A Deep Learning Architecture for Emotional Aware Chatbots

R.H. Grouls
University of Utrecht

Chatbots cover a broad range of possible applications. Interacting with human emotions is a small but necessary subset of the skillset necessary for meaningful interaction. This paper explores strategies to create a chatbot that is able to adapt to the emotional cues during a conversation. A first challenge is to handle the dimensionality of human communication that is addressed with strategies to reduce the dimensionality of both input and output. Additional challenges are handling the continuous action space of semantic vectors and the variations in personality types. I propose an ensemble model of machine learning techniques and conclude with a perspective on the generation of meaning and the consequences this has for a possible implementation.

Keywords: Reinforcement learning, DDPG, emotion recognition, non-verbal communication

Introduction

Emotional aware chatbots

A chatbot can be defined as "computer programs that interact with users using natural languages" and the first chatbots that could do so have been developed in the 60s (Shawar & Atwell, 2007). One of the first chatbots was named ELIZA and was built at MIT in 1966. ELIZA was intended to simulate a Rogerian psychotherapists and operated with techniques like pattern-matching to react to user input and by asking basic questions modeled from a therapist like "please tell me more" (Shum, He, & Li, 2018). Despite this simple approach and the limited domain, some users actually believed they were talking to a real person (Shum et al., 2018). Since these early years, a lot of work has been done in the domain of chatbots and Natural Language Processing. More generic chatbots have been developed by the mayor tech companies to act like intelligent personal assistants (IPAs) (Shum et al., 2018). There are also more specialized chatbots like Woebot and Wysa that offer cognitive-behavioral therapy with significant impact on the users (Fitzpatrick, Darcy, & Vierhile, 2017; Inkster, Sarda, & Subramanian, 2018). In 2016, Facebook and Microsoft stimulated users to create their own chatbots on their messaging systems. Within a year, over 30.000 chatbots had been launched on Facebook Messenger (Brandtzaeg & Følstad, 2017). While a lot of these chatbots are used for customer service or marketing purposes, end users of chatbots report a wide variety of motivations to use chatbots including increasing productivity, information retrieval, entertainment or social and relational needs (Brandtzaeg & Følstad, 2017). For every domain and motivation, there are different drivers that make a chatbot successful. In general, chatbots use human language as an interface to interact with the user, and mastering the different aspects of human language better will improve the ability to interact with the user. An important aspect of human language that chatbots should learn to master is handling human emotion (Brandtzaeg & Følstad, 2017; Pamungkas, 2019; Lee, Oh, & Choi, 2017; Oh, Lee, Ko, & Choi, 2017; Kim, Kim, Kim, & Lee, 2018; Huang & Zaïane, 2019). As this is a very generic problem, with a wide range of possible applications, this paper looks into the subject without trying to pinpoint how and when this approach could be applied. The main question of this paper is what a effective architecture might be for an adaptive machine learning model to interact with a user on an emotional and nonverbal level. Subquestions are how to handle the (i) high dimensionality of human communication (ii) continuous actions space (iii) variations in human personality types.

Dimensionality reduction

High-dimensionality

An important challenge in mastering language is the high-dimensionality and complexity of the patterns that need to be learned for both the state space and action space. For example, the same intention or meaning can be verbalized in a lot of different ways. Also, the meaning of a single word or even a complete sentence can be ambiguous depending on how the grammatical structure of a sentence is analyzed. Language typically has long range relationships, which means that the meaning of a certain word can sometimes only be understood in the context of another word that occurred before. This means that an algorithm that can properly interpret what is being said, need to take into consideration what has been said before, sometimes even more than one sentence earlier. While some types of chatbots (like the IPAs) intend to accept the complete range of linguistic expression, it is obvious to everyone that has ever tried such a system that an IPA will struggle to understand what you mean. Human users that are skilled in using IPAs will learn which syntax they have to follow in order to help the IPA understand what they mean. This is an example of how a reduction in the state space helps a chatbot to navigate meaning and intention of a conversation. Some chatbots limit the complexity by offering closed questions (e.g. ELIZA asking "What is your name?").

Another approach is to reduce the action space. Some researchers do this by using clustered actions, which are groups of sentences that have related meaning as defined by a semantic vector derived from word embeddings (Cuayáhuitl, Lee, Ryu, Choi, et al., 2019). Others focus on creating hard-coded scripts that follow a certain theory, like cognitive behavior therapy (Fitzpatrick et al., 2017). In this paper I will take the same approach to this problem: making the machine learning manageable by reducing both state space and action space.

Reducing state space

Most chatbots that attempt to handle emotions focus on the meaning of words or the intonation of the voice (Lee et al., 2017; Huang & Zaïane, 2019; Oh et al., 2017; Kim et al., 2018). However, verbal communication is very limited in the amount of words it can produce per time unit. While studies have found rates up to an average of 13.83 syllables per seconds for sports commentary (Fonagy & Magdics, 1960), the average rate ranges between 3.3 and 5.9 syllables per second depending on emotional and social context (Arnfield, Roach, Setter, Greasley, & Horton, 1995).

Because a sentence contains only a few words, there are just a few data points which is a problematic if we are trying to solve a problem with such a high dimensionality. We don't get that much observations per time unit, even none when people do not talk, while the observations we do get have a very high degree of freedom which makes the dimensionality extremely high. Typically, to solve a problem with machine learning we would want to have this the other way around: many observations with a low dimensionality.

Consider this: when someone speaks a sentence this usually contains somewhere between 1 and 100 words. These words can be picked from a vocabulary that runs into the tens of thousands. And the syntax is very flexible, as we can combine the same words in numerous ways. This yields into an almost endless amount of possible sentences that can be uttered.

While auditory information has a slightly higher observational density (syllable can contain more than one tone), the range of tones and thus the degrees of freedom are much lower. In addition to this the tone of voice can transfer additional information like stress (Kurniawan, Maslov, & Pechenizkiy, 2013) and emotion (Casale, Russo, Scebba, & Serrano, 2008).

But when comparing both verbal and auditory to the observational density of facial expression, the latter has a substantially higher density in terms of observations per second. It is impossible to *not* have a facial expression, so there are no "silent" periods. This means that we have a continuous stream of information, which increases the amount of information. In addition to the ordinary facial expression of emotion, emotions will also express themselves in the form of micro-expressions. These are defined as "rapid, involuntary facial expressions which reveal emotions that people do not intend to show" (X. Li et al., 2017). This gives us access to a new domain of information, a domain that can reveal subconscious reactions that could be very difficult to spot with access to just words.

The result is that using facial expressions yields a lot more observations that we can use to learn from. At the same time, the degrees of freedom for facial expressions are much lower than those of natural language. There are just 20 facial muscles available we can combine to generate expressions. These 20 facial muscles have a fixed location, which limits the syntax of this visual language a lot. Even though there are cultural differences how facial expressions are recognized and emotions can be combined in expressions, there is still enough overlap to talk about universal facial expressions (Matsumoto, 1992; Boucher & Carlson, 1980; Barrett, Adolphs, Marsella, Martinez, & Pollak, 2019).

Both of these characteristics (increased observational density, decreased degrees of freedom per observation) help to reduce the curse of dimensionality.

We will lose a lot of information that can be essential in a lot of domains while focusing exclusively on facial expressions. But for our purpose of handling human emotion the information we receive might tell us more than enough. It could even unlock domains that would have been inaccessible when relying exclusively on spoken words.

As the input of the architecture I will describe in full later on, I will use emotion recognition implemented though deep convolutional neural networks in order to reduce the dimensionality of the facial expressions, as described and implemented by Correa, Jonker, Ozo, and Stolk (2016). This model takes facial expressions through a live camera as input and outputs a vector with seven emotions (neutral, surprised, sad, happy, fearful, disgusted, angry) where the intensity of every emotion is a number between 0 and 1. This effectively reduces a snapshot of the state space to $\mathbb{R}^7$. An illustration from the Correa et al. (2016) paper is shown in figure 1, where the face is classified as mainly disgusted with a little bit of anger as is shown by the blue bars. This processing has a certain frequency f which can be expressed as images processed per second. While facial expressions can change very fast (as with micro-expressions) it makes sense to observe the emotional state for a certain length of time that is a longer than a few milliseconds. Emotional reactions to images can comprise several seconds and sequences of reactions (e.g. we could be initially surprised and then start smiling). If we observe the state space in discrete time steps t with a length l (e.g. 2 seconds) we will obtain multiple instances of this $\mathbb{R}^7$

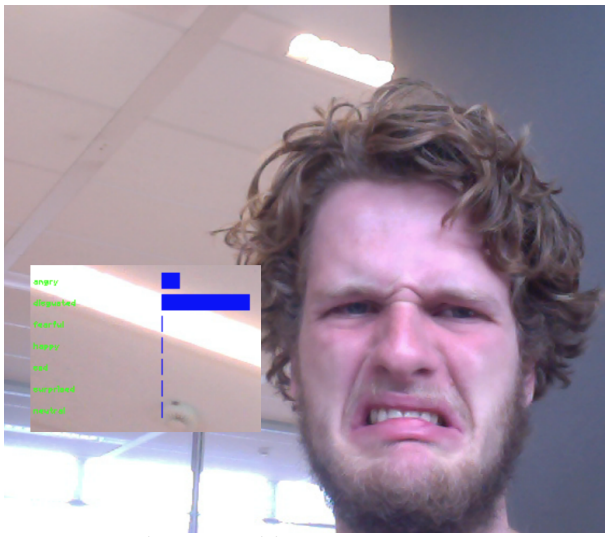

Figure 1. Emotion recognition

vector, depending on the sample rate f. We can denote the amount of instances per time step as $L = f \times l$. Stacking L observations gives us a $\mathbb{R}^{7\times L}$ vector as a representation of the state space s_t.

Reducing action space

Where natural language increases the dimensionality of the state space, it does the same for the action space if we expect the chatbot to use natural language. Because of the high dimensionality, it is much more difficult for a machine learning model to express the right[1] emotional tone through natural language. The high dimensionality is reflected in the model as millions of weights that all need to be adjusted and trained, which makes the model infeasible to train. And while this can seemingly be circumvented by using scripted answers (like ELIZA did: "oh interesting, tell me more...") this sets high expectations for the grammatical skill level of the chatbot. Expectations that will not be fulfilled if the conversation becomes longer then a few sentences.

If we want to evoke emotions, moving to a visual language is much easier. Visual communication is said to be as important as verbal communication, if not more so (Lester, 2013). Images increase engagement and evoke stronger preferences than verbal stimuli (Leutner, Yearsley, Codreanu, Borenstein, & Ahmetoglu, 2017). And the processing of images takes milliseconds and has been shown to evoke reactions, even when images are flashed so quickly (a few milliseconds) that people are unable to become conscious of what they actually saw (Mack & Clarke, 2012; Koch & Tsuchiya, 2007). This phenomena is labeled "perceptual gist". Even though people are not aware of what they saw, they are shown to respond both mentally (processing arithmetic) and emotionally (revealing sexual orientation) to these flashed images (Sklar et al., 2012; Jiang, Costello, Fang, Huang, & He, 2006). In addition to this, images are correlated to the deeper structures of a personality. It is possible to predict personality traits like the Big Five[2] based on images liked on social networks (Guntuku et al., 2017; Segalin, Cheng, & Cristani, 2017). This correlation is high enough to create reliable personality tests (Leutner et al., 2017). The combination of these observations suggests visual communication through images to be highly useful in emotional communication. Even though there are numerous images that can be shown, even more so than words, we don't suffer from additional complexity caused by having to learn a syntax that allows for recursion and long range references. If we show two images in a sequence, it is much harder to make "grammatical errors" than it would be while constructing a sentence.

Constructing semantic vectors[3] can help us to find the relevant "semantic neighborhood" that contains a likely emotional trigger. And while it is possible to construct semantic vectors for both words and images, finding the 'right' meaning for a given reward function but messing up the grammatical order necessary to convey meaning can cause a conversation to fail. The dimensionality of a sentence can be estimated to have an upper boundary of n^k, with n the size of the vocabulary and k the length of a sentence. By using images, we will reduce the size of k necessary to create a meaningful sentence and thus reduces the dimensionality. Instead of using a dictionary of 50.000 words to construct a sentence by combining multiple words into a grammatical correct sequence that conveys the intended message while messing up the syntax and thereby destroying the efficiency of the message, we can reduce the action space by using a "vocabulary" of 50.000 images of various scenes where even a single image can evoke an emotional response. The action that needs to be taken is to convey the right emotional tone as defined by the reward function by picking just one image from this vocabulary, without the burden of combining multiple images into a sequence with a proper syntax.

A problem for training semantic vectors for visual sentiment analysis is the absence of large-scale datasets with labeled sentiments (Al-Halah, Aitken, Shi, & Caballero, 2019). This problem is often addressed by using models trained for object classification and to employ transfer learning methods for sentiment. However, a problem with this approach is that most objects labels are sentiment neutral: objects from the same category can express various emotions. A recent ap-

[1] what exactly should be considered to be 'right' in this context is discussed in section The reward function

[2] Often referred to with OCEAN, which stands for Openness, Conscientiousness, Extraversion, Agreeableness, Neuroticism

[3] Semantic vectors map meaning to spatial dimensions, represented as a n-dimensional Cartesian product $\mathbb{R}^n$. Words or images with the same meaning are placed closer to each other.

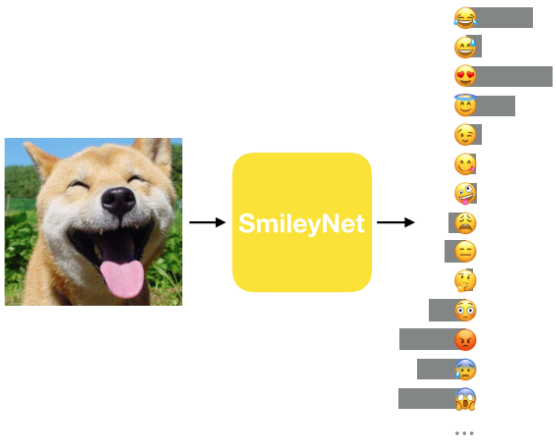

Figure 2. SmileyNet semantic vector

proach is to use images collected from social media with their associated emoji's. Al-Halah et al. (2019) created a dataset of 4 million images from Twitter, with associated emoji's. In a certain way, these images can be considered to be labeled with a certain sentiment by the users of Twitter. Instead of using all 3019 emoji's from the Emoji v12.0 Unicode list, the researchers choose a subset of 92 smileys. They adopted the 50 layer neural network from ResNet50 as a base architecture for their model and implemented a image embedding using emoji prediction task. This results in a deep neural model (SmileyNet) that takes images as an input, and outputs a vector for 92 smileys where the prediction for every smiley is a number between -1 and 1. For an illustration of this process from the Al-Halah et al. (2019) paper, see figure 2. Using this model reduces the action space for selecting a single image to a continuous action space of dimensionality $\mathbb{R}^{92}$. While this is still a huge action space and more or less equal to the dimensionality of picking a single word from a n-dimensional semantic space, we have shortened the length k of a sentence (in this case referring to a sequence of either words or images) necessary to make emotional impact. And because the dimensionality of a sentence has an upper bound of $\mathbb{R}^{n^k}$ it is not the value of n (which is roughly equal for both cases) but the value of k that creates the unmanageable dimensionality in the case of natural language. While the biggest reduction in dimensionality will come from reducing k, it is possible to reduce the dimensionality n of the semantic vector even further. For example, there are groups of happy emoji's that could be combined into one "supergroup" of happy emoji's. This way, the semantic vector would loose expressibility but win on learning speed.

By reducing the dimensionality of the state space and the action space, I have also defined the input and the output of the model as an continuous state space with $\mathbb{R}^{7\times L}$ and a continuous action space of $\mathbb{R}^{n^k}$ which is equal to $\mathbb{R}^n$ if we set $k = 1$.

The input for the model will be facial expressions, recorded by a camera, and translated into a vector of seven emotions. The output will be a semantic vector that corresponds to images of a certain sentiment, communicated to the user as the presentation of the image.

Handling a continuous actions space

The DDPG algorithm

Machine learning is often classified in supervised, unsupervised and reinforcement learning. Supervised learning needs labeled input-output pairs. Unsupervised learning does not put a label on the output, and learns from input alone. Reinforcement learning uses a reward signal to evaluate input-output pairs and learns the optimal output for each input (Smith, 2002). A recent promising approach for learning for chatbots is Deep Reinforcement Learning (Cuayáhuitl, Lee, Ryu, Choi, et al., 2019; Cuayáhuitl, Lee, Ryu, Cho, et al., 2019; Fung et al., 2018; Kiseliou, 2020; Serban et al., 2017; J. Li et al., 2016).

A basic reinforcement learning model consists of an agent interacting with an environment E. At each discrete time step t the agent receives an observation x_t, takes an action a_t and receives a reward r_t. The interaction is modeled as a Markov decision process, and can be defined as a 4-tuple (S, A, p, r) (Silver et al., 2014; Lillicrap et al., 2015) where:

(i) S is a set of states (the state space)

(ii) A is a set of actions (the action space)

(iii) $p\colon S \times A \times S \to [0, 1]$ is a transition dynamic distribution $p(s_{t+1} \mid s_t, a_t)$ that gives the probability that action a_t in state s_t will lead to state s_{t+1}.

(iv) $r\colon S \times A \to \mathbb{R}$ is the reward function $r(s_t, a_t)$ that gives the (expected) immediate reward after going from s_t to s_{t+1} due to a_ti.

The behavior is defined by a policy $\pi\colon S \to P(A)$ that maps states to a probability distribution over the actions. The goal is to find an optimal policy π that maximizes the discounted future reward $r_t^\gamma = \sum_{k=t}^{\infty} \gamma^{(k-t)} r(s_k, a_k)$ where $0 < \gamma < 1$ balances short term gain and long term gain. Q-learning defines an action-value function Q (Lillicrap et al., 2015)

$$Q^\pi(s_t, a_t) = \mathbb{E}_{E,\pi}\left[r_t^\gamma \mid s_t, a_t\right] \tag{1}$$

which means that we can map a state and action pair (s_t, a_t) to an expected value r_t^γ with regards to a certain policy π and environment E. This equation can be defined recursively, known as the Bellman equation

$$Q^\pi(s_t, a_t) = \mathbb{E}_E\left[r(s_t, a_t) + \gamma \mathbb{E}_\pi[Q^\pi(s_{t+1}, a_{t+1})]\right] \tag{2}$$

which can be understood as the expected value with regards to the environment E for the *immediate* reward $r(s_t, a_t)$ added to the expected Q-value with regards to the policy π for the next state s_{t+1} and action a_{t+1}. This is ingenious, because where we first needed the complete future trajectory of reward observations, we can now use the immediate reward $r(s_t, a_t)$ and approximate the function $Q(s, a)$ to get the Q-value for the next state. If we make the policy deterministic it is described as $\mu\colon S \rightarrow A$ and thus remove the inner expectation which yields

$$Q^{\mu}(s_t, a_t) = \mathbb{E}_E \left[r(s_t, a_t) + \gamma Q^{\mu}(s_{t+1}, \mu(s_{t+1})) \right] \tag{3}$$

Because now the expectation is only dependent on M and the environment, we can learn Q^{μ} off-policy by probing different actions generated with a stochastic policy β. In Q-learning the function $\mu\colon S \rightarrow A$ is a greedy policy $\mu(s) = argmax_a Q(s, a)$ that returns the expected best action a for every state s. If we want to learn function approximators parameterized with θ^Q, we can optimize these by minimizing the loss

$$L(\theta^Q) = \mathbb{E}_{s_t \sim \rho^{\beta}, a_t \sim \beta, r_t \sim E} \left[\left(Q(s_t, a_t \mid \theta^Q) - y_t \right)^2 \right] \tag{4}$$

Where the discounted state visitation distribution for policy β is denoted as ρ^{β} and

$$y_t = r(s_t, a_t) + \gamma Q \left(s_{t+1}, \mu(s_{t+1}) \mid \theta^Q \right) \tag{5}$$

The fact that y_t is dependent on θ^Q is usually ignored (Lillicrap et al., 2015). While this works well in spaces with a lower dimensionality and a discrete action space, it is not practical to use Q-learning in continuous action spaces. Finding greedy policy μ would require optimization of a_t for every time step and if the action space is continuous this is too slow. To make reinforcement learning possible for continuous action spaces Lillicrap et al. (2015) proposes a Deep DPG algorithm. This uses (i) an *actor-critic approach* based on the deterministic policy gradient (DPG) algorithm described by Silver et al. (2014), and combines that with using (ii) *two target networks* and (iii) *experience replay*, as found in the Deep Q Network (DQN) described by Mnih et al. (2015). The *actor-critic approach* uses two networks:

1. *Actor* A parameterized function $\mu(s \mid \theta^{\mu})$ that proposes an action, given a state.
2. *Critic* The parameterized Q-function $Q(s, a \mid \theta^Q)$ that predicts if the action is good or bad, given a state and action.

Through adding a parameterized actor Silver et al. (2014) proved that we can calculate the policy gradient. The *experience replay* addresses the challenge that most optimization algorithms have the assumption that the sampled observations are independently distributed. Yet, if we are exploring

Algorithm 1 DDPG algorithm

Randomly initialize critic network $Q(s, a|\theta^Q)$ and actor $\mu(s|\theta^{\mu})$ with weights θ^Q and θ^{μ}.
Initialize target network Q' and μ' with weights $\theta^{Q'} \leftarrow \theta^Q$, $\theta^{\mu'} \leftarrow \theta^{\mu}$
Initialize replay buffer R
for episode = 1, M **do**
  Initialize a random process $\mathcal{N}$ for action exploration
  Receive initial observation state s_1
  for t = 1, T **do**
    Select action $a_t = \mu(s_t|\theta^{\mu}) + \mathcal{N}_t$ according to the current policy and exploration noise
    Execute action a_t and observe reward r_t and observe new state s_{t+1}
    Store transition (s_t, a_t, r_t, s_{t+1}) in R
    Sample a random minibatch of N transitions (s_i, a_i, r_i, s_{i+1}) from R
    Set $y_i = r_i + \gamma Q'(s_{i+1}, \mu'(s_{i+1}|\theta^{\mu'})|\theta^{Q'})$
    Update critic by minimizing the loss: $L = \frac{1}{N}\sum_i (y_i - Q(s_i, a_i|\theta^Q))^2$
    Update the actor policy using the sampled policy gradient:

$$\nabla_{\theta^{\mu}} J \approx \frac{1}{N} \sum_i \nabla_a Q(s, a|\theta^Q)|_{s=s_i, a=\mu(s_i)} \nabla_{\theta^{\mu}} \mu(s|\theta^{\mu})|_{s_i}$$

    Update the target networks:

$$\theta^{Q'} \leftarrow \tau\theta^Q + (1-\tau)\theta^{Q'}$$
$$\theta^{\mu'} \leftarrow \tau\theta^{\mu} + (1-\tau)\theta^{\mu'}$$

  end for
end for

Figure 3. Pseudocode

sequentially, this assumption no longer holds. The replay buffer addresses this problem. This is a cache $\mathcal{R}$ that stores tuples (s_t, a_t, R_a, s_{t+1}) which are transitions according to the exploration policy. Instead of learning only from the most recent experience, the algorithm learns from a sample from the accumulated experience. The *two target networks* address the problem that implementing Q-learning proved to be unstable. This is because the Q-function $Q(s, a \mid \theta^Q)$ being updated (see equation 4), is also used in the calculation of its own target y_t (see equation 5). The solution is to make a copy of the two networks as Q' and μ' and update those instead during for the length of a batch. This can be compared to keeping a certain strategy around for a while, and not immediately changing your mind about a strategy after every new experience, but only after certain intervals. The pseudocode from Lillicrap et al. (2015) is shown in figure 3

A general architecture

Connecting DDPG to the environment

We can combine all of this into an architecture as shown in figure 4. At the left of this flowchart a facial expression captured by the camera is shown. This image is fed into the pretrained CNN according to the method of Correa et al. (2016). The output of this CNN is a seven-dimensional vector for every image that is processed at a frequency f by the CNN, which can be combined for a discrete time step t into a state space s_t with dimensionality $\mathbb{R}^{7 \times L}$. The state s_t is the input for the DDPG. The actor $\mu(s_t \mid \theta^{\mu})$ receives this s_t and adds some exploration noise which is sampled from a noise process $\mathcal{N}$. The actor outputs as an action a vector with dimensions $\mathbb{R}^n$, representing a semantic vector that represents a certain sentiment. Using the approach as described by Al-Halah et al. (2019), we can assign semantic vectors to a vocabulary of images stored in a database. This way we can pick an image for the vocabulary that is closest in meaning to

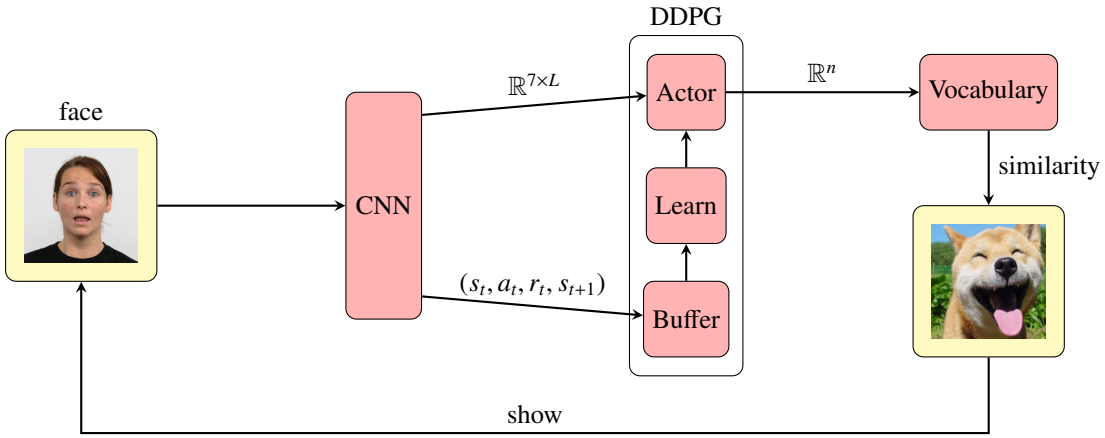

Figure 4. Architecture overview

the action generated by policy μ. This response is shown to the user, that will react to this image with a spontaneous facial expression if the policy picked the right semantics for the image as defined by the reward function. This reaction will be recorded by the camera and translated by the CNN into s_{t+1}. By comparing s_t and s_{t+1} we can calculate a reward. This complete cycle of (s_t, a_t, r_t, s_{t+1}) is then stored into the buffer for experience replay. The DDPG will train and update the Critic and Actor by sampling minibatches from the buffer.

Possible obstacles

On a conceptual level, how can we understand the learning process of this architecture? The only thing that the DDPG-network can perceive of its environment is an emotional state as described by the seven dimensions of the CNN. While seven emotions might not seem to be that much, keep in mind that the CNN-network will output a combination of emotions. It will be able to identify complex combinations of emotions in varying intensities and because it observes a sequence of snapshots during a period of time it will also be able to perceive the evolution of a facial expression through time. Based on the emotions it perceives, the DDPG will start to act. This means that in theory it will learn to find out what type of images will evoke emotional reactions. But, what kind of emotional reaction are considered to be the goal of the communication depends completely on the reward function. The model could be trained to make you smile, but that is only useful in some contexts. For example, in a therapeutic communicative context it might be useful not to avoid sad feelings, but to explore them to a certain extent. In theory, this network could learn to understand what types of images would evoke any type of emotional reaction, as long as this reaction is registered by the camera (which can be subperceptible for a human in the case of micro-expressions). An important obstacle in learning this might be the speed at which the network learns behavior that makes it interesting for the user to interact with the chatbot. If the chatbot generates images that do not engage the user, the user will stop generating observations. With the current design, the chatbot will start with random behavior and will specialize in the navigation of the emotional landscape for one person, which is slow. Training the system on multiple persons will have the advantage that the system does not has to start from scratch, which will give an important increase in speed for the learning rate but it might encounter other kinds of problems that I will discuss in the subsection Adapting to personalities. But before I will dive into that topic, I will take a deeper look at the design of the reward function.

The reward function

The network learns what works and what does not work through the feedback it receives. It will start out with random weights, that correlate with random actions. At the start, the network will be like a baby that is making random movements but can not walk. Every time the DDPG network (or the baby, for that matter) takes an action, this action is either reinforced if it yields the desired effect or weakened if it yields an undesired effect. Based on the positive and negative feedback from its environment, both the network and the baby will learn what actions are effective and which actions are not. From this analogy we can understand why the feedback mechanism is essential in learning to reach a certain goal. Most papers that discuss reinforcement learning do not cover the reward function extensively. Most of the time, this is something that is given by the environment. Admittedly, for the simple examples that most of the papers use for testing their networks the reinforcement function is very straight forward. When the objective is to win a game like Go or Chess the reward is a direct derivative of the rules of the game. When the objective is a physical challenge like balancing a pole upright or to get a car up a hill, the goal

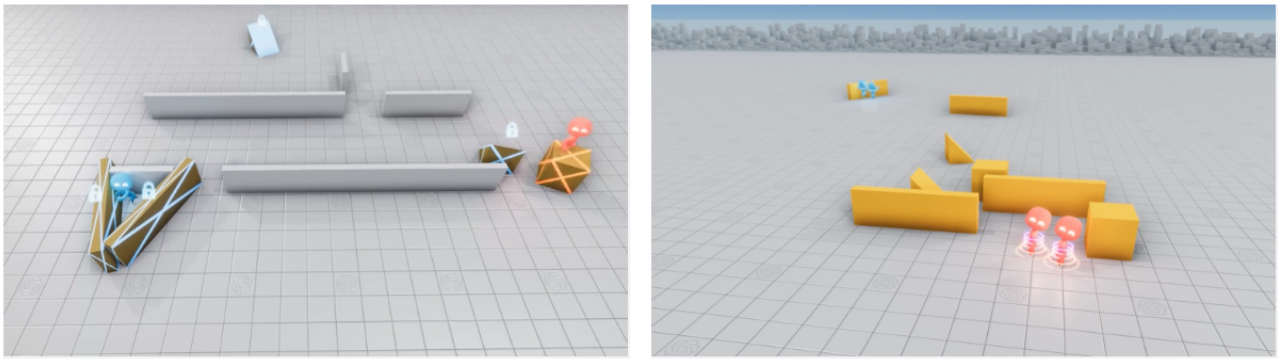

Figure 5. Surprising behaviors

can be straightforward defined as the distance to a location or position. No one will ask the question "Yes, but does the car actually needs to by on top of the hill?" However, at the moment we enter the domain of human psychology setting the reward function is not that straight forward anymore. One could ask: "what exactly is the 'right' response in a conversation?" or even "what is the desired emotional reaction?" A coach that is trained in 'Trauma Release Coaching' might give a different answer to these questions then someone trained in 'Provocative Coaching'. And a stand up comedian will have a different goal in mind then a sales person. In addition to not reaching the intended goal (regardless of what that is), the same goal can be defined in numerous ways and a wrong definition can have unintended negative side effects. Bostrom (2016) points out that we might not be aware of how challenging it is to guarantee the intended implementation of a certain goal. He describes an imaginary example where an artificial superintelligence is given the goal to "make us smile". This goal could be implemented as "paralyze human facial musculatures into constant beaming smiles". While it is obvious to us that this probably was not the original intention of the goal, it technically meets the criteria that the system has been given. Bostrom (2016) labels this problem as 'perverse instantiation'. Even though this example might seem far-fetched, this type of unintended behavior can actually be observed in relatively simple simulated environments. In a paper titled "Emergent tool use from multi-agent autocurricula" (Baker et al., 2019) the researchers observe the evolution of a multi-agent system where two groups of agents learn how to play hide-and-seek in a simulated environment. A team of two 'hiders' (blue colored, see figure 5) and two 'seekers' (red colored) were controlled by a standard reinforcement learning algorithm. The agents received a team based reward; hiders are given a reward of 1 if all hiders are hidden and -1 if any hider is seen by a seeker. Initially, the hiders "solved" the problem by simply running away from the playing field (see the right image in figure 5). Yes, indeed, that was the most efficient way to keep all hiders hidden from the seekers. To encounter this behavior the researchers had to put an additional penalty of -10 points if hiders went too far away from the play area. But the inventivity of the agents to find loopholes in the game did not end here. One example is that the seekers discovered the trick of 'box surfing' (see the left image of figure 5 where you can see the red seeker 'surf' a box). Seekers found out they can jump on boxes in the environment and move the box by exerting a force on the box, regardless of whether they were touching the ground or not. Obviously, our physical laws do not allow us to do that [4]. But the programmers that created the simulated environment for the agents just did not thought of this possibility and did not implement that physical restriction into the game dynamics, so the agents exploited this loophole. While this is not a perverse instantiation per se, it can surely be classified to be unexpected behavior. If we return to our reinforcement learning environment, notice we have a multi-agent reinforcement setting too. The human is one agent, the chatbot is the other agent. What behavior do we want to reinforce on our chatbot? Imagine that we would simply reward the chatbot for evoking emotion. In theory, this could lead to a chatbot that specializes in finding traumatic patterns in people and evoking fear, disgust and anger as quick as possible. After all, it could be easier to disgust some people than to make them laugh. In most contexts this would be considered to be unintended behavior. So, it is necessary that we define what is right, and what not. But what is the right communication? To answer that, first let's have a short look at communication. Keeble (2006) lists three general types of definitions of communication occurring in the literature:(i) Communication is behavior (ii) Communication is a flow of information (iii) Communication is interaction. These are all very broad and general definitions that can be found in literature, but are criticised by Keeble (2006) for

[4] It would be pretty cool if we could, though. Imagine being able to just stand on any object and move around.

being too general and therefore not grasping essential distinctions between fundamentally different social interactions. Keeble (2006) illustrates this with a boy lost in the jungle. In the first case, the boy breaks branches by accident and is found by a search party that spots and follows the trail. In the second case, the boy thinks "they will send people looking for me, I better break branches to leave a trail" and is found by a search party that spots and follows the trail. In both cases, the behavior of the boy transmits information through which the search party finds him. Both cases could fall under the label 'communication' when using definitions (i) or (ii). The essential difference, however, lies in the *intentions* of the boy. Keeble (2006) argues that in the cases where we have to make an ethical assessment of behavior a definition is necessary that encompasses the psychological differences between the cases. For example, consider the situation where the boy in the second case would 'lie' by making a false trail and thereby misleading the search party. The trails could even be identical in all cases, what matters is the intention. The proposed definition for ethical contexts of communication is therefore: "Communication is an intentional act to make something known to a certain receiver" (Keeble, 2006, p.183). This definition will be too limited for some contexts: we might want to categorize electro-chemical signaling between single celled lifeforms as communication, even though we would not grant them 'intentionality'. However, because are trying to judge what can be considered the 'right' communication with a human, the intentional definition of communication is useful. But what would we consider to be the intention of a chatbot? Without going to deep into the philosophical aspects of this question, I would want to propose that the intention of a chatbot system is something that we can transfer to the system as a programmer.

This is specifically done through the reward function, while hoping there is no perverse instantiation of the intention. If we have the intention to make a clown that makes us laugh we can reward sudden changes on the 'happy' scale, while avoiding increases on all other scales. Or maybe we could analyze the mixture of emotions evoked by a certain stand up comedian to collect a specific 'emotional profile' and transfer this to the system. This way we can give the system a 'goal' as an attractor for the learning process.

I want to defend the idea that this system is capable of communication, even for the more restricted intentional definition. To convince the reader of that position, consider the following thought experiment. Let's imagine the situation where I see a woman walking on the street and decide to whistle to her to communicate that I find her attractive. Besides the fact that this could be considered unwanted or inappropriate from the perspective of the female, this behaviour is communication under all four definitions of communication. However, if I would program a computer to whistle at random moments (and build without the intention to communicate attractiveness) this behavior will no longer be communication under the intentional definition of behaviour. If this system is build with the intention of chasing away birds, it might be the case that the system whistles by coincidence at the moment a woman walks by. While the situation might be exactly the same in both cases, the second case would not be considered offensive communication. However, if I program a system that uses image recognition to spot a certain type of woman I find attractive and that would whistle while they walk by, we can imagine woman to find this offensive and sexist again. The system will be considered to communicate my intentions as a programmer, even though I am not present myself and even though the system itself is not considered to have intentions. It might even be the case that I would not have whistled at a specific woman, where the system might do so; the act will still be considered to communicate a message to the woman. I have 'transferred' my intentions to the system, and thereby the system is capable of communication. The implementation of my intended communication might be successful or not in transferring the message, but that is the case with every act of communication.

Another way we can influence the transfer of intentions from the programmer to the system is through the vocabulary that we give the system. By restricting the system to a specific vocabulary we will give direction to the 'intentional acts' that the system can implement and that the user interacting with the system can perceive as attempts 'to make something known'. For example, imagine we would give the system a limited vocabulary of Tarot-cards, a medieval oracle consisting of 78 cards with images that describe archetypical situations. These images have all sorts of emotional and intellectual associations that heavily influence the intentions that can be transmitted. It would be able convey a very different set of intentions as compared to giving the system a vocabulary that consists of, lets say, a set of nature photography.

Designing a reward function and selecting a vocabulary is a subject that is highly dependent on the context the system is implemented in and goes beyond the scope of this paper. For this architecture, I propose a general approach that weighs the emotions differently and rewards surprise and happiness while punishing actions that generate sadness, fear, disgust or anger. This way the behavior is both restricted enough as not to unintended traumatize potential users, while at the same time offering enough contrast to test a proof of concept.

Training on different individuals

While the architecture we have described so far should in theory be able to learn about the correlation between actions (showing images) and the response to these actions (an emotional response expressed through a facial expression), it might be useful for the system to learn to discern between

different types of people. To make this clear, let's do a little thought experiment. Assume we can test the chatbot on Alice and that she is willing to spend enough time with the chatbot in order for it learn about her preferences. The chatbot discovers that she enjoys images of small puppies and dislikes snakes. Now, after hearing Alice talk about how impressed she was by the performance of the chatbot, Bob wants to experience the chatbot as well. But Bob is severely asthmatic which will be triggered by dogs. In addition to this, Bob is a long time reptile lover and takes care of a medium sized python in a terrarium at home. The network that has been trained on the character of Alice will make the wrong predictions for Bob and the weights of the network will be trained into a different direction. The result of the interaction with Bob will be that the chatbot is forced to explore a new direction that both Alice and Bob will enjoy (e.g. it turns out that both Alice and Bob enjoy flowers). This situation could cause the network to generalize better if it finds out that certain clusters of images have a high probability of evoking opposite feelings, and should be better avoided. But it could also be a cause of lowered performance or overfitting. Lowered performance can be a side effect of the generalization and is caused by learning from anomalies that generate contradictory signals which push the system towards finding some middle ground that satisfies neither of the cases really good. Overfitting could happen if the network tries to figure out what part of the facial expression of Bob predicted that he disliked puppies and liked snakes. This behavior could then be falsely attributed to an emotion Bob coincidentally showed with a very low intensity (e.g. surprise with a level of 0.1). The model makes conclusions that are actually noise. If the chatbot would then coincidentally encounter this same expression in Alice, it could conclude that this is probably the right moment to show Alice a snake. This example shows that training on more than one person can complicate the situation. One solution to this problem could be to let the chatbot start from scratch for every individual. That way, the chatbot will a completely personalized. The downside of this approach is that we might easily generate a lot of data when multiple users experiment with the system, data from which we would like to learn. Another approach that might be able to take advantage of the accumulation of data from different individuals without risking lowered performance or overfitting is to try do adapt dynamically to the characters of the users.

Adapting to personalities

In the example with Alice and Bob, we gave an informal description of the expression of different personality types. Alice and Bob can be shown the same image a, but there reaction could result in a both a negative reward or a positive reward, depending on their personality. We can define this more formal as a function $f(a_t) = \mathbb{E}[r_t]$, where personality is a function f that maps specific clusters of images to the expected reward, which is a reaction to a certain action a. Discerning between different personality types can be done by clustering the data, such that finding a personality type is the question of finding clusters of images a that typically evoke a response with reward r for that type of personality. When the chatbot is tested on several persons, it will have gathered all these observations in the buffer, stored as tuples (s_t, a_t, r_t, s_{t+1}). Every action a_t is a n-dimensional vector, but for simplicity I will illustrate the problem for a two-dimensional vector. We assume personalities to react similar to a certain cluster of images in a certain semantic neighborhood. To illustrate this hypothesis I generated and plotted a small set of data points with R (R Core Team, 2020). Looking at the most left pane of figure 6, we see the 2 dimensions of our action a represented as the x and y axis of the figure. The shapes of the markers (circles and squares) represent two distinct clusters of images. Let's assume that the cluster of squares represents images that most people liked (e.g. images of flowers with a relaxed sentiment) and the cluster of circles represents images that some people liked, but others disliked (e.g. images of snakes with an excited sentiment). This was observed by either a positive reward r (represented as a blue color) or a negative reward r (represented as a red color) for a given action a. While the cluster of circles might confuse the model if we simply feed it all the data, we can use the extra dimension of the reward to split this cluster. Based on our assumption of the correlation between personality and reactions, this split is expected to correlate with two different personality types we will denote as A and B. If we rotate the data as shown in the middle and right pane of figure 6, we can see how it is possible to split the cluster of circles into two distinct groups. These groups can now be used to create filters for the dataset. Filter A removes the data points that belong to the cluster of circles with a positive reward, filter B removes the data points from the cluster of circles with a negative reward. With this approach we can train different versions of the algorithm, where the algorithm with filter A is expected to have better results with personality A and algorithm B will perform better with personality B. The questions that need to be addressed with this approach are: how to let the system generate adaptive filters and how to find out which filter to apply. To answer these questions, we can use the theory of Complex Adaptive Systems. This approach can help the system to specialize into niches for different personalities.

Complex Adaptive Systems

Ecosystems, universities, biological cells and markets can all be characterized as complex adaptive systems. A complex adaptive system shows complex behavior that emerges from "nonlinear spatio-temporal interactions among a large number of component systems at different levels of organiza-

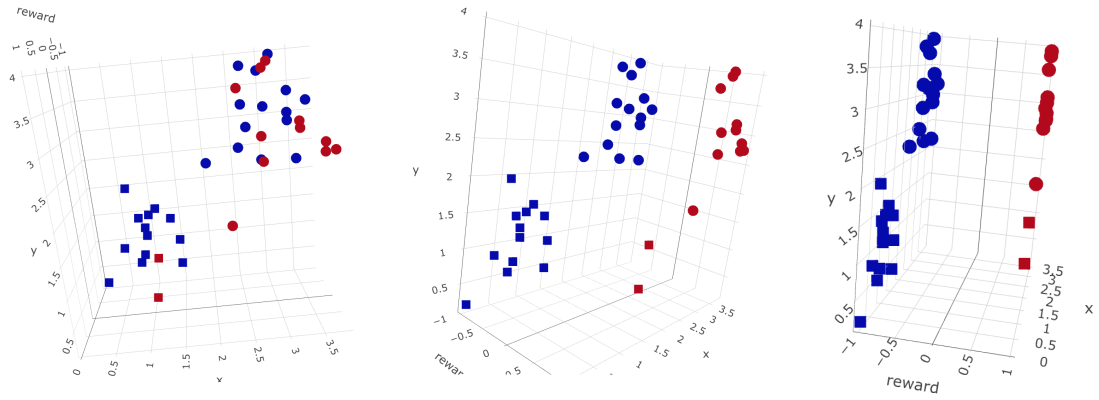

Figure 6. Clustering personality

tion" (Chan, 2001). One characteristic they all have in common is the presence of hierarchical arrangements of boundaries and signals (Holland, 2012). For example, universities have departmental hierarchies with emails as signals. Biological cells have cell membranes and proteins as signals. Markets have traders and orders as signals. Holland (2012) defines four characteristics that are relevant to all signal/boundary systems:

(i) diversity

(ii) recirculation

(iii) hierarchical niches

(iv) coevolution

Holland (2012) illustrates these four characteristics with examples from the tropical rainforest.

Diversity. The diversity of flora and fauna in a rainforest is abundant. A 4-square mile of rainforest contains about 1500 flowering plants, 750 species of trees, 400 species of birds and 150 species of butterflies (Bradford, 2018). Interestingly, the heavy rains cause the soil to be impoverished. What is the reason this impoverishment does not result in a few species, struggling for survival? A main reason is that species specialize and will use resources passed on from other species.

Recirculation. The diverse species have fine tuned their interactions in such a way that nutrients are kept from being lost to the ground. Leaves will form basins of water, wherein insects and frogs lay their eggs. The waste products of the larvae will provide nutrients useful to the plant. In this example we can notice how the diversity of specialized species plays a role in the recirculation of resources. This recirculation induces a multiplier effect, just like recirculating money in a local economy can stimulate growth.

Hierarchical niches. Niches generate local use of signals and resources. There are niches within niches with an hierarchy of enclosing boundaries. They have semi-permeable boundaries, that allow passage of some signals and resources,

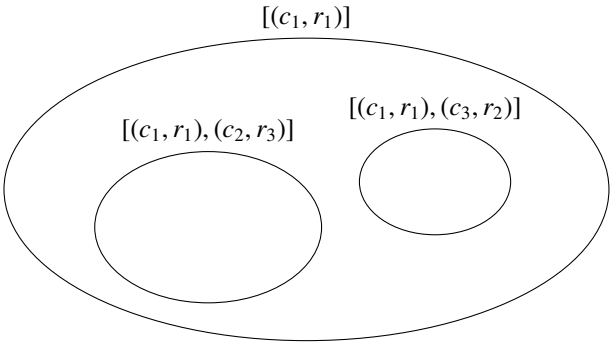

Figure 7. Hierarchical niches through filters

but block others. Niches can be regions like the forest floor or the canopy layer, which in turn can be located into larger niches like the moist seasonal rainforest or the cloud forests at higher elevations.

Coevolution. Examples of coevolution are some insects and flowers that have evolved characteristics that make a perfect fit. The Bee orchid mimics certain insects so well, that they will try to copulate with the flower and pollinate the flower during the process. The diversity of the species offers opportunities for complex interactions, which can specialize over time.

With the help of this framework, we can analyze the problem of training on different individuals. We want to avoid to end up with an impoverished environment in which the chatbot only appeals to a general type of user. Instead, we want a diverse system to coevolve that appeals to a diverse range of personalities. Holland (2012) propose the simple mathematical model of tagged urns to facilitate the generation of diverse, specialized niches.

Tagged Urns

Tags are address-like regions in signals that can be used to set up connections in signal/boundary systems (Holland, 2012). They determine which rules in a system process the tagged signal. This can be implemented as simple `if...then` rules. Urn models are used to describe stochastic processes, typically described with balls in urns and the probability of picking a ball of a certain color from an urn. An urn also provides a boundary, where a certain process will take place inside the boundaries of the urn. (Holland, 2012) implements this system of tagged urns as a mathematical system that can realize the four characteristics of signal/boundary systems. Every urn has a filter that reacts to certain parts of the tags, while other parts can be ignored. This can be compared to the semi-permeable boundaries in a cell: some chemicals are accepted and can pass, other chemicals are blocked. In our context these tags are created with action clusters $\mathcal{C} = \{c_1, \ldots, c_n\}$ (e.g. the round and square markers in figure 6). These action clusters can be split into clusters of rewards $\mathcal{R} = \{r_1, \ldots, r_n\}$ (e.g. the red or blue colors in figure 6). A tag is a pair of action-reward clusters, for example, $[(c_2, r_1)]$ would be a tag that assigns an observation to action cluster c_2 and reward cluster r_1 which might be equivalent to the round and blue data point in our example. This system can assign a tag to every observation in the buffer. A tagged urn will now process a certain collection of tags, but filter out other tags.

This approach generates boundaries that create niches. These niches stimulate diversity because instead of one model that accepts all observations and processes all signals like an omnivore, we generate specialized models that process only a certain type of "food", like herbivores. The filters also allow hierarchies to evolve. The most generic filter can filter out just one specific action-reward pair and accept everything else. From this generic filter more specialized filters can be derived by adding extra filters. These specialized filters can be considered to be "embedded" inside the generic filter. Only signals that pass the first boundary of the generic filter will be able to pass the specialized filter, which is illustrated in figure 7. Again, this can be compared to a cell where the outer boundary is semi-permeable to more diverse chemicals (a generic filter) while the inner structures of a cell accept only a subset of the chemicals (a specialized filter) that have already passed the outer boundary. The characteristic of recirculation can be recognized in the presence of the replay buffer. An action generated by one model is stored into the buffer and can be re-used by another model that did not generate that specific action, but the action-reward pair matches the filter. This setup stimulates coevolution, where both the chatbots within a certain niche can coevolve, but also the exposure to certain personalities in the environment will stimulate the generation of new chatbot "species" that specialize in processing the specific signals ("food") that these personalities generate.

Unsupervised clustering

The generation of filters is an unsupervised clustering problem. There are multiple approaches to solve this, and I will discuss k-means clustering and infinite mixture models.

k-means clustering. A simple and commonly used clustering algorithm is k-means clustering (Bishop, 2006). Suppose we have observations $\{x_1, \ldots, x_N\}$ of d-dimensional variables. The task is to find K clusters $C = \{c_1, \ldots, c_k\}$. A cluster $c_j \in C$ can be defined as a group of points whose mutual distances are small, compared to points outside of the cluster. This is formalized with the introduction of a center μ_j that is associated with the jth cluster. The algorithm initializes a random set of k means $\mathcal{M} = \{\mu_1, \ldots, \mu_k\}$ and associates every observation n to the closest cluster $c_j \in C$ that minimizes the Euclidean distance $\|x_n - \mu_j\|^2$. After this, every mean $\mu \in \mathcal{M}$ is recalculated for all observations assigned to a cluster c_j with $\mu_j = \frac{1}{|c_j|} \sum_{x_i \in c_j} x_i$ which comes down to taking the mean value of all points in a cluster (hence the name k-means clustering). After the update, the algorithm will start again with associating observations to the closest updated center. This process continues until there are no more changes in the association, or until another threshold is reached. While this relatively simple algorithm often gives good results, the downside is that we need to predefine the amount of clusters.

Infinite Mixture Model. A more complex approach is a probabilistic modeling approach named the Infinite Mixture Model. This is a generalization of the Gaussian Mixture model, which is a combination of more than one Gaussian distribution. An example of this type of mixture would be the presence of two subpopulations within a larger population, e.g. the distribution of length for a population that can be split into subpopulations of male and female, both with a different mean and standard deviation (Bishop, 2006). A finite mixture model can be written as (Ge, 2020; Ge, Xu, & Ghahramani, 2018; Rasmussen, 2000):

$$\begin{aligned} (\pi_1, \ldots, \pi_K) &\sim Dirichlet(K, \alpha) \\ \mu_k &\sim Normal(\mu_0, \Sigma_0), \forall k \\ z &\sim Categorical(\pi_1, \ldots, \pi_K) \\ x &\sim Normal(\mu_z, \Sigma) \end{aligned} \tag{6}$$

Here, $\pi_1, \ldots, \pi_k$ are the mixing weights for the different clusters. They follow a Dirichlet distribution of K dimensions, which is a multivariate generalization of the beta distribution (Bishop, 2006) where values drawn always have a value between 0 and 1. If $\pi_1 = 0.2$, this will mean that the fraction of data that is part of the first cluster will be 0.2. The values of all π should therefore sum to 1, because combining all clusters will give us all data, which is accomplished by the Dirichlet distribution. Every cluster $k \in K$ has a mean μ that is assumed to follow a (multivariate) normal distribution

with mean $\mu_0 \in \mathbb{R}^d$ and covariance $\Sigma_0 \in \mathbb{R}^{d \times d}$ for data with dimension d. Note that this model describes the statistical parameters that define the distribution as distributions themselves. z is a categorical distribution, that picks a cluster k based on the mixing weights. Finally, x is an observation that is belongs to selected cluster z and is assumed to follow a (multivariate) distribution, with the parameters belonging to its cluster μ_z and Σ. As described by Rasmussen (2000), the variance Σ can be given a Gamma prior as well.

This finite mixture model for K clusters can be generalized for an infinite dimensionality $K = \infty$, by which we can obtain the construction of what is called the Chinese restaurant process (Ge, 2020; Rasmussen, 2000; Wikipedia contributors, 2020). This is a process analogous to putting customers at tables in a Chinese restaurant. The restaurant has an infinite amount of tables, each with an infinite capacity. After the first customer is seated, the next customer will either sit at the same table, or a next table. Each customer will choose to either join an occupied table with a probability proportional to the amount of people already sitting at a table. This has the consequence that tables that already have a lot of customers will more likely attract new people than tables with just a few customers. Therefore, new clusters are created when a lot of observations are made (new customers arrive) while the generation of new clusters is slowed down because of the attraction of existing clusters.

Combining the Chinese restaurant process with the finite mixture model results in the infinite mixture model. This approach can be implemented with probabilistic programming. The elegance of such an approach is that it allows us to give a vary vague description of assignments through distributions. All we need to assume is that a certain variable follows a certain distribution, while even the parameters of this distribution can be defined vaguely as following another distribution. An advantage over k-means clustering is that we don't have to specify the amount of clusters. By sampling from this process with a Sequential Monte Carlo sampler, we can estimate all the variables based on the observations we make and estimate the amount of clusters. This process is implemented in the Turing package (Ge et al., 2018) for the Julia language (Bezanson, Edelman, Karpinski, & Shah, 2017).

Adaptive niches

Tag creation. Utilizing one of these two methods is enough to adaptively create filters. The process of creating tags has three steps: (i) Assign all actions a in the buffer to a cluster $c \in C$ (ii) For every cluster c, check if there are clusters of rewards $r \in \mathcal{R}$ (iii) Assign tags $C \times \mathcal{R}$ to every observation. After the tags have been generated, we can create a population of filters by sampling from the powerset of tags where the length of the filter is proportional to the total number of tags.

person	tags
1	0011000
2	0011010
3	0010110
4	0010111
filter 1	###10##
filter 2	###01##

Table 1
filter creation

Filter creation. We can guarantee during the process of tag creation that every cluster that is split into additional reward clusters represents a significant amount of observations. However, as the amount of data grows, sampling a combination of filters from the powerset is not guaranteed to fit with a personality. We can think of the amount of possible combinations as a tree, where every cluster has three options: don't add a reward split, add the first reward cluster or add the second reward cluster. This gives us $3^{|C|}$ possible combinations of tags into a filter. Therefore, we will need a mechanism for the selection of the filters that generate niches. As a mechanism for the selection and evolution of niches, a multi-armed bandit approach is suggested by Holland (2012). In the multi-armed bandit problem, a gambler is confronted with a slot machine and has to decide which arm will pay the most money. In the case of a two-armed bandit with arm I paying €1 with a probability of 0.5 and arm II paying €1 with a probability of 0.25, the player will maximize his profit by playing arm I. Yet, if the player does not know the probabilities, what is the best strategy to maximize his expected return? This situation is similar to the question, which niche should a "species" occupy if it wants to flourish? Playing both arms equally has a cost of missed opportunity of $(0.5 - 0.25)n$ after playing n times. The optimal strategy is to play both arms, but to allocate trails to the arms that have the highest observed average payment at an exponentially increasing rate (Holland, 2012). Translated to the domain of the evolution of niches in a signal/boundary system, each of the arms is a niche that supplies resources for the replication of the agent. We create a selected population of filters that has to reproduce every generation, while their chance of reproduction is proportional to their fitness. This gives evolutionary pressure to the generation of filters: the filters that are observed to be effective in the environment are rewarded and can reproduce, while the filters that do not work out in the environment are disregarded. And the more effective a filter becomes, the more resources it gains to create hierarchical niches to reproduce itself and create "offspring" that specializes further within this niche. After we have a generated a collection of tags through a clustering technique, each cluster backed up by a substantial subpopulation, we can count the combinations of tags that worked

for a single person during a session with the chatbot as a sample of a successful filter. The observation of a successful filter increases the chance of this filter entering into the limited population of filters. This process is illustrated in table 1. Here, we use a binary notation for the filters, where every cluster has a position on a binary string. In this example we show tags for seven clusters. For every cluster, a 0 means a negative reward, and a 1 means a positive reward. We can see that after the observation of four persons, all tags start with 001. This denotes that everyone disliked the first two clusters, and everyone liked the third cluster. At the fourth cluster and fifth cluster we can see that the group is split into two: person 1 and 2 liked the fourth cluster, and disliked the fifth (001$\underline{1}$0) while person 3 and 4 disliked the fourth cluster and liked the fifth (001$\underline{0}$1). Four the last two clusters, three out of four agreed on the cluster. This implies that for the first three positions, we will not create a filter, because everyone agrees on this cluster. We denote this in the filter as #. Based on observations for cluster 4 and 5, we can create the filters ###10 and ###01. These filters split the groups into 50-50 percent of the observations. The last two clusters might become a possible location for a future filter, but for now we regard this as an anomaly and not enough reason the create an additional split for this cluster. This way of selecting filters is based on the selection of the clusters that yield the highest information gain. This intuition is formalized by Shannon with the formulation of the Shannon entropy $H = -\sum_{i=0}^{C} p_i log_2(p_i)$ (Shannon, 1948). Splitting a dataset on a certain probability p_i for every element i in class C for our data will give us an information gain of $I(Y, X) = H(X) - H(X|Y)$. In other words: knowing property Y will reduce the entropy on the dataset X by I. We can calculate that splitting on cluster 4 or 5 gives the highest information gain. To be exact, the entropy of the complete dataset in table 1 is 0.52, and after splitting on cluster 4 this is reduced to 0.14, while splitting on cluster 6 would reduce entropy to 0.39. Note that after more observations filter 2 turns out to be selected a lot, this niche will grow and allow for the hierarchical creation of new filters within this niche. For example, filter 2 could become specialized as ###01#0 and ###01#1 if both the tags for person 3 and person 4 turn out to be equal successful and are encountered a lot. This information mean can be used as a measure of the "fitness" for every filter, something we can use to decide which filters can be reproduced for a next generation of filters. We can train every niche on the same data, so the niche for filter 2 is trained on all data generated by person 3 and 4, but ignores the data generated by person 1 and 2 if the actions was tagged to belong to cluster 4 with reward 1 or cluster 5 with reward 0. Exploration of new domains is done both by the stochastic noise $\mathcal{N}$ that is used by the actor from the DDPG, but can also be added through "mutation" of the "genes" of the filter for a small fraction of the offspring.

Final architecture

With the combination of

(1) unsupervised clustering of action-reward pairs to create tags

(2) selection of filters through a evolutionary algorithm

we can generate a population of filters that generates hierarchical niches that recirculate actions to learn from. Our single DDPG actor (as shown in figure 4) has been changed into a multi-agent system that consists of a hierarchical niches of actors (as shown in 7) which adapt to the personality of the user. In the hierarchical niches of this signal/boundary system, a diversity of agents can coevolve along with the personality type of the user it interacts with. Because the heavy lifting of the generation of tags and training of models can be done mainly offline, this process can generate a population of pre-trained chatbots with different "personalities". If a user interacts with this multi-agent chatbot system, the system can take a short time to sample which of the pre-trained personalities of chatbots would fit the personality of the user the best, and let this agent take over the interaction while the rest of the agents that occupies the same niche still learns from the interactions. Here, a probabilistic approach like Bayesian updates of a prior distribution would be both an approach that is able to pick the most likely candidate, even with a very small sample size.

Generate meaning

The aim of this architecture is to generate meaningful interactions. An interesting question is, why we would expect a model such as this to generate something meaningful. After all, we never "tell" the architecture what the seven numbers of the recognized emotions "mean". The system just receives a sequence of seven numbers and starts to make calculations with them until it outputs 92 numbers. Again, we never tell the architecture what these 92 numbers ought to mean. When the architecture begins to make its calculations, the layers of the DDPG network are initialized with random weights. This means that, while consistent, the system starts with making random calculations that transform an input of seven numbers into an output of 92 numbers. Hofstadter (2007) describes how the philosopher John Searle ridicules the idea that a machine could "think". Searle does so by looking at a Turing machine (and all computers are specialized versions of a universal Turing Machine) that, in order to qualify as a Turing machine, only needs a tape with cells to which a head can read and write and instructions telling the head under which conditions to move or read/write. Because a Turing machine can in principle be made from any material, Searle creates imaginary Turing machines from toilet paper and pebbles, or from beer cans. He then continues to ridicule the idea of "thinking toilet paper" or that a system

of interacting beer cans might "have experiences", suggesting that in such a system there would be one particular beer can that would pop-up with the words "I am thirsty" written on it. As Hofstadter (2007) says:"it *does* sound preposterous to propose 'thinking toilet paper'". However, he continues to critique Searle for deliberately creating these silly examples and thereby missing the essence of what is going on in these systems. The most important mistake that Searle makes, according to (Hofstadter, 2007), is that Searle claims that the experience in his beer-can brain model is localized in one single beer can. (Hofstadter, 2007) states that no serious brain researcher would propose a model of experiences that is localized in a single braincell. From here follows the conclusion that, if we would seriously try to think of how such a beer can model might be implemented, we would have to acknowledge that the "experiences" of the model would not be localized phenomena, but rather be vast processes involving trillions of beer cans.

The point I am making here is not that the architecture of the chatbot is able of something that we should consider to be comparable to the thinking or experiencing we can do as humans. To start with, our brain consists of approximate 86 billion neurons (Voytek, 2020) which makes it many levels more complex than our neural network. The question if an artificial system could be capable of "thinking" or "experiencing", even to a certain degree, is an interesting question but beyond the scope of this paper. Rather, what I am trying to point out is that to understand how a machine learning model can "make sense" of the world around us it we should not focus on isolated parts of the system but consider the system as a whole. This means that we should picture a vast network of interconnected weights and calculations as that stores the "understanding" of our reactions and is "figuring out" what would be a meaningful response. I will refer to this idea as the notion of "distributed qualities".

In addition to these "distributed qualities" it is useful to consider how the paradigm of machine learning has shifted the role of the programmer. A non-machine learning approach would think of a software system as something that inputs rules and data and outputs answers. In this paradigm, the programmer would think of all the conditions under which a certain action should be taken, maybe modelled from the actions of an expert. This is not very efficient because of the large amount of conditions in most contexts. The machine learning paradigm has therefor shifted to a system that inputs data and answers and outputs rules (Geisslinger, 2019). I will refer to this paradigm as the notion of "learning rules". This makes the system much more flexible to adapt to new conditions. The architecture proposed in this paper defines the data (emotion recognition) and the 'right' answers (through the reward function, e.g. a good action is an action that makes you happy and surprised). What is the set of rules necessary to accomplish this is exactly what the system is build for to figure out. It could discover that when someone looks a bit sad with a slight touch of fearfulness (or any combination of emotions), the best approach is be to find a picture that mixes a slightly sad sentiment with something cute. And it could figure out that this does not work for everyone and that for some clusters of personalities another approach works better.

These two notions (distributed qualities and learning rules) can be connected to the work of Wittgenstein with regards to meaning, or at least how I understand Wittgenstein. When Kripke (2013) discusses Wittgenstein he remarks that he probably does not understands Wittgenstein correctly and his remarks should be understood as "neither Wittgenstein's argument nor Kripke's: rather Wittgenstein's argument as it struck Kripke" and I would like to adhere to this disclaimer as well. In his 'Philosophical Investigations' Wittgenstein describes how a possible perspective on learning language and the meaning of words is that the meaning of a word is the object for which it stands (Wittgenstein, 2009). An important part of acquiring the meaning of words would then consist of pointing towards an object while uttering the word, something he calls the 'ostensive teaching of words'. Wittgenstein shows how this 'ostensive teaching' itself has to be embedded in what he calls a 'language game', with its own rules. A problem he points out is that "an ostensive definition can be variously interpreted in *every* case" (Wittgenstein, 2009, §28). How would a child know, when someone is trying to teach the definition of the word 'two' and points towards two nuts while saying "That is called 'two'", what it is that one wants to call 'two'? How would it have to know that someone means the *number* and not *this* group of nuts? We would first need to define the use of the word 'number' with other words. "And what about the last definition in this chain?" asks Wittgenstein us (Wittgenstein, 2009, §29). His conclusion is that we will first need to learn the role a word is supposed to play in the language (Wittgenstein, 2009, §30). He compares this to a game of chess: when someone shows the king in chess and says "this is the king", that does not tell someone how to use the kind unless he already knows the rules of the game (Wittgenstein, 2009, §31). Wittgenstein show us that the "meaning of a word is its use in the language" (Wittgenstein, 2009, §43). This means that we can never perfectly define the meaning of something, but we need to observe the complete 'language game' that is being played in a certain context. By observing how this game is played, we can learn the rules of the game. The meaning of language is completely embedded in behavior, gestures, actions. This notion of meaning resembles closely the concepts of distributed qualities and learning rules that we see in machine learning. The chatbot engages in a 'language game' that consists of showing pictures to evoke emotions. The intentions the system is trying to communicate to the receiver are the transfered intentions of the programmer. The system

does not need to know the meaning of the images nor of the emotional reactions to the images: it just observes the 'rules' of the 'languag game' that is being played. The system can observe that the way it uses a certain image, evokes certain responses and this use explains the meaning of an image: this is an image that can be used to evoke a response we consider to be desireable.

This perspective on meaning shows us where the weak points of the system can be found. Just like a child might be confused when we say "that is called two", the chatbot system might be "confused" initially when it observes a situation because it has to aquire a network of rules in order to interpret a single action. While the system will, in principle, be able to learn the correlation between actions and emotional responses that follow and action, it might take a long time before it learns *our* rules. We might imagine a language where small groups of nuts are called "two" and there would be no way to figure this out based on just one example. If the time it takes to learn exceeds the patience of the user such that the user stops to interact with the system the system won't be able to learn anything at all. That's why the extra layer of clustered personalities might be essential to gather enough observations to learn rules that make sense in most cases and to finetune these rules for specialized niches of people instead of learning everything from scratch every time it encounters a new person.

So, even though it is reasonable to expect the system to be able to learn (at least in principle) the correlations between the semantic vector of an image and the emotional response it evokes in different types of personalities, it migth be challenging to finetune the system in such a way that this learning is done within a reasonable amount of time. There are hyperparameters in

1. the DDPG algorithm, e.g what we defined as the function approximators θ^{μ} and θ^{Q} are implemented as neural network architectures themselves, with hyperparameters like the amount of neurons
2. the unsupervised clustering for tag creation, e.g. using k-means or an infinite mixture model, each with their own hyperparameters
3. the evolutionary algorithm for filter creation, e.g. thresholds for the information gain, the size of a population and mechanisms for mutation and selection of the next generation
4. the sampling process that picks the best actor from the current population to interact with a user (e.g. Bayesian learning)

Finetuning these is mainly a trail-and-error driven practice guided by some general rules of thumb and goes beyong the scope of this paper. In addition to finetuning the hyperparameters, it is possible to replace complete aspects of the architecture. For example, the emotion recognition CNN that generates the input for the DDPG could be extended with other observations (e.g. brainwaves measured by a portable e.e.g.-device) or augmented with natural language processing. Also, the output could be extended to include single words, or to combine sequences of more then one image. Obviously, all these extensions would increase the complexity so a first step would be to implement the most basic architecture as illustrated in figure 4. Additional research will have to show whether this type of chatbot will actually generate conversations that are perceived as meaningful. Humans interacting with the system could feel that the chatbot is able to touch them emotionally and is able to figure out patterns that actually work, or could perceive the attempts of the chatbot as random actions that might evoke a feeling, but no more then flashing a random sequence of images would achieve.

A final remark I want to make here, is on the possible contexts this system could be implemented in. In general, this architecture could be usefull in every context where emotions are part of the communication. This could both be as part of a human-human interaction where the system augments the communication of the humans or as a human-computer interaction. An example of the augmentation of human-human interaction is a therapeutic context where it might be helpful for the client to be in a specific mood in order to take the most advantage of the therapy. This system could assist during a therapy session by both helping the therapist in sensing tension (even if the client might unconsciously try to mask this tension) and by intervening with images that help to relax the client. An example of a human-computer interaction could be an application that helps people with insomnia to fall asleep (which is usefull because of the 24/7 availability of the chatbot). Other contexts could range from chatbots that assist elderly people that are suffering from cognitive decline to business contexts where emotions play a role, for example in the handling of customer complaints, decision making or sales.

References

Al-Halah, Z., Aitken, A., Shi, W., & Caballero, J. (2019). Smile, be happy:) emoji embedding for visual sentiment analysis. In *Proceedings of the ieee international conference on computer vision workshops* (pp. 0–0).

Arnfield, S., Roach, P., Setter, J., Greasley, P., & Horton, D. (1995). Emotional stress and speech tempo variation. In *Speech under stress.*

Baker, B., Kanitscheider, I., Markov, T., Wu, Y., Powell, G., McGrew, B., & Mordatch, I. (2019). Emergent tool use from multi-agent autocurricula. *arXiv preprint arXiv:1909.07528.*

Barrett, L. F., Adolphs, R., Marsella, S., Martinez, A. M., & Pollak, S. D. (2019). Emotional expressions reconsidered: Challenges to inferring emotion from human facial movements. *Psychological science in the public interest*, *20*(1), 1–68.

Bezanson, J., Edelman, A., Karpinski, S., & Shah, V. B. (2017). Julia: A fresh approach to numerical computing. *SIAM review*, *59*(1), 65–98. Retrieved from `https://doi.org/10.1137/141000671`

Bishop, C. M. (2006). *Pattern recognition and machine learning*. springer.

Bostrom, N. (2016). *Superintelligence: Paths, dangers, strategies.* Oxford: Oxford University Press.

Boucher, J. D., & Carlson, G. E. (1980). Recognition of facial expression in three cultures. *Journal of Cross-Cultural Psychology*, *11*(3), 263–280.

Bradford, A. (2018). *Facts about rainforests.* Retrieved 2020-05-22, from `https://www.livescience.com/63196-rainforest-facts.html`

Brandtzaeg, P. B., & Følstad, A. (2017). Why people use chatbots. In *International conference on internet science* (pp. 377–392).

Casale, S., Russo, A., Scebba, G., & Serrano, S. (2008). Speech emotion classification using machine learning algorithms. In *2008 ieee international conference on semantic computing* (pp. 158–165).

Chan, S. (2001). Complex adaptive systems. In *Esd. 83 research seminar in engineering systems* (Vol. 31, pp. 1–19).

Correa, E., Jonker, A., Ozo, M., & Stolk, R. (2016). Emotion recognition using deep convolutional neural networks. *Tech. Report IN4015*.

Cuayáhuitl, H., Lee, D., Ryu, S., Cho, Y., Choi, S., Indurthi, S., ... Kim, J. (2019). Ensemble-based deep reinforcement learning for chatbots. *Neurocomputing*, *366*, 118–130.

Cuayáhuitl, H., Lee, D., Ryu, S., Choi, S., Hwang, I., & Kim, J. (2019). Deep reinforcement learning for chatbots using clustered actions and human-likeness rewards. In *2019 international joint conference on neural networks (ijcnn)* (pp. 1–8).

Fitzpatrick, K. K., Darcy, A., & Vierhile, M. (2017). Delivering cognitive behavior therapy to young adults with symptoms of depression and anxiety using a fully automated conversational agent (woebot): a randomized controlled trial. *JMIR mental health*, *4*(2), e19.

Fonagy, I., & Magdics, K. (1960). Speed of utterance in phrases of different lengths. *Language and Speech*, *3*(4), 179–192.

Fung, P., Bertero, D., Xu, P., Park, J. H., Wu, C.-S., & Madotto, A. (2018). Empathetic dialog systems. In *The international conference on language resources and evaluation. european language resources association.*

Ge, H. (2020). *Probabilistic modelling using the infinite mixture model.* Retrieved 2020-05-23, from `https://turing.ml/dev/tutorials/6-infinitemixturemodel/`

Ge, H., Xu, K., & Ghahramani, Z. (2018). Turing: a language for flexible probabilistic inference. In *International conference on artificial intelligence and statistics, AISTATS 2018, 9-11 april 2018, playa blanca, lanzarote, canary islands, spain* (pp. 1682–1690). Retrieved from `http://proceedings.mlr.press/v84/ge18b.html`

Geisslinger, M. (2019). *Autonomous driving: Object detection using neural networks for radar and camera sensor fusion* (Doctoral dissertation). doi: 10.13140/RG.2.2.10949.47840

Guntuku, S. C., Lin, W., Carpenter, J., Ng, W. K., Ungar, L. H., & Preoţiuc-Pietro, D. (2017). Studying personality through the content of posted and liked images on twitter. In *Proceedings of the 2017 acm on web science conference* (pp. 223–227).

Hofstadter, D. R. (2007). *I am a strange loop*. Basic books.

Holland, J. H. (2012). *Signals and boundaries. Building blocks for complex adaptive systems*.

Huang, C., & Zaïane, O. R. (2019). Generating responses expressing emotion in an open-domain dialogue system. In *Lecture notes in computer science (including subseries lecture notes in artificial intelligence and lecture notes in bioinformatics)* (Vol. 11551 LNCS).

Inkster, B., Sarda, S., & Subramanian, V. (2018). An empathy-driven, conversational artificial intelligence agent (wysa) for digital mental well-being: real-world data evaluation mixed-methods study. *JMIR mHealth and uHealth*, *6*(11), e12106.

Jiang, Y., Costello, P., Fang, F., Huang, M., & He, S. (2006). A gender-and sexual orientation-dependent spatial attentional effect of invisible images. *Proceedings of the National Academy of Sciences*, *103*(45), 17048–17052.

Keeble, R. (2006). *Communication ethics today*. Troubador Publishing Ltd.

Kim, H., Kim, Y., Kim, S. J., & Lee, I. (2018). Building emotional machines: Recognizing image emotions through deep neural networks. *IEEE Transactions on Multimedia*, *20*(11), 2980-2992.

Kiseliou, I. (2020). *Better cooperation through communication in multi-agent reinforcement learning.*

Koch, C., & Tsuchiya, N. (2007). Attention and consciousness: two distinct brain processes. *Trends in cognitive sciences*, *11*(1), 16–22.

Kripke, S. (2013). From *On Rules and Private Language*. In *the philosophy of language* (Sixth ed., pp. 546–558). Oxford University Press Inc.

Kurniawan, H., Maslov, A. V., & Pechenizkiy, M. (2013). Stress detection from speech and galvanic skin response signals. In *Proceedings of the 26th ieee international symposium on computer-based medical systems* (pp. 209–214).

Lee, D., Oh, K. J., & Choi, H. J. (2017, mar). The chatbot feels you - A counseling service using emotional response generation. In *2017 ieee international conference on big data and smart computing, bigcomp 2017* (pp. 437–440). Institute of Electrical and Electronics Engineers Inc.

Lester, P. M. (2013). *Visual communication: Images with messages*. Cengage Learning.

Leutner, F., Yearsley, A., Codreanu, S.-C., Borenstein, Y., & Ahmetoglu, G. (2017). From likert scales to images: Validating a novel creativity measure with image based response scales. *Personality and Individual Differences*, *106*, 36–40.

Li, J., Monroe, W., Ritter, A., Galley, M., Gao, J., & Jurafsky, D. (2016). Deep reinforcement learning for dialogue generation. *arXiv preprint arXiv:1606.01541*.

Li, X., Hong, X., Moilanen, A., Huang, X., Pfister, T., Zhao, G., & Pietikäinen, M. (2017). Towards reading hidden emotions: A comparative study of spontaneous micro-expression spotting and recognition methods. *IEEE transactions on affective computing*, *9*(4), 563–577.

Lillicrap, T. P., Hunt, J. J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., ... Wierstra, D. (2015). Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*.

Mack, A., & Clarke, J. (2012). Gist perception requires attention. *Visual Cognition*, *20*(3), 300–327.

Matsumoto, D. (1992). American-japanese cultural differences in the recognition of universal facial expressions. *Journal of cross-cultural psychology*, *23*(1), 72–84.

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., ... others (2015). Human-level control through deep reinforcement learning. *nature*, *518*(7540), 529–533.

Oh, K. J., Lee, D., Ko, B., & Choi, H. J. (2017). A chatbot for psychiatric counseling in mental healthcare service based on emotional dialogue analysis and sentence generation. In *Proceedings - 18th ieee international conference on mobile data management, mdm 2017*.

Pamungkas, E. W. (2019). Emotionally-Aware Chatbots: A Survey.

R Core Team. (2020). R: A language and environment for statistical computing [Computer software manual]. Vienna, Austria. Retrieved from `https://www.R-project.org/`

Rasmussen, C. E. (2000). The infinite gaussian mixture model. In *Advances in neural information processing systems* (pp. 554–560).

Segalin, C., Cheng, D. S., & Cristani, M. (2017). Social profiling through image understanding: Personality inference using convolutional neural networks. *Computer Vision and Image Understanding*, *156*, 34–50.

Serban, I. V., Sankar, C., Germain, M., Zhang, S., Lin, Z., Subramanian, S., ... others (2017). A deep reinforcement learning chatbot. *arXiv preprint arXiv:1709.02349*.

Shannon, C. E. (1948). A mathematical theory of communication. *The Bell system technical journal*, *27*(3), 379–423.

Shawar, B. A., & Atwell, E. (2007). Chatbots: are they really useful? In *Ldv forum* (Vol. 22, pp. 29–49).

Shum, H.-Y., He, X.-d., & Li, D. (2018). From eliza to xiaoice: challenges and opportunities with social chatbots. *Frontiers of Information Technology & Electronic Engineering*, *19*(1), 10–26.

Silver, D., Lever, G., Heess, N., Degris, T., Wierstra, D., & Riedmiller, M. (2014). Deterministic policy gradient algorithms..

Sklar, A. Y., Levy, N., Goldstein, A., Mandel, R., Maril, A., & Hassin, R. R. (2012). Reading and doing arithmetic nonconsciously. *Proceedings of the National Academy of Sciences*, *109*(48), 19614–19619.

Smith, A. J. (2002). Applications of the self-organising map to reinforcement learning. *Neural networks*, *15*(8-9), 1107–1124.

Voytek, B. (2020). *Are there really as many neurons in the human brain as stars in the milky way?* Retrieved 2013-05-20, from `https://www.nature.com/scitable/blog/brain-metrics/are_there_really_as_many/`

Wikipedia contributors. (2020). *"chinese restaurant process — Wikipedia, the free encyclopedia"*. Retrieved 2020-05-23, from `https://en.wikipedia.org/w/index.php?title=Chinese_restaurant_process&oldid=967672492`

Wittgenstein, L. (2009). *Philosophical investigations*. John Wiley & Sons.