VRIJE UNIVERSITEIT BRUSSEL

Proef ingediend met het oog op het behalen van de graad van Bachelor of Science in de computerwetenschappen

HET ONSTAAN VAN EEN SYSTEEM VAN KLANKEN

En blik op hoe agenten met behulp van luisteren en horen een systeem van klanken kunnen ontwikkelen

FLYNN STEPPE
2021-2022

Promotor: Prof. Dr. Bart de Boer
Begeleider: Peter Dekker
Wetenschappen & Bio-ingenieurswetenschappen

VRIJE UNIVERSITEIT BRUSSEL

Thesis submitted in partial fulfilment of the requirements for the degree of Bachelor of Science in Computer Sciences

THE DEVELOPMENT OF A SYSTEM OF SPEECH SOUNDS

A look at how agents with the ability to hear and listen can develop a system of speech sounds

FLYNN STEPPE
2021-2022

Promotor: Prof. Dr. Bart de Boer
Advisor: Peter Dekker
Science & Bio-Engineering Sciences

VRIJE UNIVERSITEIT BRUSSEL

Table of Contents

VRIJE UNIVERSITEIT BRUSSEL

Samenvatting

Talen kunnen gerepresenteerd worden met behulp van verschillende klanken. Deze geluiden hebben allemaal een vaste frequentie (toonhoogte) die het mogelijk maakt voor ons om ze te analyseren. Een taal zoals ze de dag van vandaag gekend is, is echter niet altijd de dominante manier van communiceren geweest. Hoe is deze dan ontstaan? Een situatie kan gesimuleerd worden waarin nog geen vaste taal ontwikkeld is door gebruik te maken van 'intelligent agents' die primitieve tonen kunnen uitwisselen met hun buren en leren van wat ze horen. In deze thesis is het doel om ze een kleine populatie van agenten aan te maken en aan te tonen of het mogelijk is om een taal te vormen, gelijkaardig aan de onze, zonder interventie of bepaalde aannames die dit proces zouden kunnen versnellen. De implementatie gebruikt technieken in het veld van artificiële intelligentie. We tonen hoe onze resultaten verschillen met die van Oudeyer (2005) ter invloede van specifieke implementatie keuzes en hoe onze resultaten misschien die van Oudeyer kunnen benaderen.

Summary

Languages can be represented with the help of different speech sounds. These sounds all have a certain frequency (pitch) that makes it possible for us to analyse them. A language like the ones in the current day and age has not always been present though. How did these come to exist then? A situation can be simulated in which no concrete language has formed yet by making use of 'intelligent agents', which can exchange primitive tones with their neighbours and learn from what they hear. In this thesis the goal is to create a small population of agents and show if it is possible for them to form a language like ours without any intervention or pre-existing notions that might accelerate this process. The implementation uses techniques in the field of artificial intelligence. We show how our results differ from those of Oudeyer (2005) because of specific implementation choices end how me may obtain results similar to those of Oudeyer.

VRIJE UNIVERSITEIT BRUSSEL

Acknowledgements

First, I would like to thank my advisor Peter Dekker for his continuous support throughout the academic year of 2021-2022 by always answering any questions I had and offering constructive advice whenever needed. Similarly, I would like to thank my promotor, Prof. Dr. Bart de Boer for providing his input on some of the key problems encountered during this thesis. Lastly, the support of my friends and family was invaluable in making sure I finished this thesis successfully, thank you.

1. Introduction

A form of communication that is understood by the people around you, is something that is not often questioned. It might prove to be interesting to have a clearer idea of how the creation of languages like ours was accomplished and for this reason, this topic was chosen. However, this topic requires knowledge in more than just the field of computer sciences and as such it is difficult to try and get a complete understanding of the origin of languages. Instead, the focus will be on the origins of speech sounds, which are an essential component of languages, and the influences of certain properties. To summarize: in this thesis, an attempt will be made to try to provide a clearer image of how the languages of today came to be by creating a system of agents where each of these agents can produce and perceive sounds with their neighbours. To this end the following research question has been formulated:

What could cause a system of speech sounds to be developed in a population of communicating agents when no prior assumptions are made about the factors that influence their interactions?

This thesis will also serve as a re-implementation of "The self-organization of speech sounds" (Oudeyer, The self-organization of speech sounds, 2005). The intentions of Oudeyer's work align with the goals of this thesis and so his paper will be used extensively as a reference and will be discussed in more detail in a later section along with all the differences between the two. The modern programming language Python will be used in combination with recent frameworks like Mesa and Minisom. The programming component for this thesis could also be used as a starting point for similar research in this field.

We will start by explaining and going more in detail for a couple of relevant concepts and reoccurring terms in this thesis like agent-based models, self-organizing maps and speech sounds. Following this introductory section, we will talk about Oudeyer's paper in-depth while comparing his approach to ours and how this will be translated into Python code. Our implementation naturally follows in which we will explain certain design choices and how it will help us in formulating a response to the research question. Lastly, it is important for us to look more closely at all the results generated by our experiment and draw conclusions based on these observations. Any shortcomings or certain aspects of this thesis that did not go as planned will be discussed as well along with how we might have avoided them.

1.1 Agent-based Models

For simulating the exchange of speech sounds, we use an agent-based model (ABM). This computational model is often used for the simulation of complex interactions between intelligent agents (Axelrod, 1997). Since we make no assumptions about the conversations between agents in this paper, using a setup like the one described here will be beneficial in discovering and understanding what factors can cause a specific outcome, more specifically the development of a system of speech sounds, as we could easily tune the parameters of the agents. These intelligent agents make decisions based on a set of rules in each step without any interference of the user. A logical assumption about the rules of agents relating to languages could be that every time an agent hears a sound, they learn from it, or that not every agent produces a sound at the same time. It is important to note that a model of agents is a simplification of its real counterpart as it is impossible to make a one-to-one simulation. Properties like the number of agents, their

behaviour in certain situation, their underlying structure, their way of interacting, and many more are all examples of this limitation. It would take an immense amount of computational power otherwise (Manzo, 2014).

1.2 Self-organizing Maps

To represent the production and perception of speech sounds as accurately as possible, we will be using self-organizing maps (SOM's) (Kohonen, 2013). These SOM's have a long tradition of being used for language recognition and as cognitive models (Li & Zhao, 2013). A self-organizing map is an unsupervised machine learning technique that can be used to produce a low-dimensional representation of higher dimensional data. Often the goal is to represent data of p dimensions as a map space of two dimensions. Since unsupervised learning techniques work for unlabelled data like ours and SOM's also makes no changes to the topological structure of this data, it is ideal for clustering data. An SOM is a type of artificial neural network, like the neural maps used by Oudeyer, but instead of the usual error-correction learning, it uses competitive learning.

Initially, the weights of the neurons in our SOM will be set to a small random value. An alternative for this could be to set the initial weights of neurons to values like the training samples. This will make it easier for our SOM to approximate our training data but, as we mentioned in the research question, we do not want to make any assumptions about the conversations between agents. We will now go more in-depth about the idea of competitive learning.

Whenever a training sample is fed to the SOM, its Euclidean distance to all the weight vectors of our neurons is calculated. One of these weight vectors will be chosen as our best matching unit (BMU) because it is the most like the training sample. The neurons closest to the BMU in our network will be adjusted as well, but to a smaller degree. This update gets smaller over time and the further a neuron is away from our BMU in the network, the smaller the update will be too.

This means each neuron in our self-organizing map will compete to be the one allowed to respond to certain input data. Intuitively, you could apply this to the perception of speech sounds. The sound produced by one agent could act as our input data to which the winning neuron and to a lesser extent, the neurons around it, respond by updating their values. Each of these neurons has a weight vector associated with them. These weights will slowly move towards the input sound. Lastly, these maps can be used to classify additional observations based on what we learned during the training (Bonabeau, 2002), but this won't be of much use to our research question.

1.3 Speech Sounds

Like mentioned previously, the field of languages is complex and as such we won't go into too much detail about everything it governs. Instead, we will focus on some important concepts that may help in understanding this thesis.

The first concept we will talk about is the articulatory configuration of our speech sounds. What is the best way to convert these into data that we can work with? We will make use of the three major vowel parameters, position (p), height (h) and rounding (r) (Ladefoged & Maddieson, 1996). These will be real numbers between the range of zero and one, we get a representation of (p, h, r). A value close to zero for position means most front, while a value close to one means most back when we are looking at the tongue in the front-

back dimension. Height corresponds to the highest point of the tongue. If this point is close to one (meaning the tip of your tongue is close to touching your palate), you will produce a closed vowel as your tongue is blocking the air and sound from leaving your mouth. Meanwhile, a value close to zero means you are producing an open vowel. For rounding, we are referring to the rounding of the lips. Zero will mean most spread and one means most rounded. (de Boer, 1999) A good comparison can be made between the vowels [a] (0, 0, 0) and [u] (1, 1, 1), this will be left to the reader to try out. As such, our data will be three-dimensional, which hints at the use of self-organizing maps for dimensionality reduction.

Now that we know how to represent our articulatory model, we need to consider what it will produce. Since we are trying to answer the question of how a system of speech sounds can develop, it is important that the sounds the agents produce are realistic so we may eventually compare them with real languages. Therefore, the acoustic signals that will be exchanged are the first four formant frequencies[1] (F1, F2, F3, F4) and will be calculated directly from the articulatory representation.

The four formants would be perceived as four-dimensional data and this does not work for our self-organizing maps since we are trying to reduce the dimension of the earlier mentioned articulatory representation, and not increase it. Neither would this be very realistic. Instead, we will use a distance function between the acoustic signals we want to perceive and their acoustic prototype, as humans usually perceive speech in terms of prototypical sounds as observed by Comrie (Comrie, 1981). This will take the form of the first formant frequency, followed by the effective second formant frequency (F1, F'2). The latter of these is measured in Barks[2] (de Boer, 1999). These representations of vowels are two-dimensional data, which is exactly want we want for our self-organizing maps.

[1] Formants are frequency peaks in the spectrum which have a high degree of energy and are especially prominent in vowels. Each formant corresponds to a resonance in the vocal tract. (Priyanka, Bharti, & Suresh, 2016)

[2] The Bark scale is an (approximately) logarithmic frequency scale that models the human perception of pitch. The distance in pitch between two perceived sounds are also of equal distance in Barks, no matter their absolute frequency. (de Boer, Self-organisation in Vowel Systems, 1999)

2. Methods

This section will go more into the specifics of what we are trying to accomplish and how we will approach our research question. We briefly introduce two relevant frameworks first so we may refer to these throughout this section. This is followed by an explanation of the relevant aspects of Oudeyer's paper of 2005 for answering our research question.

2.1 Frameworks

2.1.1 Mesa

One way of building an artificial agent-based system in Python is the Mesa framework. This framework offers a wide range of options for building our model including multiple types of spaces, schedulers, data collectors, visualization, …. (Mesa).

2.1.2 MiniSom

For adding self-organizing maps to our Python implementation, we use MiniSom. MiniSom is a minimalistic and Numpy based implementation of self-organizing Maps and was designed to be easily extendable by researchers and easily understandable by students. The MiniSom implementation will have to be slightly edited for it to fit our exact needs, which will be talked about in more detail in a later section (MiniSom).

2.2 Oudeyer 2005

2.2.1 Abstract

The system Oudeyer describes in his paper is a generalization of the one he presented in his paper of 2001 (Oudeyer, 2001). The idea is as follows: We will build an artificial system that is populated by agents, each of these agents will have a way of speaking and listening to different agents. This will be accomplished by two different artificial neural networks, a perception map (for listening) and a motor map (for speaking). When taking this approach, we can easily vary the complexity and degree of realism by changing the parameters our system is working with. The biggest difference between our implementation and the idea Oudeyer describes, is that we will use self-organizing maps instead of the neural maps he makes use of.

Implementing this system in Python is relatively simple. We use Mesa to create an agent-based model with 20 different agents so we can compare our results to that of Oudeyer. He has shown that the number of agents doesn't influence the outcome directly but rather determines how fast the system will converge. These agents will need to move around, luckily Mesa provides multiple ways of adding space to the model. The simplest of these is a grid space but we will opt for a continuous space instead as this is a more realistic approach to simulating living beings. We will set the dimensions of the space to a width of 100 and a length of 100. To simulate movement, we make use of two-dimensional random walk (Hughes, 1996). In every step of the model, the agents will move a distance of 1 in one of four directions: up, down, left, or right and their starting position in the continuous space is entirely random. We decided to allow agents to appear on the other

side of the space if their coordinates become negative as we think such a radical change of position could happen in a realistic setting too. The agents will each have a perception and motor map as previously mentioned. We can make two subclasses of MiniSom for readability. Every step of the model will also be accompanied by a random number of agents that will produce a sound in a random order and this sound will be heard by their closest neighbour and no others. 20 agents are added to the model.

2.2.2 The Three Spaces

Oudeyer continues to introduce three different spaces: the motor space, the perceptual space, and the acoustic space. We have already introduced the representation for these three spaces before, but their use will be made clearer now. The motor space will be three dimensional (p, h, r), the acoustic space will be four dimensional (F1, F2, F3, F4) and lastly the perceptual space will be two dimensional (F1, F'2).

2.2.3 The Path of a Speech Sound

These three spaces illustrate the "path" a speech sound will have to go through. An agent's motor map will produce a sound in the form of an articulatory configuration (motor space), this sound will then be transformed into its acoustic representation (acoustic space) using the formulas given below as derived by (de Boer, 2001) from a database of real vowels presented in (Vallee, 1994):

$$\begin{aligned} F_1 = {} & ((-392+392r)h^2+(596-668r)h \\ & +(-146+166r))p^2+((348-348r)h^2 \\ & +(-494+606r)h+(141-175r))p \\ & +((340-72r)h^2+(-796+108r)h \\ & +(708-38r)), \end{aligned}$$

$$\begin{aligned} F_2 = {} & ((-1200+1208r)h^2+(1320-1328r)h \\ & +(118-158r))p^2+((1864-1488r)h^2 \\ & +(-2644+1510r)h+(-561+221r))p \\ & +((-670+490r)h^2+(1355-697r)h \\ & +(1517-117r)), \end{aligned}$$

$$\begin{aligned} F_3 = {} & ((604-604r)h^2+(1038-1178r)h \\ & +(246+566r))p^2+((-1150+1262r)h^2 \\ & +(-1443+1313r)h+(-317-483r))p \\ & +((1130-836r)h^2+(-315+44r)h \\ & +(2427-127r)), \end{aligned}$$

$$\begin{aligned} F_4 = {} & ((-1120+16r)h^2+(1696-180r)h \\ & +(500+522r))p^2+((-140+240r)h^2 \\ & +(-578+214r)h+(-692-419r))p \\ & +((1480-602r)h^2+(-1220+289r)h \\ & +(3678-178r)). \end{aligned}$$

Finally, the agent that is being spoken to, will listen to this vocalisation and change it to a perceptual representation before feeding it to its perception map (perceptual space). This switch from a four-dimensional space to a two-dimensional space also comes with calculations which can be seen in the formulas given below as stated in (de Boer, 2001) and (Boe, Schwartz, & Vallee, 1995) on observations by (Carlson, Granstrom, & Fant, 1970):

$$F_2' = \begin{cases} F_2 & \text{if } F_3 - F_2 > c, \\ \dfrac{(2-w_1)F_2 + w_1 F_3}{2} & \text{if } F_3 - F_2 \leqslant c \text{ and } F_4 - F_2 \geqslant c, \\ \dfrac{w_2 F_2 + (2-w_2)F_3}{2} - 1 & \text{if } F_4 - F_2 \leqslant c \text{ and } F_3 - F_2 \leqslant F_4 - F_3, \\ \dfrac{(2+w_2)F_3 - w_2 F_4}{2} - 1 & \text{if } F_4 - F_2 \leqslant c \text{ and } F_3 - F_2 \geqslant F_4 - F_3, \end{cases}$$

with

$$w_1 = \frac{c - (F_3 - F_2)}{c},$$

$$w_2 = \frac{(F_4 - F_3) - (F_3 - F_2)}{F_4 - F_2},$$

C in the above formulas is a constant value of 3.5 barks. The only step left before the cycle starts again is for the motor map to update its values, incorporating what it just learned through its perception map.

2.2.4 The Motor Map

Let us now talk a bit more about the element in our agents that starts the exchange of sounds, the motor map. A major difference between our implementation and the one described by Oudeyer is what happens when an agent is randomly chosen to produce a sound. He mentions how two, three or four neurons will be activated in sequence to create an articulatory trajectory. However, our implementation will only feature one neuron being activated, an articulatory point. This point is a neuron j in the motor map and is characterised by a preferred vector v_j. It is this vector that will be updated in all neurons whenever the motor map is activated. An update can occur in two different situations: the agent produces a vocalisation or hears a vocalisation produced by another agent. The later of these will only update the perception map initially but will affect the motor neurons when the agent decides to speak again. The updating process happens in two steps: we will have to compute the neuron that is the most activated in our motor map and take its preferred vector. The second step is modifying the preferred vectors of all neurons. Oudeyer describes a formula for this exact purpose, but we will use the update functionality provided to us by our Python implementation of self-organizing maps, MiniSom. This update operates in a similar way, and we invite the reader to find out more about its inner workings in the well-documented code of MiniSom.

2.2.5 The Perception Map

Unlike the motor map, a neuron i in the perception map will only be updated whenever the agent hears a vocalisation. This stimulus will prompt the perception map to compute the activation of all its neurons to this input. Once again, we will rely on MiniSom to take care of this process for us as the standard tuning function for computing the activation in both Oudeyer's paper and MiniSom is gaussian.

2.2.6 Coupling the Two Maps

One question you might have, is how the motor map will "learn" as currently these two maps have no relation to each other whatsoever, but the updating of the motor map is

partially reliant on the perception map. Oudeyer solved this problem by coupling the two maps so that hearing a vocalisation will affect the production of a new vocalisation. Intuitively, it might be better to see this coupling of the perception and motor map as a "brain" since the motor map and perception map are "learning". If this coupling can be considered the brain, then the motor map can be considered the "mouth" and the perception map the "ears" of the system. Each neuron i in the perception map is connected unidirectionally to all the neurons j in the motor map and this connection between neurons is characterised by a weight w_i,j. These weights are set to a small random value at the start and will evolve over time, a hebbian learning rule (Sejnowsky, 1977) will be used:

$$\delta w_{i,j} = c_2(G_i - \langle G_i \rangle)(G_j - \langle G_j \rangle)$$

In this formula, G_i refers to the activation of neuron i of the perception map and <G_i> to the mean activation of neuron i over a certain time interval. Intuitively, G_j refers to the activation of neuron j of the motor map and <G_j> to the mean activation of neuron j over a certain time interval. The final element in this formula c_2 is a small constant. It is important to note that this learning rule will only come into effect when an agent hears a vocalization produced by itself. This process is known as vocal babbling[3] and shows how agents will learn the perception/motor mapping It has already been mentioned how MiniSom will provide the implementation of G_i for us. The formula for G_j is as follows:

$$\frac{1}{1 + e^{-\sum_i w_{i,j} G_{i,t}(\mathbf{s})}} \cong G_{j,t}(\mathbf{s})$$

This is not the activation formula given in Oudeyer's paper but rather by de Boer as the formula used by him has been shown to be more accurate (de Boer, Evolution of speech (course at VUB), 2021-2022). MiniSom's functionality for computing activation would not work in this case since it does not account for coupled maps, so we will have to implement this formula ourselves.

Before we go more in-depth into the implementation of this coupling of maps, it would be beneficial to consider how many neurons our maps will require. Oudeyer uses 500 neurons in both his motor and perception map. However, he also states that there were no changes in his observations once the number of neurons was larger than ±150. In our initial setup we will try to approach the original 500 neurons by setting the dimensions of our maps to 22 x 22, or a total of 484 neurons.

How are we going to implement this coupling of maps? We will start by creating the array that contains our weights w_i,j that represent the unidirectional connection between the neurons i of the perception map and the neurons j of the motor map. The easiest way to implement this would be using a four-dimensional array. Updating each element of a four-dimensional array every time an agent produces a sound would be extremely costly though, so we decided to pick a two-dimensional array instead that can still fit all the values. Oudeyer does not specify the initial values of the weights in this array, just that it should be a small random value. We chose values between 1x10^(-5) and 1x10^(-6). Any value larger than this range would cause the negative power in the formula for G_j

[3] Vocal babbling is the process of language acquisition in the early stages of an infant in which they experiment by making speech sounds that aren't yet fully recognizable words.

to default to zero, resulting in the array of weights being filled with ones. This allows us to compute the activation of a neuron j with the formula of G_j.

Updating the weight w_i,j is a bit more challenging. Obtaining the result of G_i and G_j poses no problem but calculating the mean activation of neurons i and j over a certain time interval is a bit abstract. We had the idea of keeping track of several past activations and computing their mean value but the number of elements to keep track of is not stated so another approach was necessary. An exponential moving average proved to be the solution. Initially, the first activation of a neuron is recorded. In the following steps, the weighting for each older activation decreases exponentially by multiplying them with a certain α. We set this α to 0.125. The value of c_2 in our Hebbian learning rule is not specified either but a value of 1x10^(-5) was picked as this will scale the result of our hebbian learning rule to the same range as the initial values of the weights w_i,j, which we are trying to update. It is worth noting how the updating of the weights w_i,j involves many different components, which are almost all two-dimensional arrays. For our implementation to perform as efficiently as possible, we will perform matrix operations on these arrays using Numpy instead of updating or computing each value one at a time. This increased the performance significantly.

3. Results

Before looking at any result, it is of paramount importance that the created model is tested extensively to make sure that it does not contain any coding errors and to decide on the ideal parameters. When starting the first of these tests using parameters as close to that of Oudeyer's as possible, it immediately became clear that parameters close to Oudeyer's were not viable for testing because of the total estimated runtime. Even though care was taken to improve the performance as much as possible, a test of 2000 steps would still take close to three days to fully complete. Because of this unexpectedly long runtime, it was decided to perform our tests on smaller scale initially. Instead of Oudeyer's 2000 steps of the model, 1000 were chosen. However, the biggest time sink was the updating of the weights of the connections between the perception and motor map. Since the choice to approach Oudeyer's 500 neurons was made, two maps of size 22 x 22 were chosen for a total of 484 neurons. This means the number of values that would have to be updated in the connections between the maps equals 22 x 22 x 22 x 22, or 234,256 values. Luckily, Oudeyer mentions that no significant changes were spotted in his results once the number of neurons broke ±150, so while testing the model, the size of the maps could be reduced to 15 x 15, for a total of 225 neurons in each map, and only a fraction of the original number of weights that would have to be updated.

3.1 Preliminary Results

The initial tests were performed to confirm the correctness of the model. For this purpose, the distance map of the neurons in the motor map was plotted. Distance maps are useful for visualizing the neurons of self-organizing maps since they can give a clear representation of how the neurons evolve after a number of steps and show any signs of clustering. Darker sections of this plot mean that the distance between neurons is quite significant while lighter areas show us neurons that are closely related. The distance maps of two random agents are shown here:

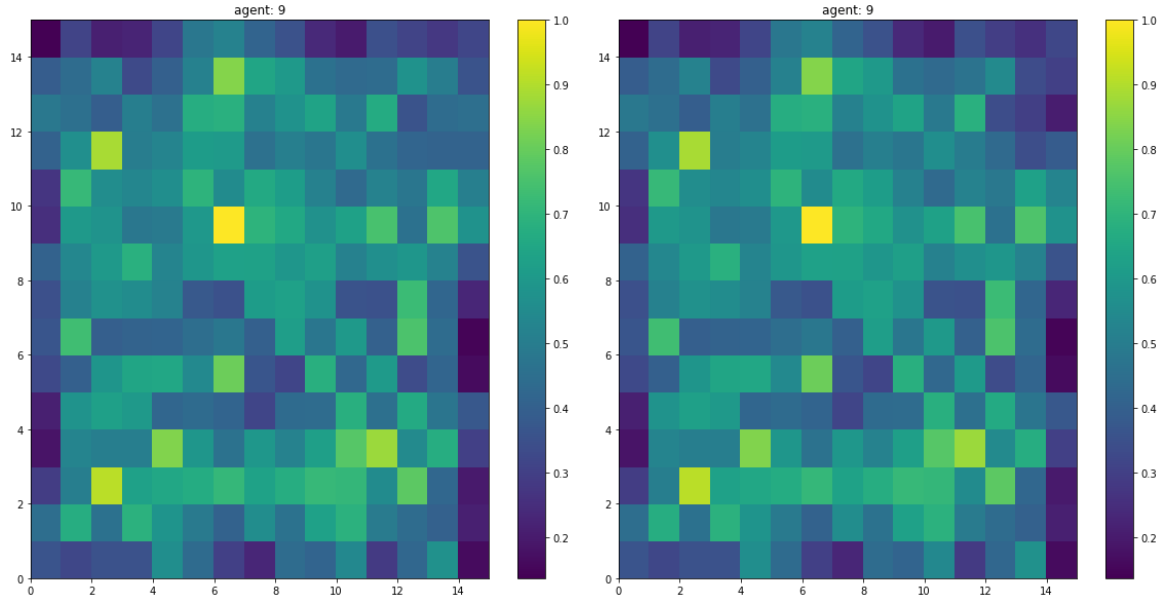


Figure 1: Distance map of the neurons of agent 9 in the motor map after 0 steps (left) and 1000 steps (right)

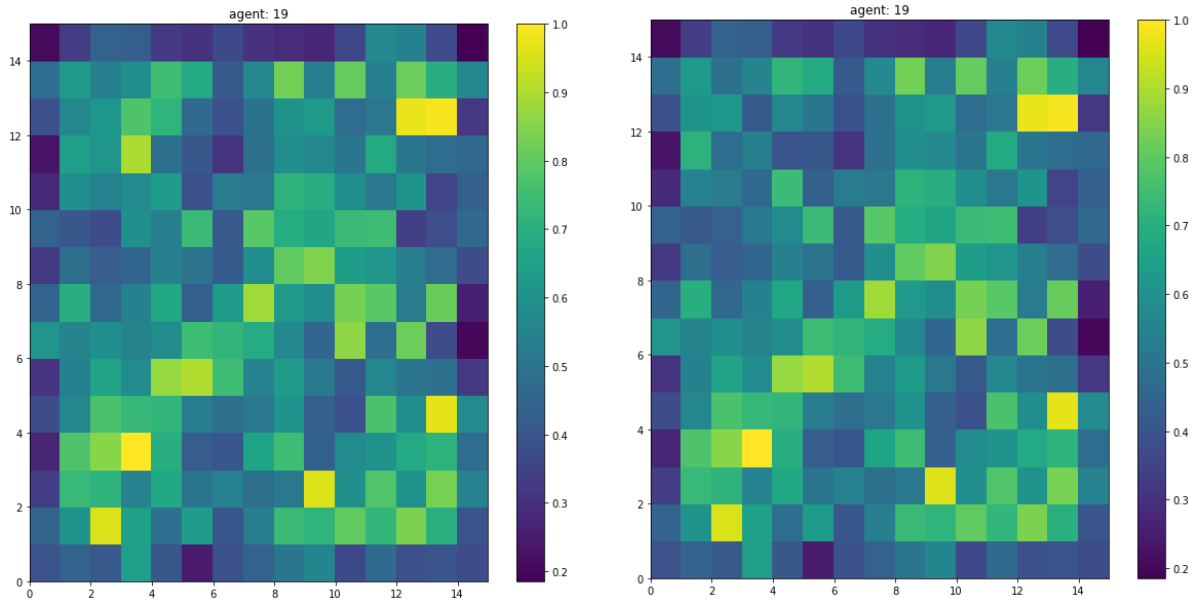


Figure 2: Distance map of the neurons of agent 19 in the motor map after 0 steps (left) and 1000 steps (right)

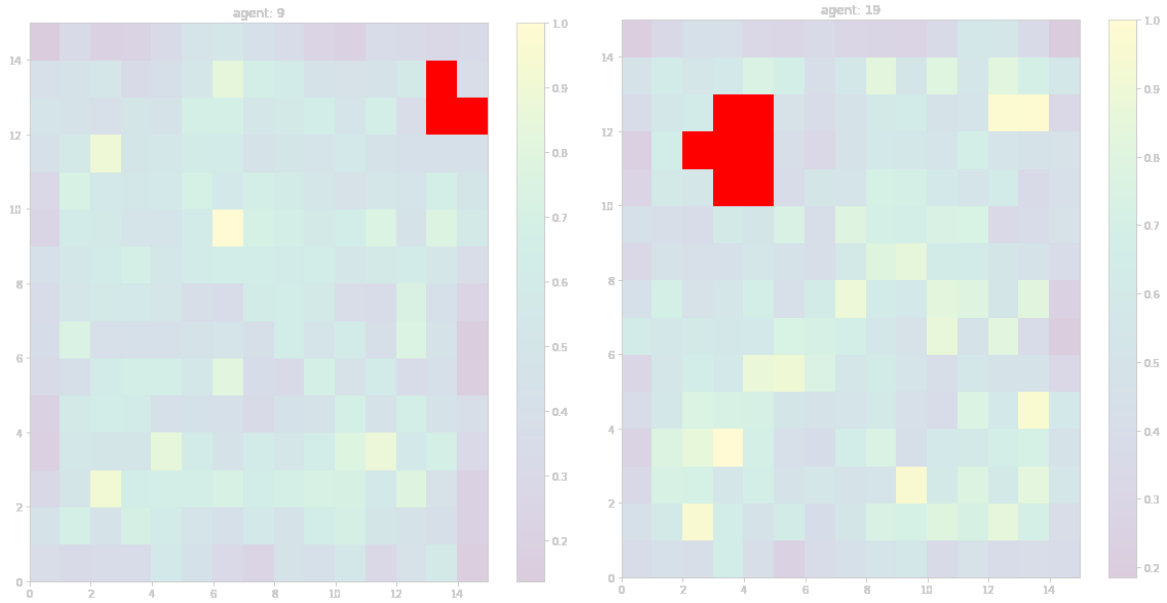


Figure 3: Evolution of the motor map of agent 9 (left) and agent 19 (right)

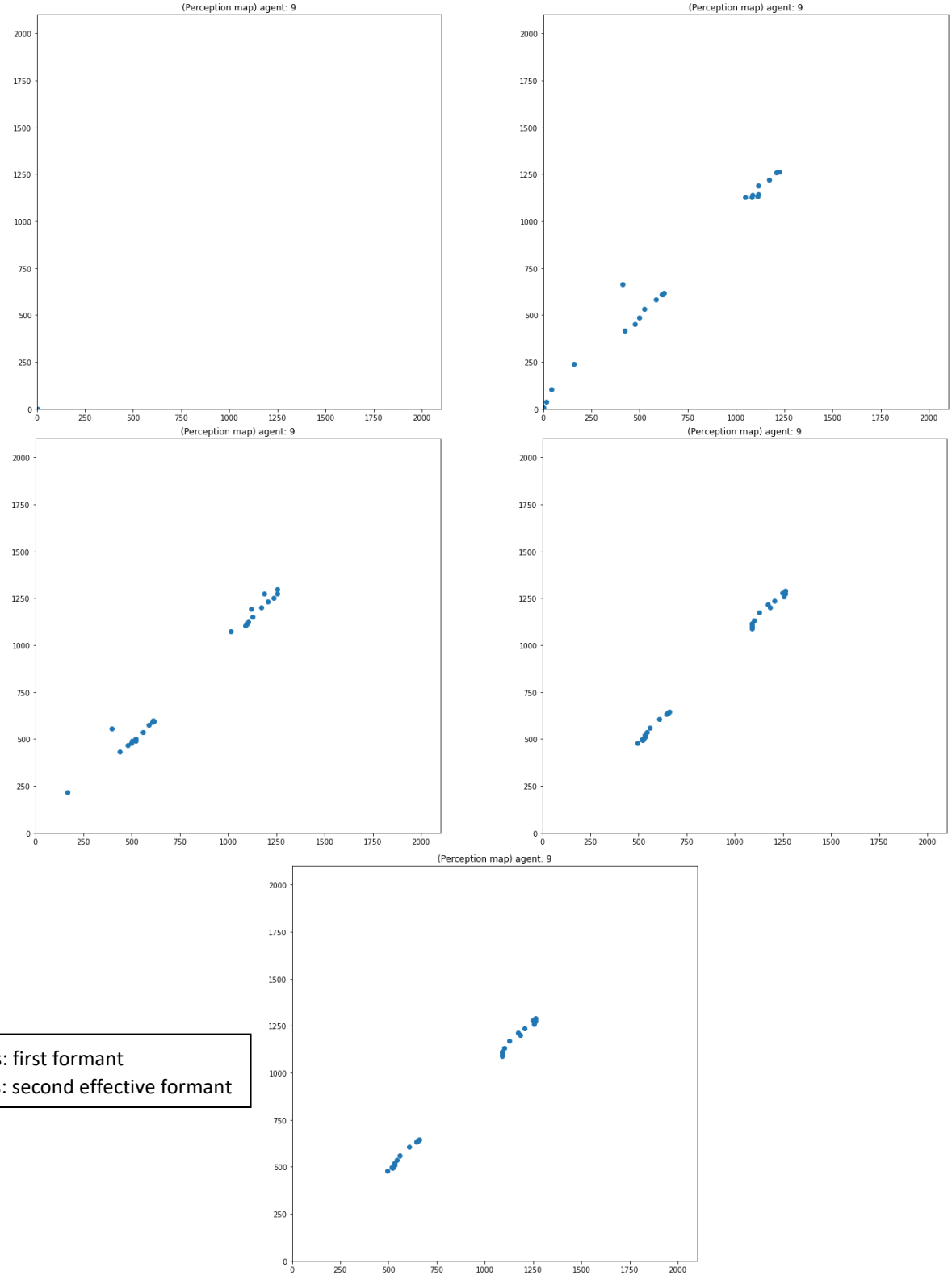
By taking a short look at the above distance maps, no changes are immediately visible. They can be compared using a tool that highlights changes between two images (Online image compare tool). Only changes notable enough will be marked. The changes seen using this tool are minimal, but all happen to be in the same section of the neurons, leading to the believe that some form of clustering between neurons was happening and the model was working as intended.

3.2 Faulty Results

In this section, the model was run for a total of 2000 steps, with a perception and motor map of dimensions 22 x 22, to follow Oudeyer's model as closely as possible. In addition to distance maps for the motor map, distance maps for the agents' perception map and scatter plots containing the values of the weights of the neurons of both maps were added. The results for this run of the model are not shown since they were determined to be faulty. A mistake was made during the preliminary tests since only the neurons of the motor map were looked at, and not those of the perception map. These neurons showed no consistent results and were even on completely different scales between individual agents. Because of these faulty results, some mistakes in the code were found and corrected, allowing us to test the model again.

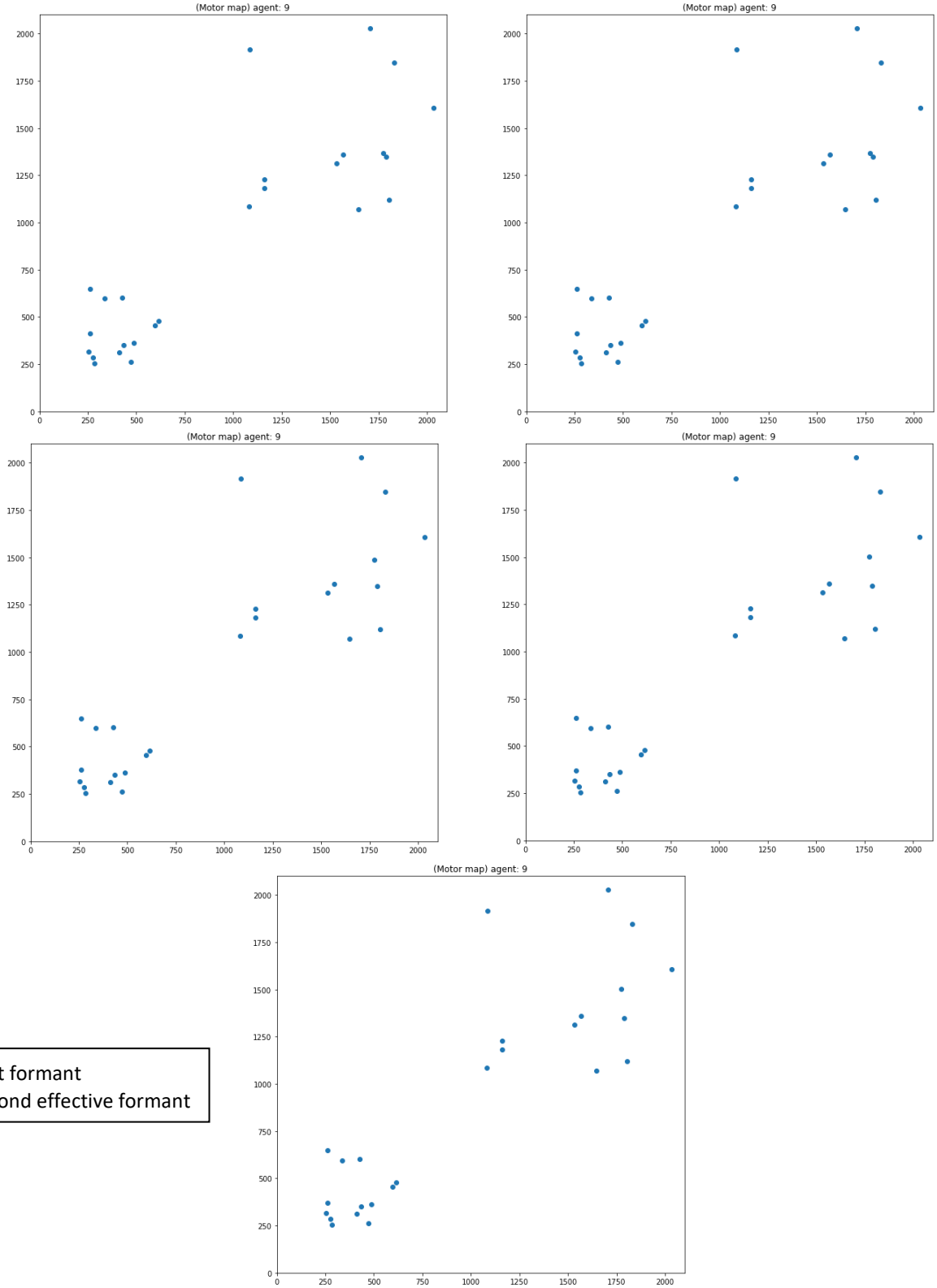
3.3 Final Results

Because a lot of time was spent on testing the model and correcting mistakes in the Python implementation, the final results were obtained by running the model for 2000 steps, but with self-organizing maps of dimensions 13 x 13. This was the lowest number of neurons that would still be above the minimum of ±150 as stated by Oudeyer. The results are shown below in the form of scatter plots:



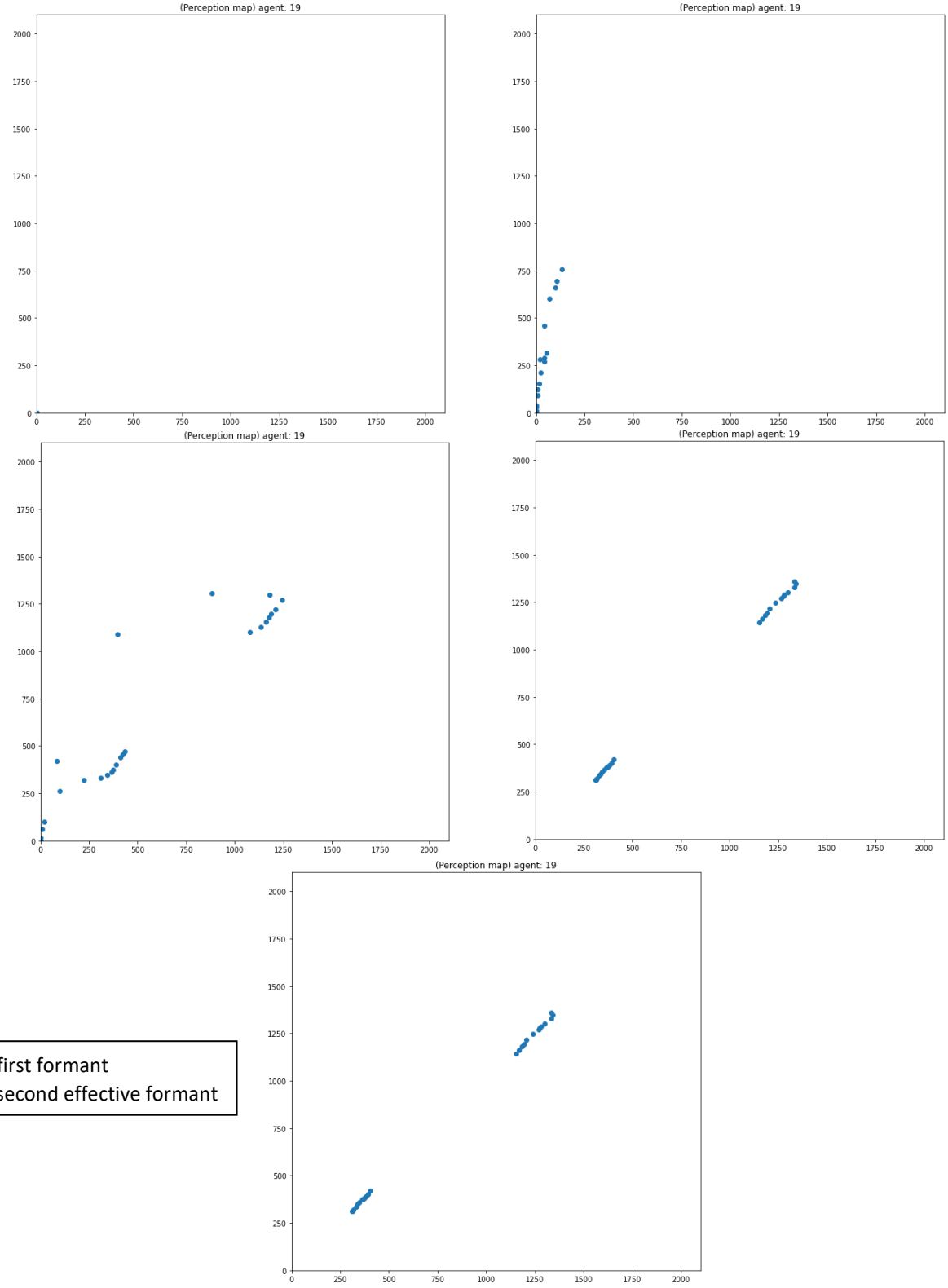
X-axis: first formant
Y-axis: second effective formant

Figure 4: Evolution of the perception map of agent 9 after 0 steps (top left), 500 steps (top right), 1000 steps (middle left), 1500 steps (middle right), and 2000 steps (bottom)



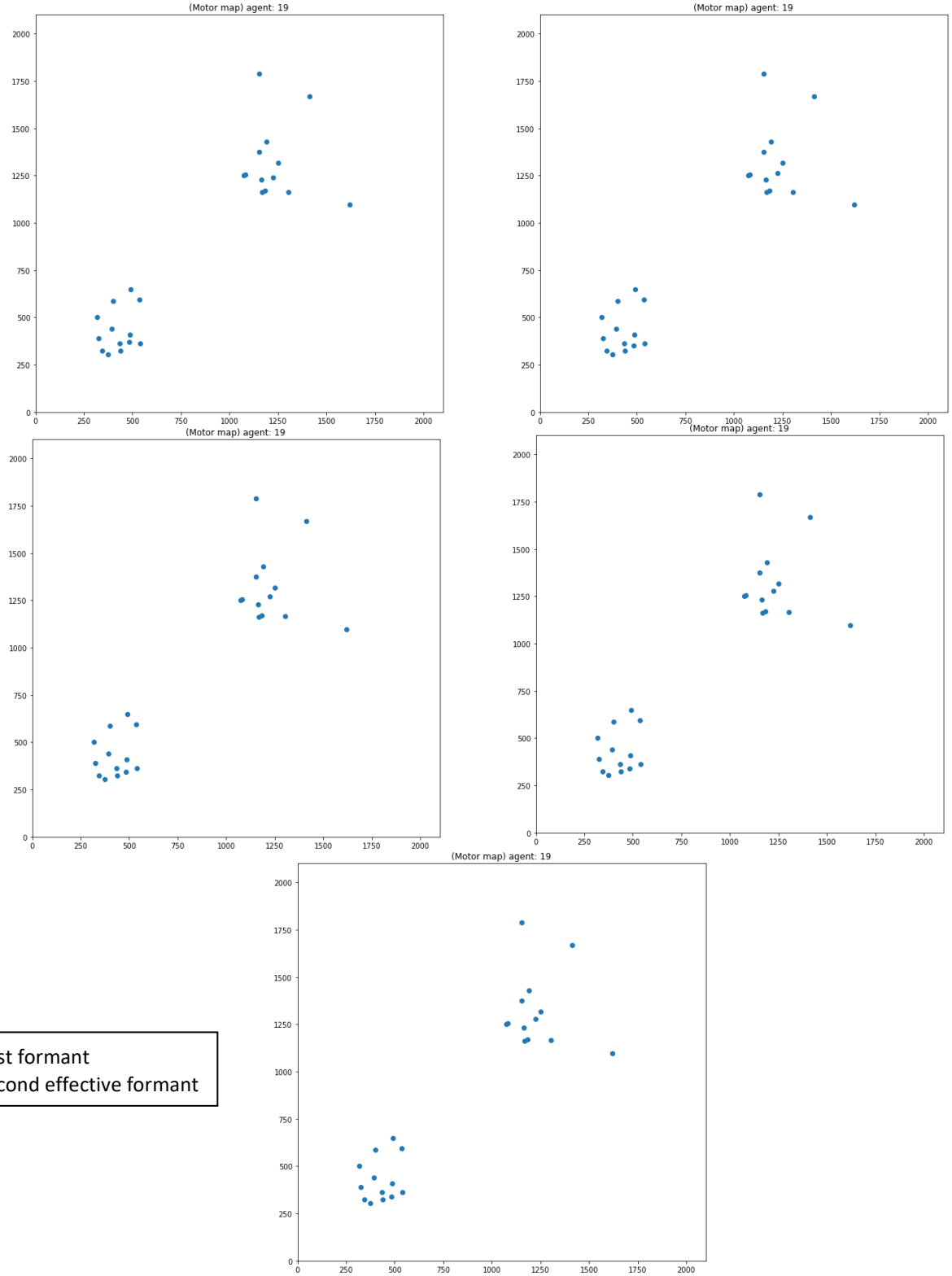
X-axis: first formant
Y-axis: second effective formant

Figure 5: Evolution of the motor map of agent 9 after 0 steps (top left), 500 steps (top right), 1000 steps (middle left), 1500 steps (middle right), and 2000 steps (bottom)



X-axis: first formant
Y-axis: second effective formant

Figure 6: Evolution of the perception map of agent 19 after 0 steps (top left), 500 steps (top right), 1000 steps (middle left), 1500 steps (middle right), and 2000 steps (bottom)



X-axis: first formant
Y-axis: second effective formant

Figure 7: Evolution of the motor map of agent 9 after 0 steps (top left), 500 steps (top right), 1000 steps (middle left), 1500 steps (middle right), and 2000 steps (bottom)

Looking at the figures above, there are some interesting results. Oudeyer only shows the scatter plots of the perception map of a random agent as he states that the motor maps of this same agent and the maps of every other agent in the system is extremely similar. The results shown above do not agree with this statement. There is a clear evolution visible in the perception map in figures 4 and 6 but the motor maps of these two agents in figures 5 and 7 seem to remain unchanged. Changes did occur, but only very small ones as can be seen when a comparison is made between distance maps:

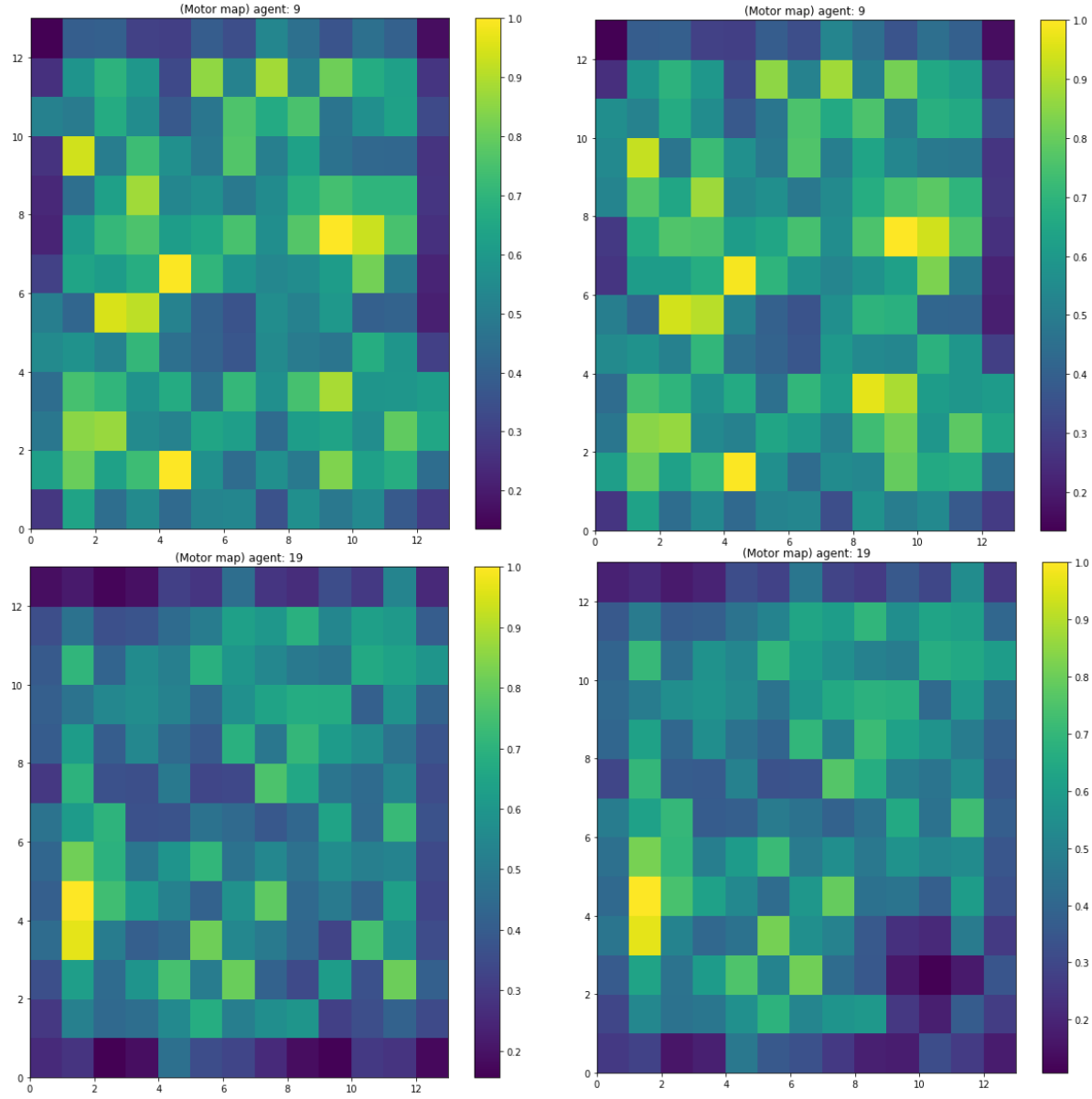


Figure 8: The distance maps of the neurons of agent 9 and 19 in the motor map after 0 steps (left) and 2000 steps (right)

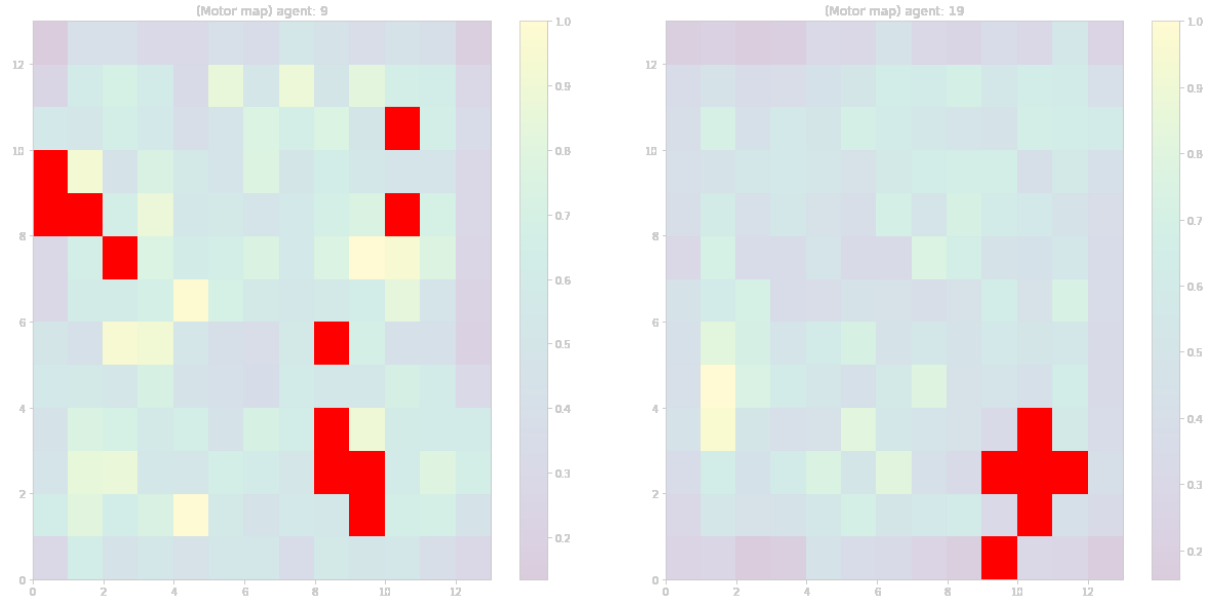


Figure 9: Evolution of the motor map of agent 9 (left) and agent 19 (right)

An observation that can be made is how the perception map of an agent seems to have "learned" the speech code of their motor map by slowly changing its neurons to form a linear line through the neurons of the motor map. This can be caused by the vocal babbling mentioned earlier. An agent updates its perception map based on the speech sound it produces. This process comes to a halt after 1500 steps.

What could have caused this difference in results between Oudeyer's model presented in (Oudeyer, 2005)? It seems that agents know how to learn from themselves, but only to a small extent from others. A possible cause could be that agents don't interact enough with other agents. This lack of conversation can be attributed to the amount of motor neurons that are activated in each step of the model. Oudeyer forms an articulatory trajectory by activating two, three or four neurons in sequence. Meanwhile, the re-implementation only features one of these neurons being activated, an articulatory point. This could mean two, three or even four times the number of steps is needed for our model to experience significant changes in the motor map of its agents.

However, as noted before, the updating of the perception map already seems to come to a halt after 1500 steps and as the updating of the motor map is based on the perception map, these updates to the motor map seem to be too slow. In the paper written by Oudeyer, a couple of implementation choices are left open to the reader. Examples of these are small constants, initial weights but also the mean activation over a certain time interval. Any of these could be the perpetrator behind these results. It is never mentioned in his paper what type of space he uses either, or how large this space is, but it is mentioned that increasing the number of agents or the amount of interaction in every step only influences how fast the system will converge so this is most likely not the cause.

The last possible cause could be the cognitive model used. The re-implementation uses self-organizing maps instead of the neural maps used by Oudeyer. This doesn't imply that neural maps are a superior cognitive model for finding an answer to the research question,

but rather that the coupling between self-organizing maps needs to be changed a bit more. Self-organizing maps update not only the neuron with the highest activation like the neural maps in Oudeyer's model, but also their neighbours to a lesser extent. An SOM seems to learn faster as a result.

4. Conclusion

Research question:

What could cause a system of speech sounds to be developed in a population of communicating agents when no prior assumptions are made about the factors that influence their interactions?

Currently, it is not possible to formulate a valid answer to this question based on the results stated in the previous section. The mechanism described above could provide an explanation as to how a system of speech sounds forms, when agents with the ability to produce and perceive speech sounds are present, if a better balance can be found between the updating of the motor map and the perception map of agents. An expansion on the model described is certainly possible.

Bibliography

Axelrod, R. (1997). *The Complexity of Cooperation: Agent-Based Models of Competition and Collaboration.* Princeton: Princeton University Press.

Boe, L., Schwartz, J., & Vallee, N. (1995). The prediction of vowel systems: perceptual contrast and stability. In *E, K. (Ed.)* (pp. 185-213). Wiley, Chichester.

Bonabeau, E. (2002, May 12). *Agent-based modeling: Methods and techniques for simulating human systems*. Retrieved from PNAS: https://www.pnas.org/doi/10.1073/pnas.082080899

Carlson, R., Granstrom, B., & Fant, G. (1970). *Some studies concerning perception of isolated vowels.* Speech Transmission Laboratory.

Comer, K. W. (2014). *Who Goes First? An Examination of the Impact of Activation on Outcome Behavior in Agent-based Models.* Washington, DC: The George Washington University. Retrieved from http://mars.gmu.edu/bitstream/handle/1920/9070/Comer_gmu_0883E_10539.pdf

Comrie, B. (1981). Language universals and linguistic typology. Syntax and morphology. In *Studies in Language* (pp. 259-286). North-Holland: John Benjamins Publishing Company.

de Boer, B. (1999). *Self-organisation in Vowel Systems.* Brussels: Vrije Universiteit Brussel.

de Boer, B. (2001). *The Origins of Vowel Systems.* Oxford Linguistics. Oxford University Press.

de Boer, B. (2021-2022). Evolution of speech (course at VUB). Brussels.

Hughes, B. D. (1996). *Random Walks and Random Environments.* Oxford University Press.

Kohonen, T. (2013). Essentials of the self-organizing map. In *Neural Networks (vol. 37)* (pp. 52-65).

Ladefoged, P., & Maddieson, I. (1996). *The Sounds of the World's Languages.* Oxford: Blackwell Publishers.

Li, P., & Zhao, X. (2013). Self-organizing map models of language acquisition. *Frontiers in Psychology 19*.

Manzo, G. (2014). Potentialités et limites de la simulation multi-agents : une introduction. In *Revue française de sociologie (Vol. 55)* (pp. 653-688). Presses de Sciences Po.

Nicolis, G., & Prigogine, I. (1977). *Self-Organization in Nonequilibrium Systems: From Dissipative Structures to Order through Fluctuations.* Wiley, New York.

Oudeyer, P.-Y. (2001). Coupled Neural Maps for the origin of Vowel Systems. In *Artificial Neural Networks - ICANN 2001* (pp. 1161-1166). Vienna, Austria: Springer.

Oudeyer, P.-Y. (2005). The self-organization of speech sounds. In *Journal of Theoretical Biology 233* (pp. 435-449). Paris: Elsevier inc.

Priyanka, A. A., Bharti, G. W., & Suresh, M. C. (2016). Chapter 1 - Introduction to Emotion, Electroencephalography, and Speech Processing. In *Introduction to EEG- and Speech-Based Emotion Recognition* (pp. 1-17). Aurangabad: Elsevier Inc.

Sejnowsky, T. (1977). Storing covariance with non-linearly interacting neurons. In *J. Math. Biol. 4* (pp. 303-312).

Vallee, N. (1994). *Systemes vocaliques: de la typologie aux predictions.* Grenoble, France: Ph.D. Thesis.