DEPARTMENT OF COMPUTER SCIENCE

Agent Based Simulation Of Language Change

Mathematics and Computer Science 20-credit Project

[redacted]

A dissertation submitted to the University of Bristol in accordance with the requirements of the degree of Bachelor of Science in the Faculty of Engineering.

Wednesday 3rd May, 2023

Abstract

The Iterated Learning Model is an agent-based model which simulates language evolution across generations. This is a process wherein an agent learns a limited sample of the language from a teacher and then generalises from these examples, when it itself becomes the teacher. Typically, this model relies on obversion, a crude inversion process first suggested by Oliphant and Batali [13]; however, this procedure can be very computationally expensive, preventing the scaling of models to greater language spaces or to contain more agents.

Here we first replicate the 'Simple Iterated Learning Model' proposed by Kirby and Hurford [10] which implements an obverter as part of the learning cycle. This model only simulates the most basic aspects of language, in part because of the limitation produced by the obverter; however, it does demonstrate an evolution towards properties more typical of a language. One such property is compositionality; informally, this is a measure of how well structured the language is. Beyond this more informal notion, a precise quantitative definition is needed. Here, a novel method for quantifying compositionality is proposed which uses binary entropy of the language the agents' use; this is compared to the more traditional correlation coefficient method for compositionality.

This paper introduces three new models that do not use obversion and analyses their results to discuss their viability as an alternative to the Simple ILM. The models differ in their implementation of each agent's neural network and the learning procedure on the network. By reviewing the success of these models, the properties of obversion within the Simple ILM learning process have been better understood; in particular, its role in preventing the language from collapsing to very few utterances.

Declaration

I declare that the work in this dissertation was carried out in accordance with the requirements of the University's Regulations and Code of Practice for Taught Programmes and that it has not been submitted for any other academic award. Except where indicated by specific reference in the text, this work is my own work. Work done in collaboration with, or with the assistance of others, is indicated as such. I have identified all material in this dissertation which is not my own work through appropriate referencing and acknowledgement. Where I have quoted or otherwise incorporated material which is the work of others, I have included the source in the references. Any views expressed in the dissertation, other than referenced material, are those of the author.

[redacted] Wednesday 3rd May, 2023

Contents

List of Figures

List of Tables

Ethics Statement

This project did not require ethical review, as determined by my supervisor, Conor Houghton.

Supporting Technologies

The following technologies were used for this study:

- `Python 3.10` and its standard library
- `PyTorch (ver 1.13.1)` library for its neural network implementations
- `NumPy (ver 1.23.5)` library for mathematical operations
- `PyPlot (ver 3.7.0)` library for results visualisations

Notation and Acronyms

ILM	:	Iterated Learning Model
SILM	:	Kirby and Hurford's Simple Iterated Learning Model, taken from [10]
RILM	:	Recurrent Iterated Learning Model
AAILM	:	Auto-Associative Iterated Learning Model
CILM	:	Contrastive Iterated Learning Model

Chapter 1

Introduction

1.1 Motivation

According to Deutscher [6], language evolution within a population is driven by three factors: economy, expressiveness and analogy. Research into how these factors apply to the change in composition of a language is of keen interest [5, 14]; as such, models have been created to try and emulate the change of language that incorporates these factors as best as possible.

One such model is the Iterated Learning Model, or ILM, which uses inductive learning to simulate language evolution [9]. This form of inducted learning involves an observation phase and a generalisation phase. During observation, a naïve agent learns what it can from a mature language user and then extrapolates this into its own language, generalising in the process. The objective is to understand the underlying syntax of language and how it emerges. Subsequently, Kirby and Hurford implemented a simpler ILM design, which uses an 8-bit language space [10]. The results of both ILMs showed that agents tend towards a language that is both expressive and compositional. An expressive language uses as much of the language space as possible, meaning that it has mappings from every signal to every meaning. A compositional language has sub-structure within its mapping from signals to meanings, for example: a fully compositional 8-bit language would mean a language where knowing one bit of the signal tells you exactly one bit of the associated meaning. An interesting detail is that there is no motivation in the model itself towards generating a compositional language, they occur as a direct consequence of the interactions between the agents.

This ILM, which will be referred to as the Simple ILM, or SILM, focused on investigating the expressiveness and analogous behaviour of language. A problem with this implementation is its reliance on using an obversion learning procedure within its training cycle [13]. Each agent has its own neural network which is trained with the purpose of providing a mapping from the signal space to the meaning space; obversion provides a mapping from meanings to signals, as shown in Figure 1.1.

However, obversion is computationally expensive which prevents the model from being scaled up with more agents or a larger or more complex language space without the use of more powerful resources, such as a supercomputer.

Also, the concept of "obverting" language does not seem realistic, as when people interact in the real-world we do not iterate through our knowledge of every possible phrase to choose the best one, we implicitly know the choice of words we wish to use.

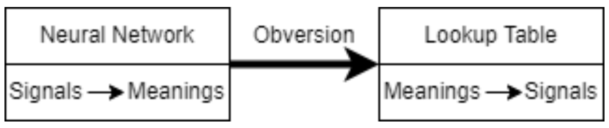

Figure 1.1: **Diagram to show the relation between an agent's neural network and their obversion lookup table.** The agent uses different mechanisms to understand incoming signals as meanings and for generating the correct signal for a given meaning, obversion is the process used to create the lookup table from the neural network.

1.2 Objectives

The aim of this project is to create a new ILM that maintains the same results and properties of the SILM, but does not use an obverter within its learning process. Consequently, the objective is to produce a computationally less expensive version of this model. The following steps will be our process in reaching the goal:

1. Recreate the SILM and compare its results with the original paper to ensure the correctness of the implementation.
2. Develop a new learning process for the ILM that does not use obversion learning, through experimenting with different configurations of neural networks and their backpropogation functions. The main focus is to ensure the models still tend toward an expressive, compositional language; additionally, through our experiments we aim to learn more about the role of obversion in the SILM.
3. From these trials, either propose a new working ILM that is computationally more conservative, or suggest the next steps in developing such a model.

Chapter 2

Background

2.1 Emergence of Language

According to Kirby and Hurford [10], language evolution occurs in relation to three systems for a species:

- Ontogeny: the learning period before a creature reaches maturity, thought to be the most important time for learning language. This is known as the "critical period" [4].
- Cultural Evolution: the change of a language community's words and phrases that come in and out of usage and the shifting of meanings and concepts within the language.
- Biological Evolution: refers to the actual mechanism for learning within the brain, that changes over time due to the evolution of the species.

In particular, the existence of a "critical period" for learning a language, or the critical period hypothesis, is a fiercely debated subject within the emergence of language research field, with many papers falling either side of the argument [1, 4, 8]. Important for the model implemented later is that, Hurford [8] suggests that the ability to learn languages is no longer encouraged after puberty as it is no longer a necessity for the brain, implying that this "critical period" for learning a language is prior to puberty.

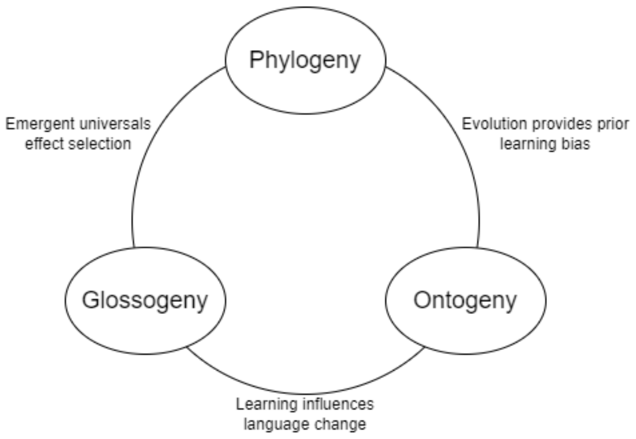
Phylogeny
Emergent universals effect selection
Evolution provides prior learning bias
Glossogeny
Ontogeny
Learning influences language change

Figure 2.1: **The three adaptive systems that give rise to language.** Some of the interactions between each system, phylogeny, ontogeny and glossogeny are shown, adapted from [10].

The difficulty of studying these systems is in the modelling of their interactions. One model, suggested prior to an ILM, was Luc Steel's Talking Heads [17, 18] experiment performed in 1999, which involved physical robot agents communicating to discuss objects in their vision at different locations around the globe. Each agent was the "brain" of the robot and would be swapped to different bodies after comlpeting a training cycle with another agent. His agents were very complex, splitting their processing into separate layers which handled the complexities of image recognition as well as language acquisition. Kirby and Hurford wanted a model that could focus solely on the evolution of a language between agents.

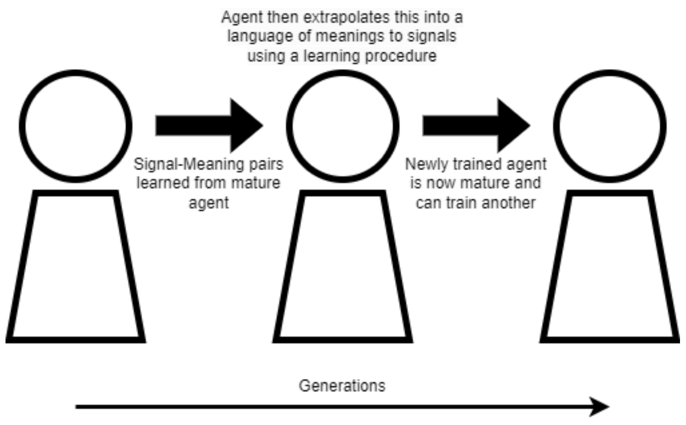
Agent then extrapolates this into a language of meanings to signals using a learning procedure
Signal-Meaning pairs learned from mature agent
Newly trained agent is now mature and can train another
Generations

Figure 2.2: **Life cycle of an agent within an ILM.** This process is repeated for each generation of the ILM and describes the tasks performed by each agent.

2.2 The Simple Iterated Learning Model

ILMs are models used to replicate the process of language evolution across generations of simulated agents, with a focus on the critical period hypothesis implemented as a bottleneck for immature agents when learning. The ILM was first proposed by Kirby in 2001 [9]. Since then, there have been other similar models proposed to investigate different phenomena within language evolution relating to different population sizes and structures [10, 14, 2]. In particular we focus on the Simple ILM proposed by Kirby and Hurford in 2002 [10].

The SILM revolves around the interactions between at most two agents for each generation of the simulation. Each generation contains a mature language user and an immature language user. The mature language user has been trained from the previous generation and they will educate the naïve agent via a finite series of utterances from a subset of its language; this subset represents the bottleneck or "critical period" for a child learning a language before puberty as, during this time, they will not be able to be taught every single utterance in this language.

When the mature agent teaches the naïve one, the immature neural network is trained on the signal-meaning pairs provided and then performs some learning procedure to extrapolate, from the subset learned, into a full mapping from all possible meanings to an associated signal.

The mature agent is now discarded and the previously immature agent is now ready to train the next agent created. This new agent has no prior knowledge, its neural network has random weightings and as such they need the previous, now mature, agent to teach them the language. This process repeats for the number of generations specified. Figure 2.2 is an illustration of a typical agent's role within an ILM iteration.

Relating back to the three systems for language evolution, shown in Figure 2.1, an ILM is a model for studying the cultural evolution of a language, since the ontogeny and biology is same for every agent due to their design and the language bottleneck in the model.

2.2.1 The Language Model

Within an ILM, language is expressed as a collection of meaning-signal pairs. A meaning represents a concept, object or idea that can be conveyed in language, whereas the signals are the spoken phrases to convey the desired meanings. In the SILM, they are both 8-bit strings, each distinct meaning and signal is assigned a unique 8-bit binary number in their respective meaning and signal spaces, an example of this is shown in Figure 2.3. This means that there are 256 possible meanings and 256 possible signals. The language itself is designed so that each bit of the meaning represents a feature of the concept being explained, similarly each bit of the signal represents a specific piece of syntax in the phrase.

There are many possible mappings between the signal space and meaning space. Only some are fully expressive, for example a one-to-one bijection from the signal to meaning space. Further, only some of these mappings are compositional. We define these terms in greater detail later in this section.

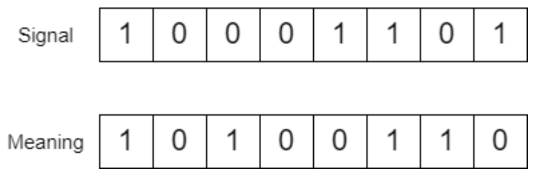

Figure 2.3: **An example signal-meaning pair from an arbitrary 8-bit language.** Within the SILM, the signal and meaning spaces are represented by 8-bit strings, as such these two strings shown are an example signal and example meaning within this language framework.

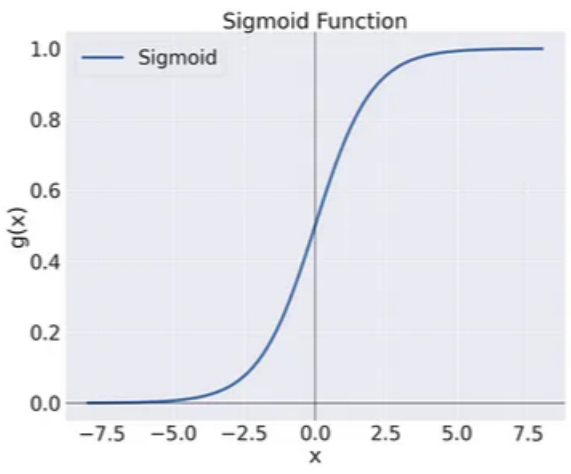

Figure 2.4: **Plot showing the sigmoid function.** This is a graph presenting the range of Equation 2.1, taken from [11].

2.2.2 Neural Networks

Each agent will need to be able to decode from signals to meanings. This is done using a neural network which will take in the 8-bit signal as eight separate inputs to its eight input nodes and will output eight probabilities across its eight output nodes. These probabilities are for that bit being a 1 in the meaning string associated with the signal string that was input to the network.

Structure and Activation

The neural network used will contain 8 nodes in each of its three layers: input, hidden and output, shown in Figure 2.5. Each node in a layer will provide a weighted input to every node in the next layer and the network will use the sigmoid function for its activation function, as shown below.

$$\sigma(x) = \frac{1}{1 + e^{-x}} \tag{2.1}$$

For our work, the input x is the output of a node in the neural network and the output from the function is used as input for the next layer of nodes.

This function is bounded, ensuring a result between 0 and 1, intended to prevent the values of the output of any layer from becoming too large. A plot of this is shown in Figure 2.4. This is most important for the output layer nodes as they need to be probabilities which must also fall between 0 and 1.

Backpropagation

To train the neural network, backpropagation is used to optimise the weights of the network to make the output as close to the desired target as possible.

The gradient of the loss function is calculated with respect to the weights of the network, so that we can apply stochastic gradient descent to update the weights. This is an optimisation procedure that changes the weights between each node in the opposite direction of the loss function's gradient, which will minimise the loss function. After a forward pass of the network is made, the error is calculated between

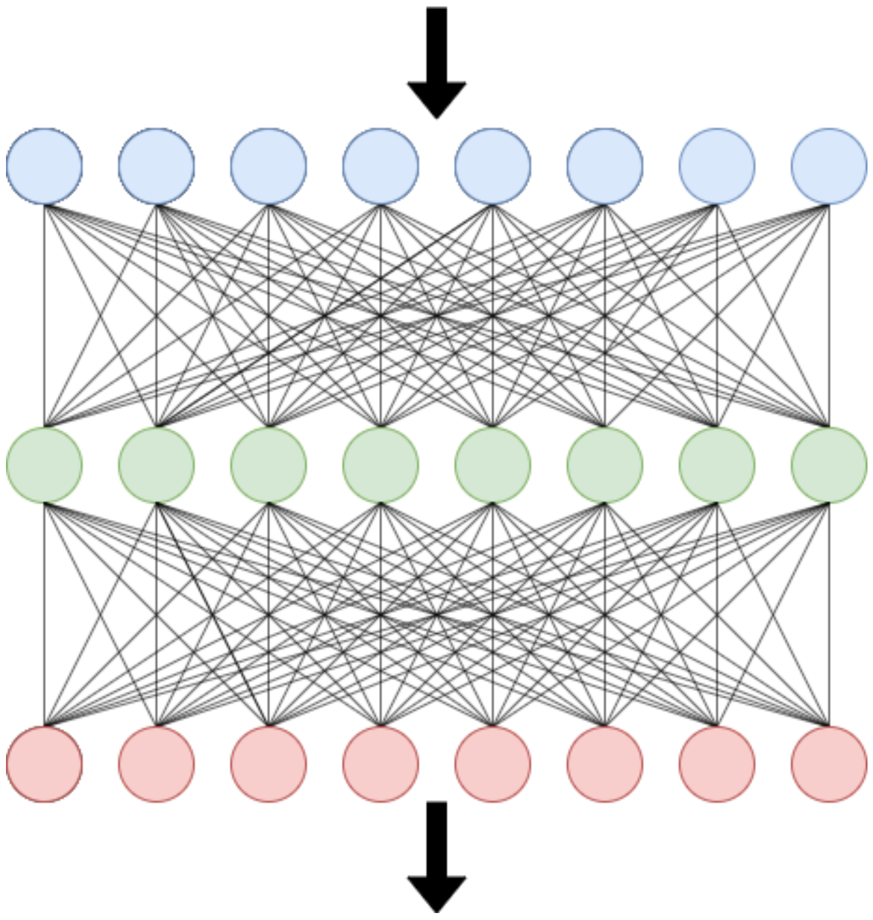

Figure 2.5: **Neural network architecture for the SILM.** Each layer contains 8 nodes and is fully connected with the adjacent layers, and the sigmoid function is used for the activation function. The blue nodes represent the signal part of the language and the red represent the meaning. Green nodes are a hidden layer that do not represent a part of the utterance. This colour convention will be used throughout.

the expected value and the actual result. This error is then propagated backwards using stochastic gradient descent, adjusting the weights of the relevant nodes, to reduce the error between the desired output and the current output of the network.

The sigmoid activation function is a popular choice for neural networks as it has properties that are desirable for backpropagation. Since the derivative is needed for stochastic gradient descent, the fact that the sigmoid function is differentiable and its derivative, Equation 2.2, is a simple expression makes it computationally efficient.

$$\frac{d\sigma}{dx} = \frac{e^{-x}}{(1+e^{-x})^2} = \sigma(x)(1-\sigma(x)) \tag{2.2}$$

Additionally, the sigmoid function has a smooth, bounded output between 0 and 1 which prevents any node in the network from becoming saturated. This would mean that it becomes unresponsive to changes in its input.

2.2.3 Measurements

The aim of the simulation is to track the degree to which the language, evolved by the agents in the ILM, acquires attributes associated with language; to track this, a set of statistics, expressivity, stability and compositionality, are defined and are calculated at the end of each iteration. Here I will introduce these statistics.

Expressivity

This measure represents how much of the language space is being utilised by the agents. It is calculated by passing the entire signal space, each of the 2^8 possible signals, through the freshly trained agent's neural network and counting the number of unique outputs generated. This is then divided by the size of the meaning space, 256, to produce a result between 0 and 1. A value of 1 would mean that an agent has a unique meaning associated with every possible signal.

$$\textbf{expressivity} = \frac{\textbf{\# of unique meanings}}{\textbf{256}} \tag{2.3}$$

Stability

After training, the stability between the two agents' languages is scored. This is a comparison between their meaning-signal pairs after obversion. For each meaning-signal pair of the teacher and student, we check if they are equal. We then subtract the number of pairs that are the same from the size of the meaning space and then divide by this size as well. This produces a score between 0 and 1 for how stable the language is between the agents, with a score of 0 representing the case that both agents share the exact same language.

Although this is the standard definition, it is potentially confusing and I have changed the terminology here.

It may seem more intuitive for a score of 1 to represent a stable language rather than 0, as such the measurement described above will be referred to as instability instead. Logically, a score of 0 for instability implies that the agents agree on all pairs.

$$\textbf{instability} = \frac{\textbf{256 - (\# of equal signal-meaning pairs)}}{\textbf{256}} \tag{2.4}$$

Compositionality

A compositional language is one where consistent parts of the signal convey definite parts of the meaning. In the real world, these languages also have emphasis on their arrangement, or syntax, to also convey parts of the meaning.

If a language is compositional within an ILM, every bit of the meaning is conveyed by one bit of the signal; however, in our 8-bit scheme, each part of the signal is always in the same location of the sting, there is no concept of syntax arrangement. For example, a fully compositional language is shown in Table 2.1.

However, a fully compositional language could also be described as a language where the first bit of the signal gives information on the second bit of the meaning. As long as these rules hold for the whole

Signal	Meaning
(0,0)	(0,0)
(0,1)	(0,1)
(1,0)	(1,0)
(1,1)	(1,1)

Table 2.1: **Simplest compositional language for a 2-bit language user.** The column on the left represents the signal space and the column on the right represents the meanings associated with each signal. This is the simplest language since it is a one-to-one mapping where the bits of the signal are the same as in the meaning. It is fully compositional since, for each column, a 0 in the signal represents a 0 in the meaning.

S	α	β	γ
x	0.1	0.2	0.7
y	0.6	0.2	0.2
z	0.3	0.5	0.2

R	x	y	z
α	0.1	0.8	0.1
β	0.4	0.2	0.6
γ	0.6	0.3	0.3

Table 2.2: **Example send and receive tables.** These tables are for a hypothetical language user. The send table on the left providing the average probability that a language user will send the indicated signal for a given meaning. The receive table on the right shows the average probability that a language user will interpret a given signal as a given meaning.

language space then it is fully compositional. An example of a semi-compositional language is shown in Table 2.5.

This measure is again between 0 and 1, with 1 being a fully compositional language. Consider figure 2.3, if this was taken from a fully compositional language then, for the whole language, a 1 in the fifth bit of the signal could imply a 1 in the first bit of the meaning. Alternatively, a 1 in the fifth bit of the signal could imply a 0 in second bit of the meaning. Either of these rules could make the language more compositional if they hold for the whole language space. Consequently, compositionality is effectively a measure of how well structured the language is that the agents are using.

2.2.4 The Language Bottleneck

Within the SILM cycle, a mature agent provides a finite series of utterances to the naïve agent to learn and extrapolate its language from. This is described as the "bottleneck" in the model.

One of the key findings from Kirby and Hurford's work is that the bottleneck influences the compositionality of the languages produced by an ILM. Considering that the language space is all possible 8-bit binary strings, there are at most 256 utterances for a language to use, so the number of utterances passed between agents is a key variable for an ILM to consider.

For example, with a bottleneck larger than 256 it is possible that all utterances of the language can be conveyed between each generation of agents. If each agent can directly mimic the language of the previous agent then the language will not change over generations. However, using a bottleneck smaller than the size of the language will cause the naïve agent to have to extrapolate the full language from this subset.

Kirby and Hurford concluded that a bottleneck of around 50 was sufficient to observe the agents in the ILM developing a compositional language across generations [10]. This is because a compositional language has rules within its structure, these rules can be captured within the bottleneck and can correctly be extrapolated and applied to the whole language.

2.3 Obversion Learning

For ILMs, the utterances, or signal-meaning pairs, are a simple language ideally forming a bijective mapping from signals to meanings. Where a bijective mapping implies that exactly one signal represents each meaning. Oliphant and Batali describe this situation as a simple communication system and propose various learning techniques that could be applied to these scenarios [13]. The paper concludes that, of the proposed methods, obverting performed better than other learning techniques in their models; this is why Kirby and Hurford elected to use such a mechanism in their simple ILM.

S'	α	β	γ
x	0.0	0.0	1.0
y	1.0	0.0	0.0
z	0.0	0.1	0.0

R'	x	y	z
α	0.0	1.0	0.0
β	0.0	0.0	1.0
γ	1.0	0.0	0.0

Table 2.3: **Example send and receive tables.** These are for a hypothetical language user that has undergone the obversion procedure in Algorithm 1 for their send and receive tables. The interpretation of the tables is the same as in Table 2.2, but now only have probabilities of either 0 or 1.

R	x	y	z
α	0.1	0.8	0.1
β	0.4	0.2	0.6
γ	0.6	0.3	0.3

$\Rightarrow$

R'	α	β	γ
x	0.0	0.0	1.0
y	1.0	0.0	0.0
z	0.0	1.0	0.0

Table 2.4: **Example obversion for an agent's receive table.** These tables represent the extent of obversion needed for the SILM. The table on the left represents the agents' probability space for understanding a given signal represents a given meaning. The table on the right represents the same, but after obversion has been performed, now each signal has a probability of 1 for exactly one of its meanings.

Obversion relies on the relation between send and receive tables 2.2, which are functions of the language user's mapping of meaning to signals and signals to meanings respectively.

Essentially, obversion is a process where, given a table of probabilities that some x is the given relation for some y, the result is a dictionary where every input has exactly one output. It does this by iterating through every combination of the inputs and outputs and calculating some 'confidence' value for every possible output for a given input. Then, the output with the greatest confidence value is selected as definitive output for that input.

Algorithm 1 The Obversion Procedure

```
For each meaning $\mu$:
    1. Find the signal $\tau$ for which $R(\tau, \mu)$ is maximum.
    2. Set $R'(\tau, \mu) = 1$, and set $R'(\sigma, \mu) = 0$ for all $\sigma \neq \tau$.
For each signal $\sigma$:
    1. Find the meaning $\tau$ for which $R(\sigma, \tau)$ is maximum.
    2. Set $R'(\sigma, \tau) = 1$, and set $R'(\sigma, \mu) = 0$ for all $\mu \neq \tau$.
```

Explained in Algorithm 1 above, obversion can be described as a process for deciding how confident an agent is on the input given to it, for a particular scenario. In our case, for understanding a given meaning from a signal or for generating the correct signal for a given meaning.

However, since the process relies on iterating through the entire output space for every input in the input space it is computationally expensive which is undesirable for any model. Also, the concept of obverting within the language user does not seem very natural. In reality, when one is conversing with another of their language population they do not iterate through every single word possible before choosing the correct one. The purpose of obversion within the ILM is to facilitate the inverting of the neural network into a meaning to signal mapping; it is not some analogy for real-word language comprehension.

2.4 Obversion in the ILM

Within the ILM, we only apply obversion one way, from meanings to signals. This is because during training, the agents' neural networks represent their mapping from signals to meanings and we use the obversion process to extrapolate their mapping in the other direction. As such the agents mapping is generated into a table, by iterating over all possible signal inputs to the agent's network and we apply obversion on just this table 2.4.

The original receive table, R, represents mappings from the signal space (α, β, γ) to the meaning space (x, y, z), where α, β, γ, x, y and z represent arbitrary signals and meanings such as the 8-bit strings used in an ILM. After obversion, R' represents a map from the meaning space to the signal space. The process is the same as described above, Section 2.3, but only for the agent's receive table.

Signal	Meaning
(0,0)	(0,1)
(0,1)	(0,0)
(1,0)	(1,0)
(1,1)	(1,1)

Table 2.5: **Example language for a 2-bit language user.** The column on the left represents all possible 2-bit signals and the right represents the associated 2-bit meaning for each signal.

Once this process is complete, the agent is now a mature language user, understanding both signals and meanings. This is important for training the next agent as we ask the mature agent to produce signals for a subset of the meaning space to be taught to the new agent.

2.5 An Entropy Measure for Compositionality

A different way of thinking about compositionality of a language is by considering its entropy. Starting with concepts discussed by Resnick et al.[15] and their use of a "residual entropy" calculation, we motivate our own measure of compositionality within an ILM using the binary entropy function, Equation 2.5, below.

If $\Pr(X=1)=p$, then $\Pr(X=0)=1-p$ and the entropy of X is given by

$$H(X) = -p\log(p) - (1-p)\log(1-p) \tag{2.5}$$

where $0log_2(0)$ is taken to be 0.

To use this, we will define Algorithm 2, which will be used for calculating the column-wise entropy of the signal and meaning space being used by the newly trained agent. Consider this example language 2.5 in the 2-bit case, which will calculate the compositionality of using the binary entropy function.

For the first column of signal bits, there are 1's in the 3rd and 4th phrase; comparing this to the first column of meaning bits, for the respective phrases there are 1's in both so the p_{11} value for column 1 is 1. Repeating for column 1 of the signals and column 2 of the meanings gives $p_{12} = \frac{1}{2}$. Similarly, for column 2 of the signal bits with each column of the meanings gives $p_{21} = \frac{1}{2}$ and $p_{22} = \frac{1}{2}$. Now the entropy's for each p, applied using the binary entropy function above, gives $H(p_{11}, p_{12}, p_{21}, p_{22}) = (0, 1, 1, 1)$. Now, for each column of the signal bits the averages are: for column 1 as 0.5 and column 2 as 1. To calculate the compositionality, we subtract these values from the number of bits, therefore: $compositionality = 2 - (0.5 + 1) = 0.5$.

Chapter 3

Project Execution

3.1 Implementing the SILM

For development of the ILMs, `Python` was selected as the environment of choice due to its range of libraries for modelling and simulation. The particular libraries used were:

- `NumPy` for its extension of mathematical functions on the standard `Python 3.10` set.
- `PyTorch` for its implementation of neural networks and their supporting methods.
- `PyPlot` for its plots and figures.

The chapter prior describes the basic overview of the SILM works, below we go into more technical detail.

3.1.1 Language Generation

Languages were generated at random for the initial agent to learn from. At the beginning of each simulation, an array of all possible 8-bit numbers is generated, where every entry is an array of length eight and containing only zeroes and ones. That was then assigned as the initial set of signals. This array of arrays was then copied, randomly rearranged and assigned as the initial set of meanings. Together they form a random one-to-one mapping from signals to meanings for the first agent of the population to use as its language.

Due to the randomness of the meaning shuffling, this language will be maximally expressive, every signal will have an associated meaning, but the level of compositionality will be random and thus will be difficult for the first agent to learn due to the bottleneck. This provides a different starting point for each run of the ILM to test how well the agents themselves can develop their own language.

3.1.2 Agents

Within the environment, agents were defined as classes, the design of which is shown in Figure 3.1, each with their own neural network and methods for learning and training another agent.

The first agent instantiated is designated mature and given a number of signal-meaning pairs from the initial language to train the next agent with. It stores these pairs as a map from meanings to signals. A new agent is initialised and designated as a naïve agent.

Now, for each generation the mature agent is asked to provide signals for a number of randomly selected meanings. These signal-meaning pairs are then given to the naïve agent to train its neural network with, for 100 randomised epochs of these pairs. The agent then uses its obverter to turn its neural network of signals to meanings into a dictionary of meanings to signals. Now the agent is designated as mature and its teacher is discarded for a new naïve agent to be taught.

This process then repeats for the number of generations specified, typically 50. Calculations for measurements are taken after the naïve agent has been trained but before its teacher is discarded.

3.1.3 Neural Networks

Each agent has a neural network object instantiated from `PyTorch`, this is used to train their ability to process signals to meanings from a subset provided by the adult agent. The same library also provides implementation for forward and backward propagation on the network.

Agent
+status: int +network: NeuralNetwork +mapping: dict
+training(): void +forward(): array +backward(): array +obverter(): dict +teach(): Agent

Table 3.1: **UML diagram for the agent's class.** Each agent has three attributes, its status, which refers to whether it is mature or naïve, its neural network and a mapping of meanings to signals it has been taught, if it is mature. Then the methods control its own learning (obversion for the SILM) and its network's forward and backward propagation. It also has a method for training a naïve agent.

```
def obverter(self, signalSpace, meaningSpace):
    predMeanings = self.forward(signalSpace)
    confTable = np.zeros((len(meaningSpace), len(signalSpace)))
    pairings = dict()
    for m in range(0, len(meaningSpace)):
        for s in range(0, len(signalSpace)):
            confTable[m][s] = self.confidence(meaningSpace[m], predMeanings[s])
        locationOfMaxSignal = self.findMaxIndex(confTable[m])
        maxSignal = signalSpace[locationOfMaxSignal]
        pairings[str(meaningSpace[m])] = maxSignal
    return pairings

def confidence(meaning, predMeaning):
    conf = 1
    for i in range(0, len(meaning)):
        conf = conf * (1-abs(meaning[i]-predMeaning[i]))
    return conf
```

Figure 3.1: **Code for performing obversion learning.** The `obverter` method and the `confidence` helper function. Not shown is the `findMaxIndex` method which simply finds the max value on the specified row of the `confTable` matrix.

For the backpropagation, initially the L2 error is used for the cost function when calculating error. This is also known as the Euclidean distance and is calculated by taking the square root of the sum of the squared differences between each output of the neural network and the target bit. The simulation was also run using the cross-entropy loss function [11] which had similar results, if not a slight improvement on the agents' ability to retain a compositional language. Both of these exist as standard functions in the `PyTorch` library.

3.1.4 Obverter

The obverter is implemented from Algorithm 1 and draws inspiration from Oliphant and Batali's description [13]. The code snippet is shown here in Figure 3.1.

To calculate the confidence of a signal being the associated pairing for a meaning, the below formula is used to calculate a score.

$$confidence = \prod_{i=0}^{8}(1 - |target[i] - predicted[i]|) \tag{3.1}$$

For the SILM, the input to the confidence function is the target 8-bit string and the eight probabilities that are the output of the neural network for a given signal. Each probability of the predicted output is subtracted from the target bit and then the absolute value of this is subtracted from 1 for each bit. Then all the bit scores are multiplied together to generate the confidence that this is the correct output.

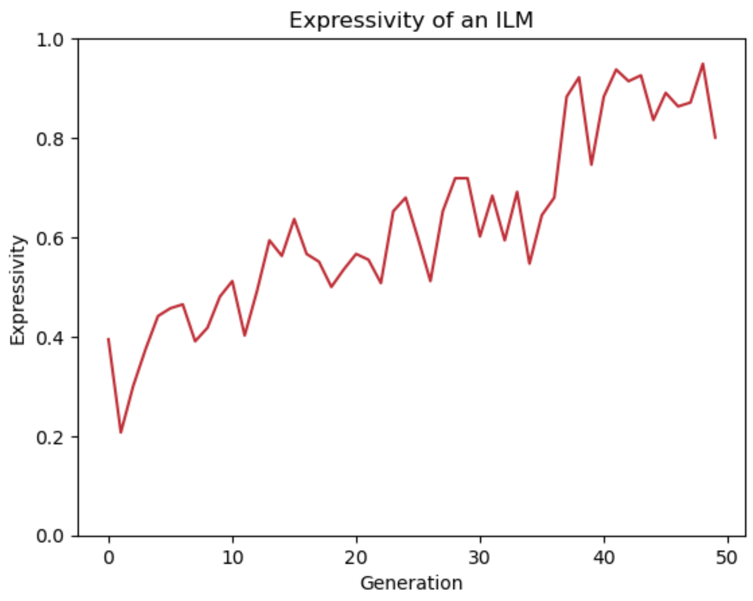

Figure 3.2: **Example of a typical expressivity graph for an ILM.** Across generations we expect expressivity to increase as the agent's language evolves to incorporate more distinct mappings.

The signal with the highest confidence for each meaning will be selected for the mapping. After the whole process, the agent will have a mapping from meanings to signals.

3.2 Measuring an ILM

To evaluate the properties of language developed by an ILM, various measures are taken of each generation to track the language being used between the agents. For our purposes, the desired result of an ILM simulation is for the agents to develop a compositional language that is both expressive and stable between agents, as shown by Kirby and Hurford [10]. Definitions of these measures were given in the previous chapter, their implementations are shown below.

3.2.1 Expressivity

Since expressivity is the measure of the language space being utilised by the agents, it is taken to be the number of unique meaning-signal pairs the agent has after learning. This is done by iterating through the signal space, using the agent's neural network to generate the associated meaning space, and counting the unique number of meanings provided and then dividing by the size of the meaning space, 256, to get a number between 0 and 1.

This calculation is performed on the newly trained agent as they represent that generation's language after going through a phase of iteration. The previous adult agent is not discarded until after all measurements are complete.

An example graph of a typical ILM run's expressivity is shown in Figure 3.2, this clearly shows the positive trend we expect for an agent's expressivity over the generations. We expect agents to improve in this way due to the language bottleneck, which encourages compositional languages. A compositional language, when extrapolated to the whole language space, will be more expressive than a non-compositional one since the inherent structure will produce more diverse utterances across the whole space.

3.2.2 Instability

When a language being used by the agents is stable, it means that they agree on as many signal-meaning pairs as possible. Therefore, the calculation of instability involves counting all the pairs that the agents disagree on and dividing by the size of the meaning space, 256, to scale the result between 0 and 1.

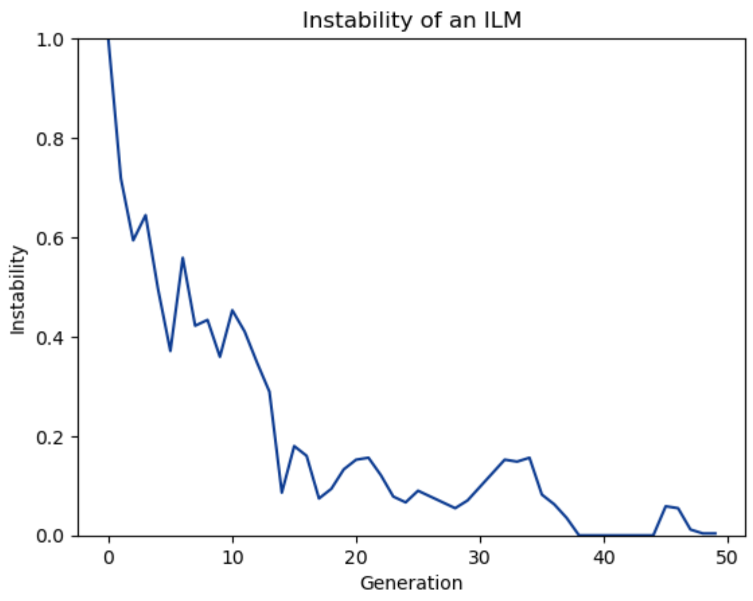

Figure 3.3: **Example of a typical instability graph for an ILM.** Across generations we expect instability to decrease as the structure of the language changes to make it easier for newly trained agents to agree with the agent that taught them.

A typical graph is shown in Figure 3.3. Over generations we expect the agents to agree more and more on the language that they are using, eventually tending towards an instability score of 0.

Rather than recomputing the newly trained agent's language, it is kept after the expressivity score is calculated and compared against the adult's from the previous generation, since their language was generated when scoring their expressivity.

After instability is scored for each generation, the previous adult agent is no longer needed and will be discarded by the program.

3.2.3 Compositionality

When calculating the previous two measures, we have already generated the language that the new agent has just learned. This language mapping is then used to calculate its compositionality.

Compositionality is calculated column-wise, we are checking for correlation between the bits in the signal against the bits in the meaning. For example, calculating whether the 2nd bit of the signal correlates with the 4th bit of the meaning. This is done for all possible combinations of positions within the bit strings.

Correlation-based Compositionality

The Pearson correlation coefficient, PCC, is commonplace in statistics and is a measure of linear correlation between two sets of data; in our case, between strings of bits. The following formula is the standard equation for PCC [7]:

$$r_{xy} = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{n}(x_i - \bar{x})^2}\sqrt{\sum_{i=1}^{n}(y_i - \bar{y})^2} \tag{3.2}$$

The inputs to the equation x and y represent the two columns of bits that we want the correlation coefficient for. Each column has its mean calculated and represented by $\bar{x}$ and $\bar{y}$, then each sum, Σ, iterates through each entry in the columns to calculate the correlation between them.

The PCC as a function has a range of $[-1, 1]$, with a 1 indicating perfect positive correlation between columns this means that a 1 in the signal bit causes a 1 in the meaning bit. A -1 for correlation is the inverse, where a 0 might imply a 1 in another bit. Either of these correlations would suggest that the language being used by the agent is compositional for that pair of bits in the signal and meaning. Therefore, for the measure of compositionality in the ILM's calculation we use the absolute value of the maximum PCC for each bit and then take the bit-wise average. This produces a measure with a range of $[0, 1]$ as required.

```
def binaryEntropy(p):
     if p == 1 or p == 0:
          return 0
     return (p*np.log2(p) + (1-p)*np.log2(1-p))
```

Figure 3.4: **Code for the binary entropy function**. The input p is a probability of a particular bit being a 1 from the output node of the neural network. It is adapted from Equation 2.5 for binary entropy.

The calculation is performed as follows, for each of the columns of the signals and meanings, the absolute correlation coefficient is calculated. This will produce 64 correlation coefficients in an 8x8 correlation matrix. Then the maximum is taken for each row of the matrix and the average of these maximums is used for the final score.

This form of compositionality calculation has been used before when studying ILMs [2]. The results of this measure were used as a baseline when trialling implementations of the following entropy-based compositionality calculation.

Entropy-based Compositionality

We implement the Algorithm 2. It takes in a language, expressed as two 256x8 matrices, representing the mapping from signals to meanings. From these matrices, a probability, p_{xy}, is calculated for each column of the signals, x, and column of the meanings, y, representing the probability of a 1 in the meaning bit given a 1 in that signal bit. This probability is then fed into the *bitwiseEntropy* cumulative variable that totals the entropies of each column of the signal bits compared to all other columns.

This is then normalised against the number of bits in the signal and added to the cumulative *compositionScore* variable. This variable is then subtracted from the number of bits in the signal, which leaves a number between 0 and 1, where 1 is a fully compositional language.

Algorithm 2 Binary Entropy Composition Procedure

```
Require: Two matrices, $S$ and $M$, such that the rows are the signal meaning pairs of the agent and the
  amount of bits in each signal or meaning, $m$.
  Let $compositionScore = 0$
  for $i$ in $0...m$ do
     Let $bitwiseEntropy = 0$
     for $j$ in $0...m$ do
        $p =$ Pr(bits in column j of M = 1 | bits in column i of S = 1)
        $bitwiseEntropy = bitwiseEntropy + H(p)$          ▷ H(X) is the binary entropy function
     end for
     $compositionScore = compositionScore + \frac{1}{m}(bitwiseEntropy)$
  end for
  return $(m - compositionScore)$
```

Within Algorithm 2, a comment explains that $H(X)$ is the binary entropy function as shown earlier. This function is implemented with the following code shown in Figure 3.4, utilising `NumPy's log2` function for performing a logarithm base-2 calculation. Unlike the formula shown earlier, the logarithms are kept positive, as we perform the necessary subtraction later in the procedure.

3.2.4 Results of the SILM

We later conclude in Chapter 4, that our findings align with what Kirby and Hurford showed [10], as such we deem that our work is successful in reproducing the original SILM.

3.3 Non-obversion ILMs

For the remainder of this chapter we present new models that represent a new ILM implementation that must satisfy two goals to be considered an improvement on the SILM.

1. The model must produce similar results to the SILM under the same conditions: expressivity increasing and instability decreasing across generations and the result being the agents using a compositional language.
2. The model must use a more computationally efficient procedure than obversion learning within its training cycle, this is to ensure the overall time complexity of the new model is better suited to scaling up.

The results of these models will be presented and discussed in Chapter 4.

3.3.1 Recurrent ILM

The greatest time complexity in the SILM's training cycle comes from the obverter calculating the mapping from meanings to signals from the neural network, which represents signals to meanings. Ultimately, the purpose of the training cycle is to provide a method by which the agent can produce the correct signal for a given meaning.

Therefore, the first idea we explore is the addition of a second neural network into the agent's architecture, a network designed to map from meanings to signals.

New Training Cycle

A second neural network that learns this reverse mapping removes the need for any separate learning algorithm. The implementation involves "attaching" a second network of the same structure to the back of the existing network as shown in Figure 3.5. The backpropagation on the network then involves optimising loss on two conditions:

1. The inputted signal is the same as the outputted signal after all layers.
2. The correct meaning is preserved in the middle layer that corresponds to the given input signal.

If successful, the agent's network could be abstracted into two separate operations, the first 3 layers encode signals to meanings and the last 3 layers can be used to reproduce the relevant signal for a given meaning.

This architecture is inspired by recurrent networks of neurons within the brain, mentioned in studies such as [20], and such could be used to draw parallels with evolutionary processes around language acquisition. As such, we dub this implementation the Recurrent Iterated Learning Model, RILM.

3.3.2 Auto-Associative ILM

Building on the concepts behind the RILM, the next consideration for an ILM was using an auto-associative neural network [12]. This model will be referred to as the Auto-Associative Iterated Learning Model, AAILM. This model was inspired by the possibility that the CA3 area of the hippocampus, known to be instrumental in memory, can be modelled as an auto-associative network [16].

Auto-Associative Neural Networks

These networks are different to a standard neural network as they are trained to reproduce the input given to them and then should be able to recognise the correct pattern from a partial input given to it. The network has the same number of input and output neurons but has a hidden layer, smaller than the outer layers, designed to compress the information given by the input layers and still produce the same activation on the output layers.

In our case, we will train the network to reproduce the signal-meaning pair supplied to it. After training the aim is that, by showing the network one part of the signal-meaning pair, the signal or the meaning, it will be able to perform pattern completion on the partial input and generate the correct signal-meaning pair.

This will remove the need for any additional training as the network is designed in such a way for the agent to understand a signal or a meaning supplied to it. As before, we will use the same activation function for the network but the code for the network and the agent teaching another agent had to be adapted to fit this new concept.

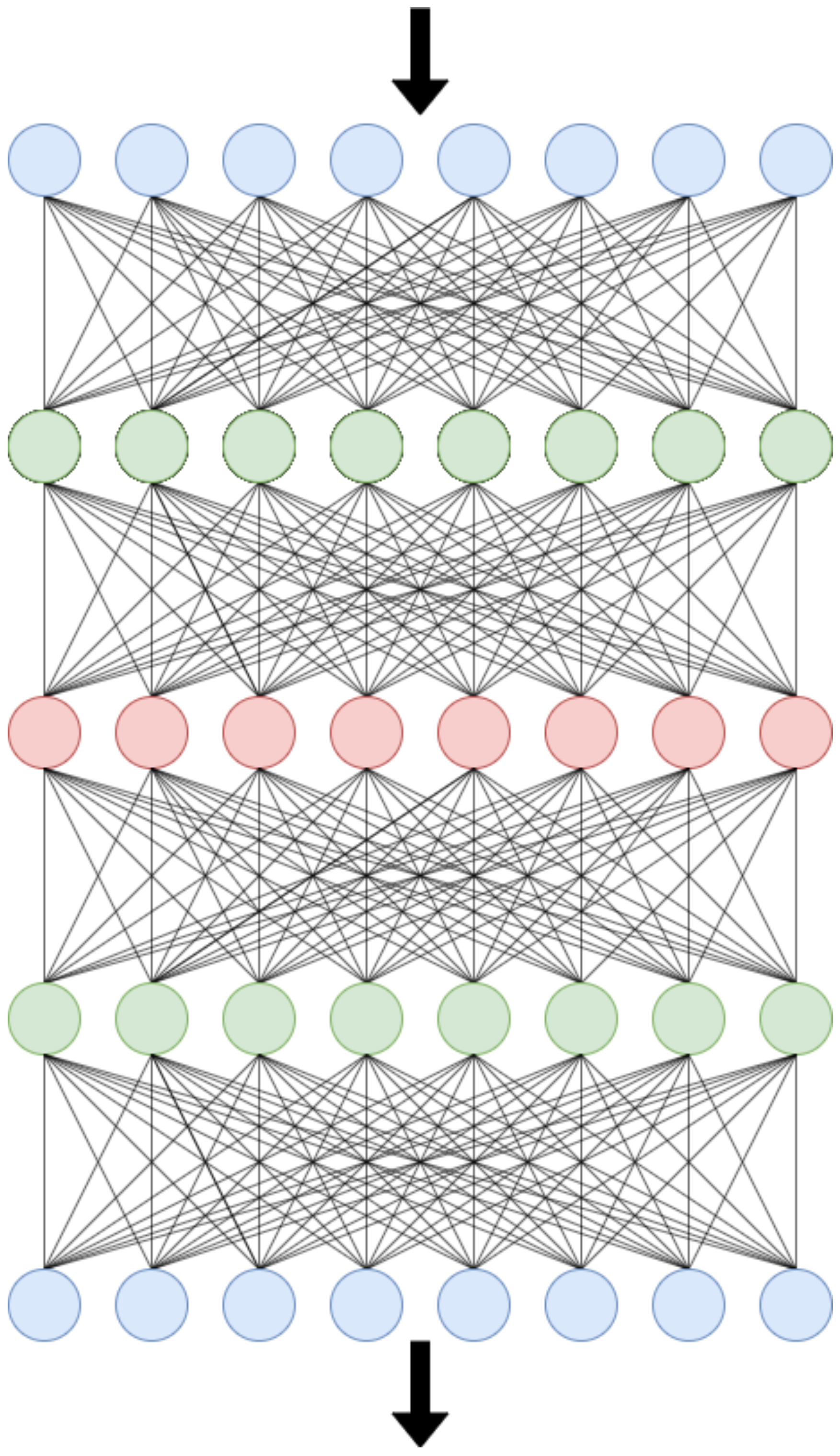

Figure 3.5: **Neural network architecture for the Recurrent ILM.** Each layer contains 8 nodes and is fully connected with the adjacent layers, and the sigmoid function is used for the activation function. There are three hidden layers along with the input and output layers. The blue nodes represent the signal bits and the red represent the meaning bits for a given utterance to be trained.

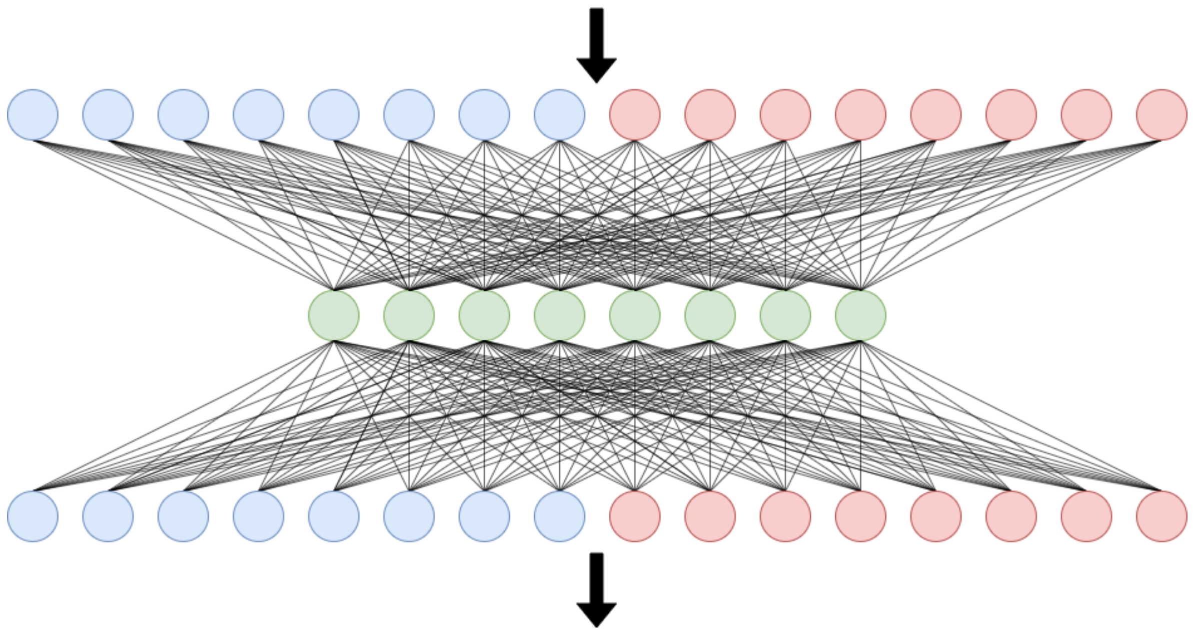

Figure 3.6: **Neural network architecture for the AAILM.** The input and output layers contain 16 nodes to input the 8-bit signal and 8-bit meaning as a pair. The hidden layer contains only 8 nodes, this is the unique feature of an auto-associative network. The network uses the sigmoid activation function. Each layer is fully connected with the adjacent layers. The blue nodes represent the signal and the red represent the meaning bits.

New Network Structure

For our auto-associative neural network, we have 16 inputs relating to the 8 bits of the signal along with the 8 bits of the meaning. These inputs are then fed through a hidden layer which only has 8 nodes, half the size of the input. This structure, as shown in Figure 3.6, is what makes the auto-associative network function and is critical to the process later when trying to reconstruct a whole input from a partial one. All layers are still fully connected between each other.

There are also 16 output layers and, when performing backpropagation, the aim is to ensure that the input layer's activation matches the output layer's. Two objective functions were used for the backpropagation, the Euclidean distance and the cross-entropy loss.

With this setup, the adult agent presents its selected signal-meaning pairs to the child agent and they attempt to learn this mapping into their network. This is the only form of training required as now, in theory, the child agent can translate between signals to meanings or vice versa.

Performance

Figure 4.7 shows the expressivity and instability graph for the AAILM across multiple runs. Unexpectedly, this graph did not match the SILM but was similar to the result of the RILM. However, compared to the RILM, the language had a higher expressivity consistently and still had the overall instability between the agents decrease which suggests that it is an improvement on the RILM.

The results will be discussed later but ultimately for our final model, we needed a mechanism to prevent the language collapsing down to a smaller subset after each generation.

3.3.3 Contrastive Learning ILM

From the previous models, we learned that without obversion the language over time will become simpler as the basic setup of the neural networks alone does not encourage the agents to be as expressive as possible. Therefore, for this model we decided to improve the neural network's cost function for backpropagation rather than just changing its structure.

The key idea this time will be to use the backpropogation to encourage the output space of the neural network to be as separated as possible, a concept inspired by contrastive deep learning neural networks. Such networks have been applied in various use cases such as image recognition and classification [19].

As such we refer to this model as the Contrastive Iterated Learning Model, CILM.

New Cost Function

The objective of the agent being taught could be described as maximising the probability of giving the correct signal when attempting to represent a meaning. Symbolically, $\max(p(s|m))$ where s and m represent signal and meaning respectively. This is the role usually performed by the obverter, since we don't want that we need to find a new mechanism to do this within the neural network's training.

Starting with Equation 3.3, which is Bayes' Law, we can separate the components of the probability to attempt to calculate each separately.

$$p(s|m) = \frac{p(m|s)p(s)}{p(m)} \tag{3.3}$$

Taking logs of both sides of Bayes' Law gives Equation 3.4.

$$log(p(s|m)) = log(p(m|s)) + log(p(s)) - log(p(m)) \tag{3.4}$$

Within this equation, we can calculate the different terms individually. For example, $log(p(s))$ is simply the log-probability of that signal being the one chosen; this term can be ignored as it doesn't depend on the neural network and, in our model, any signal is equally likely to be chosen.

Further, $log(p(m|s))$ is the cross-entropy loss function which is already commonly used in neural networks as a loss function [11].

The important term is $log(p(m))$, where $p(m)$ is defined as follows.

$$p(m) = \sum_s p(m|s)p(s) \tag{3.5}$$

where again $p(s) = 1/256$ as all signals are equally likely. Equation 3.5 is not dissimilar from the role of the obverter, calculating the confidence, or probability, that a given signal is correct for this meaning. However, the same issue would occur if we tried to calculate this at every training loop of the neural network as we need to pass the whole signal space through the equation to get a value for $p(m)$.

However, we could instead do a one-off estimate for this value using the previous iteration of the neural network. Effectively, $p(m) \approx p(m|s')$ where s' is the previous signal. If we retain the previous output vector, we can use that in the current loss function as a contrastive term in the cost.

To summarise, we can compare the current output of the neural network and the previous output to influence our cost function into pushing the output space apart and preventing a collapse in the language being used by the agents. Now our loss function is cross-entropy loss along with a contrastive term preventing the same outputs from occurring too regularly.

Balancing Cost

When running simulations of this model, the results varied greatly depending on a few critical factors placed upon the model. Namely, the weighting between the two parts of the cost function, the cross-entropy loss and the contrastive term.

For some chosen λ, the cost function is defined as:

$$\texttt{cost} = \lambda(\texttt{cross-entropy loss}) + (1 - \lambda)(\texttt{contrastive loss}) \tag{3.6}$$

where the choice of λ affects the result of the model. The different affects caused by this, and the choice of language bottleneck size, are discussed in Chapter 4. Suffice to say, it was a delicate balance of the parameters to generate Figure 4.8.

Chapter 4

Results

4.1 Entropic Composition Measure

Before using our binary entropy-based function to calculate the compositionality of a language we needed to analyse its performance on a few test cases to ensure that it behaved as expected with different language configurations. Then we can apply it to the full 8-bit ILM model to compare it to the correlation-based composition measure to confirm its validity as a measure.

Signal	Meaning
(0,0)	(0,0)
(0,1)	(0,1)
(1,0)	(1,0)
(1,1)	(1,1)

Table 4.1: **Simplest language for a 2-bit language user.** The column on the left represents the signal space and the column on the right represents the meanings associated with each signal. This is the simplest language since it is a one-to-one mapping where the bits of the signal are the same as in the meaning.

Table 4.1 presents the simplest 2-bit language, the one-to-one mapping from signals to meanings. This language is fully compositional since a 0 in the any bit of the signal always correlates to a 0 in the same bit of the meaning, the inverse is true for 1s. Our entropic compositionality function returns a 1 for this language which is what we expect.

Signal	Meaning
(0,0)	(0,1)
(0,1)	(0,0)
(1,0)	(1,1)
(1,1)	(1,0)

Table 4.2: **Another language for a 2-bit language user.** The columns of the table represent the same as explained in Table 4.1, but in this case the first two signals map to different meanings compared to the simplest language.

Another fully compositional language is shown in Table 4.2, here for the first bit a 0 indicates a 0 and a 1 indicates a 1 whilst the second bit is the opposite where a 0 indicates a 1 and a 1 indicates a 0. Our function also evaluates this with a score of 1 which is what we expect.

Signal	Meaning
(0,0)	(0,1)
(0,1)	(0,0)
(1,0)	(1,0)
(1,1)	(1,1)

Table 4.3: **A semi-compositional language for a 2-bit language user.** The columns represent the same as explained in Table 4.1. This language is semi-compositional as only one bit of the signal accurately predicts one bit of the meaning, namely if the first but of the signal is 0 then so is the first bit of the meaning and the same is true for a 1.

For the language in Table 4.3, we describe it as semi-compositional as knowing the first bit of the signal allows prediction of the first bit of the meaning; however, there is no correlation between the second bits, as such we score this language 0.5. The function evaluates this as 0.5 as well, suggesting that it is accurate for 2-bit semi-compositional languages.

Signal	Meaning
(0,0)	(0,0)
(0,1)	(0,0)
(1,0)	(0,0)
(1,1)	(0,0)

Table 4.4: **A non-expressive language for a 2-bit language user.** The columns represent the same as explained in Table 4.1. This language is non-expressive as there is only one associated meaning for all signals.

Finally, we test the function on a non-expressive language as shown in Table 4.4. The entropy compositionality score is 0, which is what we expected. This is because the language has no structure in the signal that dictates the meaning. From these 'extreme' cases we can be confident that the function is suitable for 2-bit languages.

Before moving on, we tested the function on the simplest 8-bit language, a one-to-one mapping like in Table 4.1 and it also scored 1 which is good. We then also tested the function on a variety of randomly generated 8-bit languages with all results falling in the range of 0 to 1. Therefore, we conclude that our measure is a form of composition measure, we test this against the Pearson measure for the SILM in Figure 4.4.

4.2 Simple ILM

Our first model was an implementation of the SILM, therefore to assess its success we compare it to the results shown by Kirby and Hurford [10].

Figure 4.2 shows the major result of the ILM and displays the expressivity and instability of the language, the two properties Kirby and Hurford were interested in. The trend of expressivity increasing over the generations whilst instability decreases is exactly what was observed and such we can deem the model a successful recreation of the SILM.

In their paper, Kirby and Hurford did mention that the language the agents used by the end was typically compositional but they failed to track the development in the compositionality of the language during the simulation. Therefore, as an improvement on their findings we also plot the compositionality in the language using the PCC and our own entropic measure in Figure 4.4.

From the compositionality graph, it is clear to see that the entropic measure is more sensitive to changes in the structure of the language leading to a more volatile curve. However, both measures agree that compositionality increases over time and generally follow the same trend.

Both graphs support the conclusions reached by Kirby and Hurford and thus we accept that this model is a faithful recreation and is a good baseline to compare new ILM implementations with.

4.3 Recurrent ILM

For the Recurrent ILM, the model is clearly unable to retain an expressive language. As shown in Figure 4.6, the expressivity measure mostly decreases between generations and over the course of the

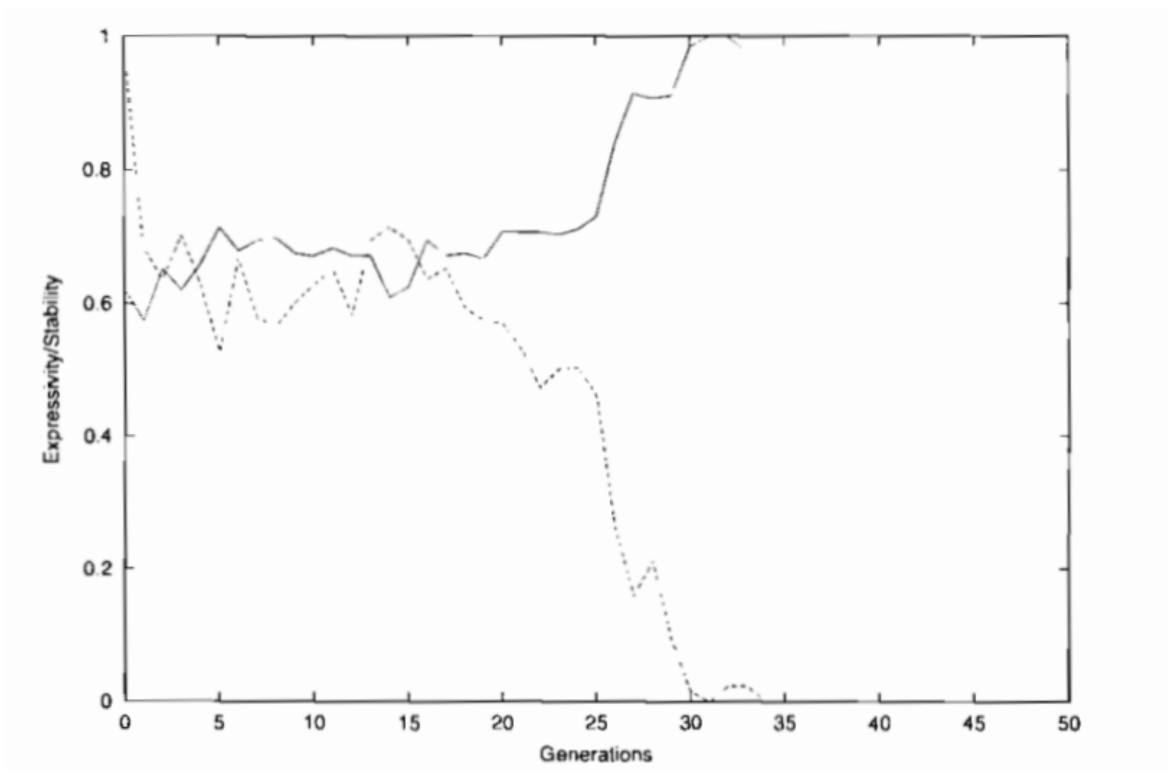

Figure 4.1: **Kirby and Hurford's Simple ILM expressivity and instability graph.** Their results showed that, over 50 generations, agents were able to evolve a language that was expressive and stable. Taken directly from [10].

simulation drops from around 0.35 down to 0.05. This curve does not behave how we expected and indicates that this model in unable to produce a language that becomes more expressive over time.

From here onward we refer to a major decrease in the expressivity of agents as a "collapse" in the language. This is a point where the space being utilised by the agents has shrunk massively. In the RILM, this is due to the neural network not capturing enough features from the utterances provided to it from the mature agent, thus leading the language trained being less structured from the one before and, ultimately, less expressive when extrapolated to the whole space.

The instability also decreases over time which is what we did expect. This at least suggests that the agents are generally agreeing more on the smaller language being used. However, compared to the SILM, the instability appears more volatile and fluctuates greater between generations. This suggests that, without an obverter, the utterances randomly selected to be taught to the new agent, from the adult agent's second neural network, are pivotal in deciding how well the new agent can replicate the language.

From Figure 4.6, the language is stable but not expressive; essentially this means that the agents have created a one-to-many mapping in their language space that they all agree on. This would be akin to using one word in a language to represent every possible meaning.

Clearly, the use of two neural networks leads to a loss of information between them. As such, a different network structure is needed to prevent the language from collapsing into a mapping that does not use as much of the signal and meaning spaces as possible.

4.4 Auto-Associative ILM

We expected the property of an auto-associative neural network, to be able to perform pattern completion [12, 3], to be a better way for the agent to extrapolate the language from the subset provided to it by the mature agent. If the network could learn a particular structure of the language from a subset, then the hope was that it would be able to apply this 'rule' to the language it generates, encouraging compositionality through enforcing these structures.

Figure 4.7 shows a similar result to the RILM from Figure 4.6. Again, the expressivity decreases over

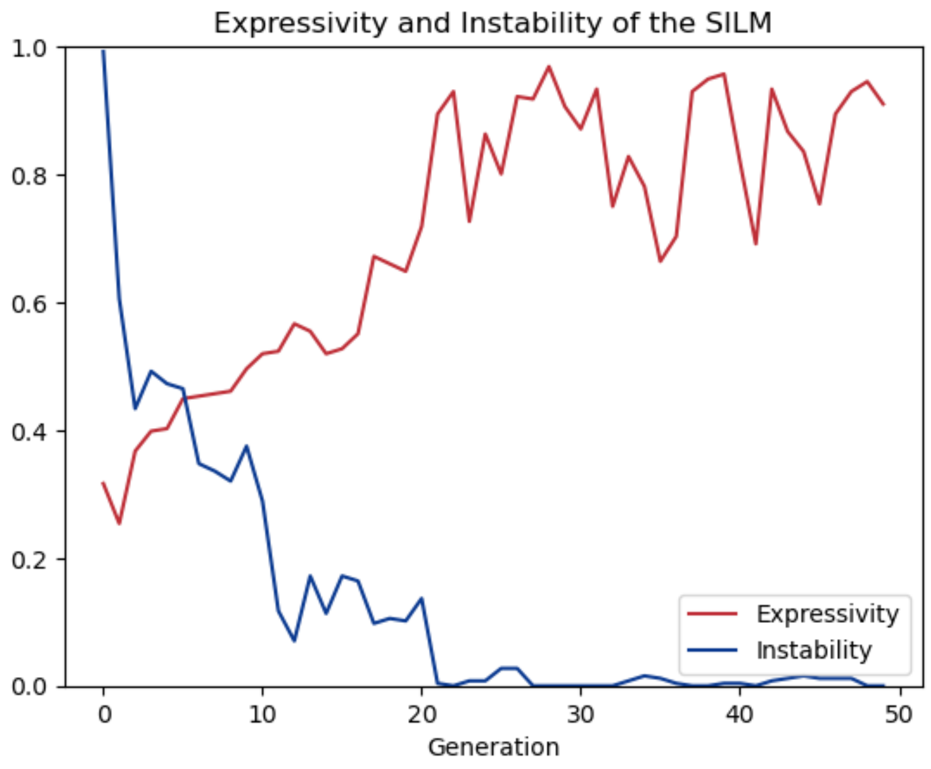

Figure 4.2: **SILM expressivity and instability graph.** This plot shows a single run of the SILM where the agents developed a language that is both expressive and stable between generations. High expressivity means that most signals had a unique meanings associated them and low instability means that the agents mostly agree on the language being used. This is very similar structurally to the results gathered by Kirby and Hurfod, shown in Figure 4.1.

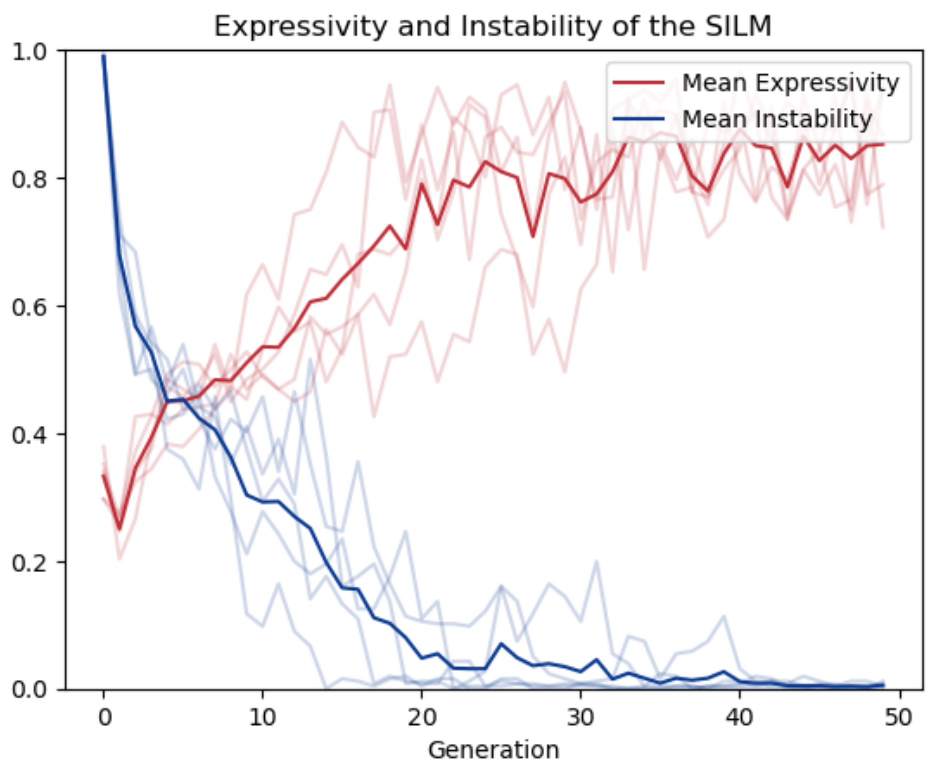

Figure 4.3: **Averaged SILM expressivity and instability graph.** This graph shows the same results as in Figure 4.2, but across five runs of the SILM where the mean scores are highlighted.

time which is not what we expected, but instability does decrease as expected. This also means that the language evolved by the end of the simulation is similar to the one described for the RILM.

An interesting feature of the expressivity result for the AAILM is that its expressivity is almost exclusively a decreasing function, it rarely increases. This implies that within the model there is a lack of pressure to encourage the language used by the agents to become more expressive over time. Evidently, from our results of the RILM and AAILM this role was performed by the obverter in the SILM.

The score also decreases in 'steps' which line up to 'spikes' in the instability score increasing. This feature is interesting and is likely caused by the random selection of the utterances used in the language bottleneck of the mature agent; when the bottleneck does not capture all the features of the language, the agents become more unstable as they disagree on more utterances in the language. When this reaches some critical point the language collapses into a smaller subset which the agents can once again agree on. In particular, Figure 4.7 shows this pattern occurring clearly twice around generation 30 and 43.

From this, we concluded that the AAILM does not produce a language with the desired properties, due to its lack of encouraging agents to become more expressive over time. This is a similar issue to the RILM; however, that model at least had greater fluctuations in the expressivity which meant that it did increase significantly between some generations. Therefore, for our third model we need to apply what we have learned about the purpose of the obverter in the SILM and our new knowledge of ILM structures.

4.5 Contrastive ILM

For the final model we present, we focused on two aspects that we feel are necessary for a non-obverter ILM:

1. A pressure within the model to encourage languages to become more expressive and that prevents a collapse in the space.
2. The structure of the network and its affect on the training cycle to allow the language to fluctuate between agents as this allows the improvement in the language.

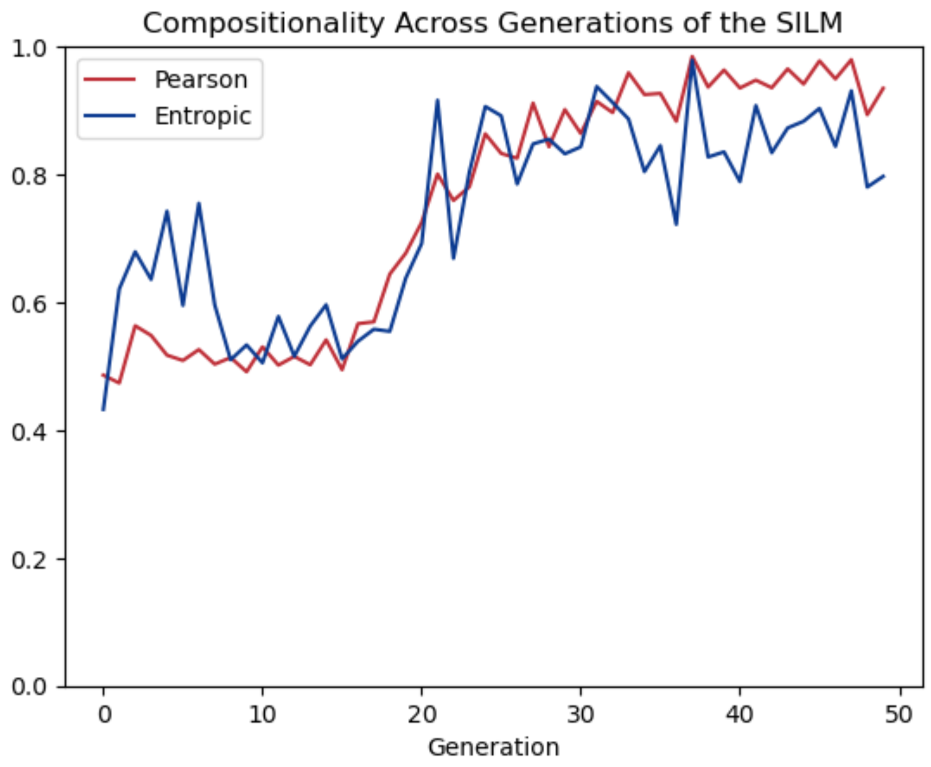

Figure 4.4: **SILM compositionalities graph.** This plot shows the compositionality of the agents' language across generations of the SILM. The blue line represents our novel entropy-based compositionality measure which seems more volatile than the Pearson correlation coefficient approach. However, both generally increase at the same points suggesting that the entropy measure is more sensitive to smaller changes in the language's structure.

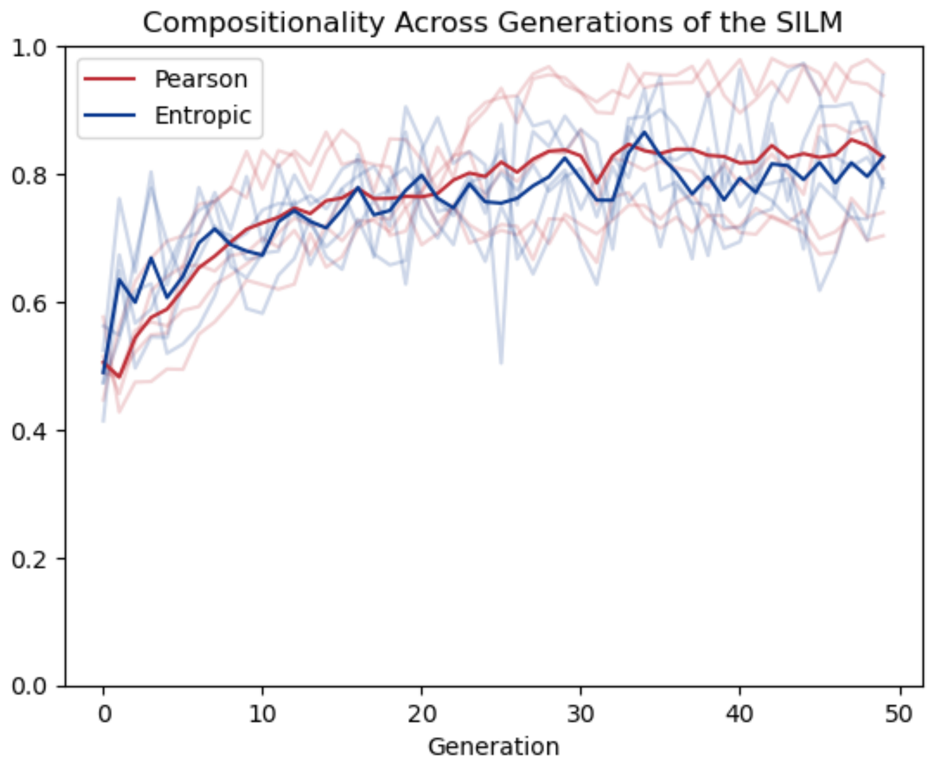

Figure 4.5: **SILM compositionalities graph.** This graph shows the same results as Figure 4.4, but across five runs of the SILM where the mean average is highlighted.

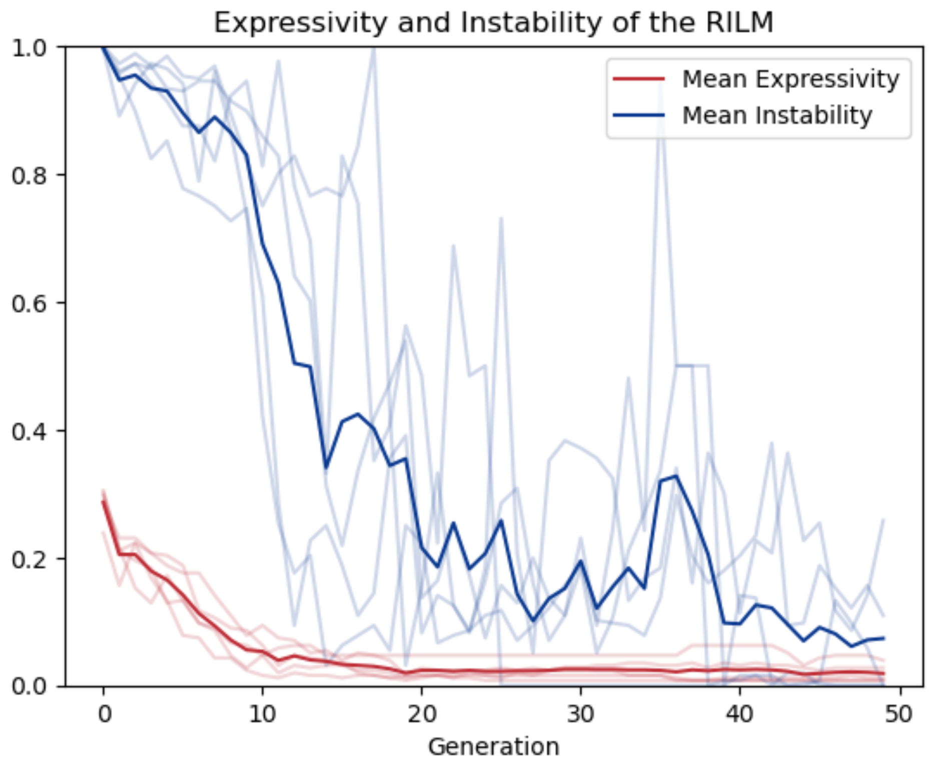

Figure 4.6: **Averaged RILM expressivity and instability graph.** This plot reports the expressivity and instability scores calculated across five runs of the RILM, with the mean average of each score highlighted. The decrease in instability is expected and desired of an ILM; however, the decrease in expressivity is not.

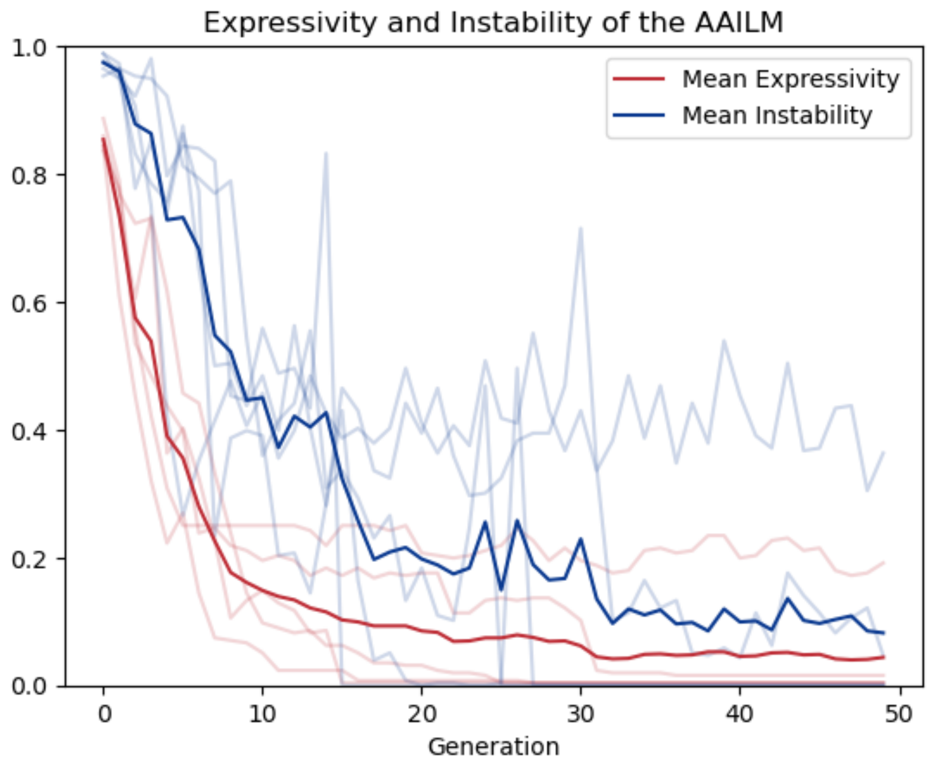

Figure 4.7: **Averaged AAILM expressivity and instability graph.** This plot reports the expressivity and instability scores across five runs of the AAILM, with the mean average highlighted. Similar to the RILM graph, Figure 4.6, both the instability and expressivity of the language decreases across generations, this was unexpected to be similar to the RILM due to the different structure of their agents' neural networks.

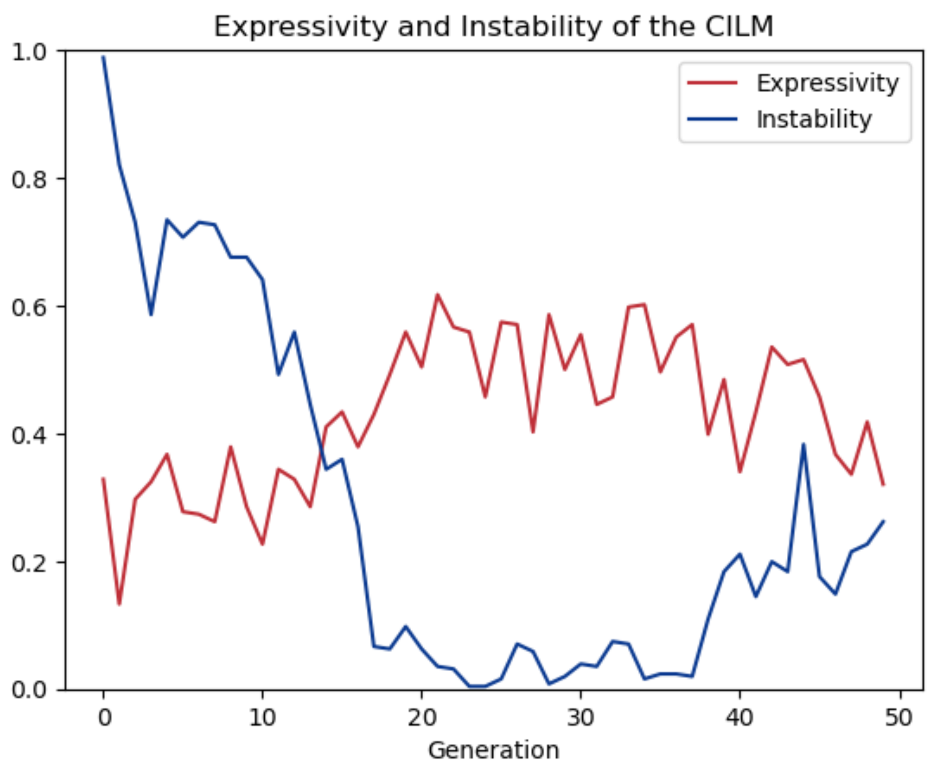

Figure 4.8: **CILM expressivity and instability graph.** This plot shows the best performance of the CILM with a particular choice of parameters for the bottleneck and cost function weightings. Similar to the SILM results, Figure 4.3, the instability decreases and the expressivity somewhat increases, but not to the same extent. From this we understand that this model is almost a replica of the SILM, but the expressivity cannot quite reach the same level at the moment.

The aim was to use the structure of the RILM, shown in Figure 3.5, along with an improved cost function for the backpropagation to use. This cost function would use a contrastive term to discourage the network from repeating the same output when training, ideally preventing the language from collapsing into only a few words.

From Figure 4.8 we see that this model was much more successful than the RILM and AAILM in encouraging an expressive language. Also, the shape of the curves are not dissimilar to those in the SILM from Figure 4.2; however, the expressivity curve does not reach the similar level and both curves are noisier. This suggests that the language being used by the agents is becoming generally more expressive over time, but is more volatile between the agents, they cannot agree as often on given signal-meaning pairs.

However, the success of the model is massively dependant on the hyperparameters selected for the cost function. For example, when the contrastive 'punishment' is given too much bearing over the backpropagation the language immediately collapses as the agents can never settle on any compositional rules, this is because when two utterances are similar but non-identical the cost function can still punish them too harshly and cause the network to produce meaningless outputs.

Ultimately, this model is the closest recreation of the SILM that we were able to achieve without the use of an obverter. The properties that it injects into the model are clearly vital to its performance as a language model. However, the CILM provides the most future promise for an ILM implementation that does not the use of a costly learning function like obversion. We discuss further suggestions for this model in Chapter 5.

Chapter 5

Evaluation

5.1 Results

From Chapter 4, our work has not resulted in a complete replacement for the ILM presented by Kirby and Hurford [10] that does not use an obverter. However, we have explored numerous avenues in the attempt of this and have learned a lot about what role the obverter plays in the SILM. Also, the third model presented, the CILM, has the most potential for being an ILM with more technical experimentation and improvement.

Most models display noisy results that can vary from run to run; this is a limitation on the ILM as a whole since the model involves random processes that may have favourable starting conditions compared to other runs of the simulation. To mitigate this, we could use more trials when producing graphs of the results distributions as this would provide more information to compare different runs of each model to understand their features.

In particular, our most interesting result comes from the CILM as this model performed similarly to the SILM. However, as mentioned in Chapter 4, this 'ideal' run had to use very specific parameters to generate this performance. This suggests that the model, as a whole, may not be a great alternative to the SILM without further investigation into why these parameters produced this result.

5.2 Implementation Critique

All models had their neural networks implemented using `PyTorch` and as such I have confidence in their results, as this library is used widely in research and industry projects alike.

The learning procedures and the model mechanisms as a whole were created myself, inspired by previous work done on ILMs and learning functions [9, 10, 2, 13, 11]. The framework of the ILMs created revolve around object-orientated principles within `Python` for the agents and for each model. This made running each model with different parameters simple for testing and evaluating different configurations. The framework used was taken from the SILM, since this was successful. I was confident to use the same code for the other models, changing functions and methods where necessary for each model's features, due to its success with the SILM.

In particular, the obverter followed the procedure proposed by Brace [2] as they filled in the gaps of implementation left by Kirby and Hurford in their paper. However, since this obverter was successfully able to replicate the results of the SILM, I am confident in the solution used.

The results generated by each model are faithful to the architecture proposed for each, the reason that they were not able to recreate the SILM was due to their own flaws, not in the execution of each model.

5.3 Evaluation of Models

Here we discuss the positives and negatives of each model's implementation and suggest reasons for the results generated.

5.3.1 SILM

As we have already concluded that this was a faithful recreation of Kirby and Hurford's ILM [10], here we criticise the model's strengths and weaknesses.

The model's results rely heavily on the size of language bottleneck used, the number of meanings the mature agent can present to the naïve agent. This number is fairly arbitrary and is meant to represent the fact that, in reality, a child can only learn so much of a language from its parental figure before it needs to be able to extrapolate the rest. This parameter of the model is necessary to facilitate the results generated and such the choice of 50 for the SILM is key.

Also, the signal and meaning spaces each being represented by 8-bit strings does not feel like a natural connection to real-world language.

However, with this setup the model is able to show that agents are able to evolve an expressive, compositional language with very little instability between generations; this was the objective of the model. Now, the debate is whether this result occurs because the model is a good representation of real-world linguistic development or whether it is solely the result of the underlying mathematics of neural networks paired with obversion learning.

Further, the obverter itself is a poor learning technique if the objective is to model the evolution of language in real-world scenarios. This is because the concept behind obversion, iterating over a language user's whole meaning space to select the correct signal to use, is extremely inefficient and is not how people within a population communicate, by thinking of all possible words and selecting the best one.

5.3.2 RILM

Our first proposed model utilised two neural networks in a recurrent pattern to attempt to extrapolate the language from the bottleneck. This felt like a neat way to remove the obverter using mechanisms already present in the ILM. However, as our results showed this was not a viable setup for an ILM.

Positives from this model was that agents still become less unstable between generations and expressivity was still able to fluctuate; it was not able to have a positive trend over generations though. Also, this model did run much faster than the SILM, but without the correct results this benefit is useless.

This model taught us that, without the proper setup, the language space used by the agents will collapse into simpler forms.

5.3.3 AAILM

The concept behind the auto-associative ILM was to remove the need for a separate training procedure after the network has been optimised on the bottleneck of the language provided to it. The extrapolation of the full language is part of the process.

Surprisingly, this ILM could not encourage the expressivity to increase whatsoever between generations and the instability was much higher than other models, but did still generally decrease over generations. The structure of this network was unsuited for allowing expressivity to increase, potentially due to the pattern completion unable to extrapolate new patterns for signal-meaning pairs, from existing rules of the language.

Even though auto-associative neural networks can be extremely powerful for the right case, they are unsuited for ILMs and as such we learned from this model that we need to be actively encouraging the language to become expressive. Evidently, this role was being performed in the SILM by the obverter.

5.3.4 CILM

For this model, we revisited the structure of the RILM as this had the better potential to be an ILM compared to the AAILM.

In particular, since the flaw of the RILM was that expressivity was not encouraged, if we could add this as a feature then hopefully the model would succeed. The contrastive learning term was able to successfully encourage the expressivity of the language to increase, to a certain level. Due to the nature of contrastive learning, the reason that it may not be able to reach a perfect expressivity is the fault of the randomness of the training process.

As explained in Chapter 3, we contrast against the previous output of the neural network. If this utterance is similar in structure to the next utterance to be learned, the cost function may be unfairly punishing the network and causing some outputs to be unreachable due to similarity with the previous outputs.

Despite this flaw, with the correct parameters, and tweaking to the cost function, the model may be able to accurately replicate the SILM's results. Already we have shown that it can encourage a language up to an expressivity of around 0.6.

5.4 Outcome

The ultimate result of this project is twofold: deeper understanding of the SILM, the obverter in particular, and the proposal of a new style of ILMs using contrastive learning techniques.

The role of the obverter within the SILM is not just to invert the neural network's mapping of signals to meanings. Inherent to the confidence calculations is a mechanism that encourages the language to be expressive. This is the case since this was the main problem encountered by our models that did not include an obverter.

Our results have shown that the most likely candidate for a new ILM that does not include the obverter is the contrastive ILM. The implementation of contrastive techniques is still fairly new and the use case here needs more investigation to be implemented effectively.

During the process we have proposed and shown a new principled method for calculating the compositionality of a language within the ILM framework. This function fits language theory better as the concept of entropy, loss of information, is related to the idea that languages evolve to minimise entropy within communication.

5.5 Future Work

Within the study of ILMs, there are some natural avenues for extra research to pursue. For example, an expansion of the signal-meaning space involving more bits would be interesting or a re-imagination of this system to better reflect language. Another path would be investigation into larger scale ILM models involving agents interacting in different ways, which has already begun [2].

However, any experimentation into increasing the size or scale of the SILM would need to also tackle the problem of the obverter which this project has attempted. Therefore I would encourage more work into this area first, once a suitable non-obversion ILM is produced, investigations will become much easier to trial.

From our results, we feel that contrastive learning techniques are the mechanism that looks most optimistic in reaching an ILM that does not require an obverter. Whether these non-obversion ILMs take the form of those presented here, or a different take entirely, an improved time complexity is crucial for deeper work with ILMs.

Chapter 6

Conclusion

This paper began with the Simple Iterated Learning Model proposed by Kirby and Hurford [10], the results of which were compared to a modern implementation using `PyTorch`'s neural network. Using this model, we developed a new method for calculating the compositionality of a language used by agents within the ILM, based on the binary entropy function. After this, we decided to focus on the obverter procedure proposed by Oliphant and Batali [13], a key feature of this ILM. We wanted to trial new models that did not use this high computation cost function, and propose a new ILM.

Throughout the project we trialled three new ILM designs, the recurrent, auto-associative and contrastive ILMs. Each was built, analysed and the results gathered were fed into our designs for the next. Each model had the expressivity and instability of their agents calculated across 50 generations in an attempt to reproduce the structure of the results generated by the SILM. From our results, none of the proposed models were able to directly replicate this model.

However, the CILM, which uses a contrastive learning approach to its cost function for the neural networks backpropagation, showed the most promising results. This model was able to achieve expressivity and instability curves of a similar structure to the SILM's; however, the expressivity did flatten out around 0.6 and couldn't reach full expressiveness.

From this, we have proposed a new investigative area of ILM research involving contrastive deep learning techniques within the neural networks of the agents. If a more suitable implementation is developed, this method could remove the need for obversion within ILMs. Thus opening more opportunities for larger ILM models to perform new experiments within the realm of agent-based language evolution.

Bibliography

[1] D. Birdsong. *Second Language Acquisition and the Critical Period Hypothesis*. Routledge, 1999.

[2] L. Brace, S. Bullock, and J. Noble. Achieving compositional language in a population of iterated learners. In *ECAL 2015: the 13th European Conference on Artificial Life*, pages 349–356. MIT Press, 2015.

[3] E. Y. Cheu, J. Yu, C. H. Tan, and H. Tang. Synaptic conditions for auto-associative memory storage and pattern completion in jensen et al.'s model of hippocampal area CA3. *Journal of Computational Neuroscience*, 33:435–447, 2012.

[4] B. R. Chiswick and P. W. Miller. A test of the critical period hypothesis for language learning. *Journal of Multilingual and Multicultural Development*, 29(1):16–29, 2008.

[5] N. Chomsky. *On Nature and Language*. Cambridge University Press, 2002.

[6] G. Deutscher. *The Unfolding of Language: An Evolutionary Tour of Mankind's Greatest Invention*. Macmillan, 2005.

[7] S. Glen. Correlation coefficient: Simple definition, formula, easy steps. Accessed on April 30th, 2023. URL: https://www.statisticshowto.com/probability-and-statistics/correlation-coefficient-formula/.

[8] J. R. Hurford. The evolution of the critical period for language acquisition. *Cognition*, 40(3):159–201, 1991.

[9] S. Kirby. Spontaneous evolution of linguistic structure-an iterated learning model of the emergence of regularity and irregularity. *IEEE Transactions on Evolutionary Computation*, 5(2):102–110, 2001.

[10] S. Kirby and J. Hurford. The emergence of linguistic structure: An overview of the iterated learning model. *Simulating the Evolution of Language*, pages 121–148, 2002.

[11] K. E. Koech. Derivative of sigmoid and cross-entropy functions, 2022. Accessed on April 16th, 2023. URL: https://towardsdatascience.com/derivative-of-sigmoid-and-cross-entropy-functions-5169525e6705.

[12] M. Kramer. Autoassociative neural networks. *Computers and Chemical Engineering*, 16(4):313–328, 1992. Neutral network applications in chemical engineering. URL: https://www.sciencedirect.com/science/article/pii/009813549280051A, doi:https://doi.org/10.1016/0098-1354(92)80051-A.

[13] M. Oliphant and J. Batali. Learning and the emergence of coordinated communication. *Center for Research on Language Newsletter*, 11(1):1–46, 1997.

[14] F. Reali, N. Chater, and M. H. Christiansen. Simpler grammar, larger vocabulary: How population size affects language. *Proceedings of the Royal Society B: Biological Sciences*, 285(1871):20172586, 2018.

[15] C. Resnick, A. Gupta, J. Foerster, A. M. Dai, and K. Cho. Capacity, bandwidth, and compositionality in emergent language learning. *arXiv preprint arXiv:1910.11424*, 2019.

[16] E. T. Rolls. A quantitative theory of the functions of the hippocampal CA3 network in memory. *Frontiers in Cellular Neuroscience*, 7:98, 2013.

[17] L. Steels. The spontaneous self-organization of an. *Machine Intelligence 15*, 15:205, 1999.

[18] L. Steels. *The Talking Heads Experiment: Origins of Words and Meanings*, volume 1. Language Science Press, 2015.

[19] E. Tiu. Understanding contrastive learning, 2021. Accessed on April 27th, 2023. URL: https://towardsdatascience.com/understanding-contrastive-learning-d5b19fd96607.

[20] Y. Wang and Q.-Q. Sun. A long-range, recurrent neuronal network linking the emotion regions with the somatic motor cortex. *Cell Reports*, 36(12):109733, 2021.