = \{s_i, a_i = \pi^p(s_i)\}_{1 \leq i \leq N}$ of 2285 examples (that is the number of 0.5s frames). The 4 algorithms were trained on this data set. In order to compare the performances of the algorithms, we use a $P$-fold cross validation. In $P$-fold cross-validation, the original data set $D$ is partitioned into $P$ equal size sub-samples $D = (D_p)_{1 \leq p \leq P}$, where $D_p = \{s_{j,p}, a_{j,p} = \pi^p(s_{j,p})\}_{1 \leq j \leq N_p}$ and $\sum_{p=1}^{P} N_p = N$. Of the $P$ sub-samples, a single sub-sample is retained as the validation data for testing the algorithm, and the remaining $P - 1$ sub-samples are used as training data. The cross-validation process is then repeated $P$ times (the folds), with each of the $P$ sub-samples used exactly once as the validation data. The $P$ results from the folds then can be averaged to produce a single estimation. For each

$$l(s, a) = \begin{cases} 0 & \text{if } a = a_i, \\ 6 & \text{if } a \neq a_i \text{ and } a_i = 0, \text{(no laugh)} \\ 1 & \text{if } a \neq a_i \text{ and } a_i = 1, \text{(laugh)} \end{cases}.$$  

Fig. 2. Real Demo.
to the behavior of the expert. The idea is to compare if relatively to the two other users the human playing the avatar and the algorithm have the same behavior.

In order to see if there is a similarity between the behavior of the user playing the avatar $A_{\text{expert}}$ and the one learnt by the algorithms $A_{\text{algo}}$, we check if the behavior of the expert $A_{\text{expert}}$ compares to the users $((U_q)_{q=1,2})$ similarly to the way the avatar’s behavior $A_{\text{algo}}$ compares to the users $((U_q)_{q=1,2})$. The idea is to show that the avatar doesn’t differ more from $U_1$ and $U_2$ than the expert does. To do so, for each user $U_q$ and each sub-sample $D_p$, we define the number of laugh samples $N_{\text{laugh}}^{U_q,D_p} = \sum_{j=1}^{N_p} 1\{v_{j,q}^{P} = \text{true}\}$ and the number of no laugh samples $N_{\text{no laugh}}^{U_q,D_p} = N_p - N_{\text{laugh}}^{U_q,D_p}$. Three criterions were used: the global rate, the laugh rate and no laugh rate. The global criterion rate $r_{\text{global}}^{U_q,D_p}$ gives the rate of agreed no laugh between the avatar and one of the users:

$$\frac{1}{2} \sum_{q=1}^{2} \sum_{p=1}^{P} \sum_{j=1}^{N_p} \frac{1}{N_p} \sum_{j=1}^{N_p} 1\{v_{j,q}^{P} = \text{true}\}.1\{v_{j,'q}^{P} = \text{true}\}.$$

The no laugh criterion rate $r_{\text{no laugh}}^{U_q,D_p}$ gives the rate of agreed no laugh between the avatar and one of the users:

$$\frac{1}{2} \sum_{q=1}^{2} \sum_{p=1}^{P} \sum_{j=1}^{N_p} \frac{1}{N_p} \sum_{j=1}^{N_p} 1\{v_{j,q}^{P} = \text{false}\}.1\{v_{j,'q}^{P} = \text{false}\}.$$

In order to have a single number representing the similarity between the expert avatar $A$ and the avatars outputs by the algorithms, a new criterion, called Naturalness $N_{\text{algo}}$, is defined as follows:

$$N_{\text{algo}} = \prod_{i=1}^{3} \frac{\min(r_{\text{global}}, r_{\text{laugh}})}{\max(r_{\text{algo}}, r_{\text{expert}})}.$$


