Language evolution in Multi Agent Systems

Amogh Singh
as3600@cam.ac.uk

Abstract

The evolution of language has helped improve coordination among humans and is also crucial in multi-agent systems in order to promote cooperation for achieving a shared objective. However, using human language for communication between agents may not be the appropriate choice. One possible option is to allow the agents to develop a language on their own and use it to communicate with each other and this is what I have attempted with this work. Further, the language used by the agents is analysed to check for if it possesses some of the characteristics of human languages.

Code and word count

The source code for the project is uploaded at : https://github.com/AmoghSingh25/R181_LanguageEvolution.

Total word count (excluding References and Appendix) - 3572

1 Introduction

Language plays a key role in communication and collaboration in humans. The shared set of communication rules and syntax allows for better coordination and information dissipation among the participants. Similarly, using a set of rules for communication among multi-agent systems can be useful for promoting better, compact representations that can cover the required important information. However, languages used by humans may not be best fit for usage in multi-agent systems. One method to allow for a natural emergence of language is to allow the agents themselves to devise and agree to a single set of communication rules by requiring communication in order to complete a task, thus promoting the usage of a common language.

The evolution of languages among agents can be performed using a reinforcement learning approach, providing a positive reinforcement when the agents are able to successfully collaborate and penalize them when they are unable to collaborate. A few works have looked at modelling a similar language evolution in agents [1, 5] and have demonstrated successfully the convergence of agents to a shared language.

Havrylov and Titov [1] were able to demonstrate that agents can communicate using a series of tokens and used LSTMs in their architecture. Their work was able to induce structure into language by tuning the reward structure and demonstrated the characteristics that it shared with real-world languages. Wieczorek et al. [5] demonstrated the emergence of language in agents using an encoder-decoder architecture where the intermediate output is used as a message. They also performed an analysis to understand the meaning of the messages being passed.

This work looks at modelling the evolution of language in agents and inspecting a few of their properties like generalizability, adaptability, stability and the key differences that differ messages in meaning. This is done by using two agents, an *Observer* and an *Explorer*, where the *Observer* receives the state information and communicates to the Explorer who then takes an action. One of the key changes that is performed in this work is the modification of the communication bits such that they are binary, introducing an information loss, that reduces the amount of information that can be expressed compared to continuous values and more closely resembles languages which have a discrete number of words and structures. The architecture is also modified such that the loss for the

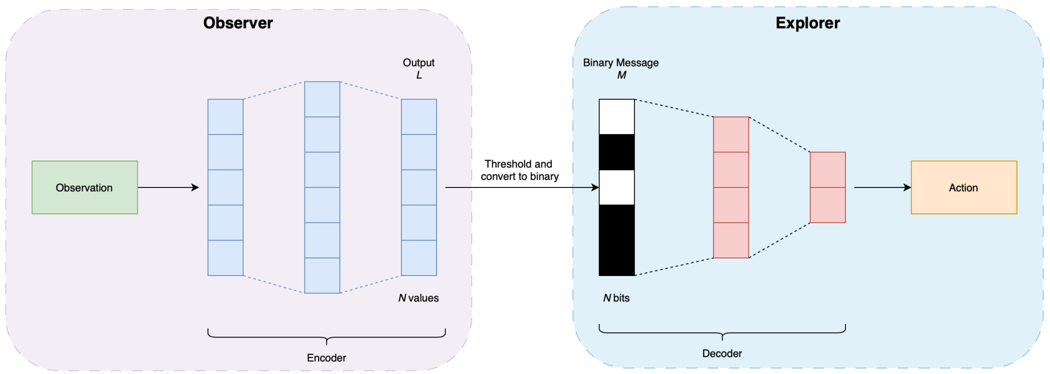


Figure 1: Architecture of the communication model

models are calculated and propagated to the *Observer* on the binary digits rather than the real valued output. This relates the feedback to the messages rather than being associated with the real valued output of the *Observer*.

Further, the characteristics of the evolved language is analysed to check if it shares any of the properties of real-world human languages.

2 Architecture and Setup

In order to simulate the evolution of language among agents, a observer-explorer setup is used where the *Observer* is able to view the environment and communicates to the *Explorer* who then takes the appropriate actions. This setup would achieve two objectives, firstly, the *Observer* would have to encode the information that is visible into a message. Secondly, the *Explorer* must use the communication and decode the meaning of the message passed and take an action. From the description, an encoder-decoder architecture would be the most suitable, with the intermediate latent dimension representing the message that would be passed on, similar to the one implemented in Wieczorek et al. [5]

In this setup, as the *Explorer* is completely blind to the environment and the only information available is the message, the agents would ideally be required to converge to a language that would be shared and understandable by both of the agents. Further, the messages are converted to a binary number before being passed on as a message. This conversion enforces structure that the agents must obey, adds in an information loss requiring better representations and allows for easier interpretability compared to continuous values.

The architecture to model communication is in the form of an encoder-decoder architecture, where the Observer receives the state information, encodes it into a latent dimension, $L \in \mathcal{R}^N$. The data in the latent dimension is then converted to a binary number, $M \in \{0, 1\}^N$, forming the message that is passed onto the *Explorer*. The binary conversion is done by checking if each number is positive or negative and assigning a 1 or 0 respectively. This method is trained using a DQN-like training [2], with the losses being propagated over from the *Explorer* to the *Observer*. A key difference in this method is that the losses for the *Observer* are computed on the discretized M rather than being computed on the real-valued L. A diagram of the architecture is given in Figure 1.

The training setup uses an initial epsilon of 0.9 which is decayed to $0.99 * e$ every epoch and the Stochastic Gradient Descent optimizer is used with a learning rate of 0.01 unless specifically mentioned. During training, a minibatch of 200 is taken from the stored memory for every epoch and is used for the DQN training. For the loss function, the Mean Squared Error loss is used for computing the difference between the outputs of the model and the Q values according to the reference model, $\hat{Q}$.

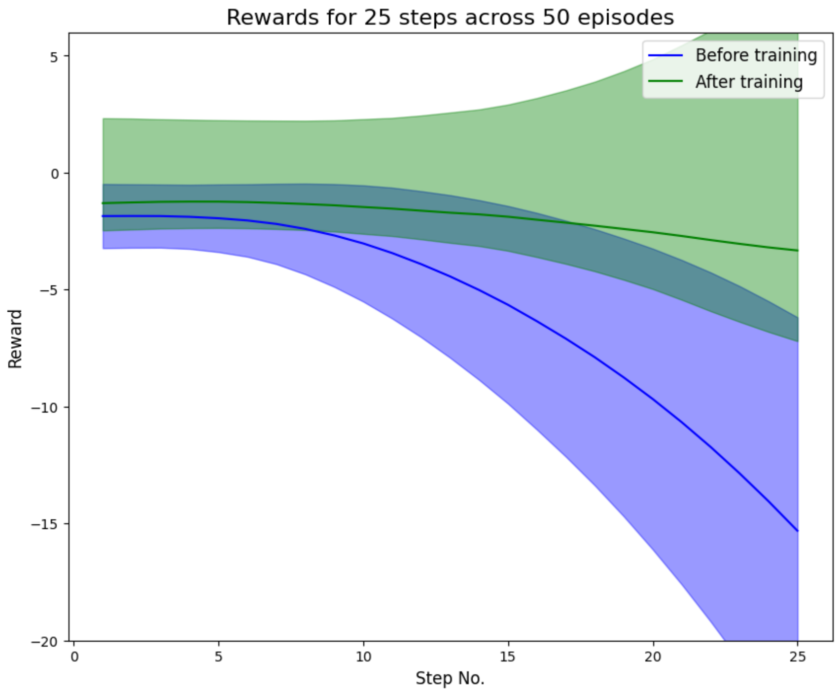


Figure 2: Reward for 25 steps averaged over 50 episodes

No. of bits	Before Training	After Training
5	-146.54 ± 85.51	-49.96 ± 30.12
8	-148.59 ± 94.81	-48.14 ± 38.48
10	-137.66 ± 86.22	-42.95 ± 24.55
16	-99.31 ± 78.50	-50.27 ± 17.04

Table 1: Total reward per episode across 50 episodes

3 Experiments

3.1 Modelling communication

The initial experiments are to confirm if it is possible for the agents to have a communication channel and to inspect if they converge such that each message has a significant link to a particular action. For this, the first environment used is the Simple environment which is a part of the Multi Particle Environments in PettingZoo [3], where the state information consists of the velocity and goal position of the agent and it is required to move to the goal position. The reward is calculated as the negative of the euclidean distance between current position and the goal position, given in Equation 1. Further information about the environment is provided in the Appendix.

$$r = -\|current\_pos - goal\_pos\| \tag{1}$$

The first step for this process is to check for the ability of the method to solve the environment as a whole when using the communication channel. This is done by comparing the rewards achieved by the trained model using communication. Figure 2 depicts the average and standard deviation for each model for each step across 50 episodes and Table 1 shows the average rewards accumulated across each episode.

By using 8 communication bits and training the architecture for 300 epochs, the method is able to converge to a communication channel, with binary numbers showing high correlation to an individual action. Figure 3 shows the heatmap of probability of each action being selected given a particular bit sequence using 8 bits. This analysis is done by collecting information from running the trained model on 1000 episodes and calculate the probability of a sequence leading to each of the possible actions.

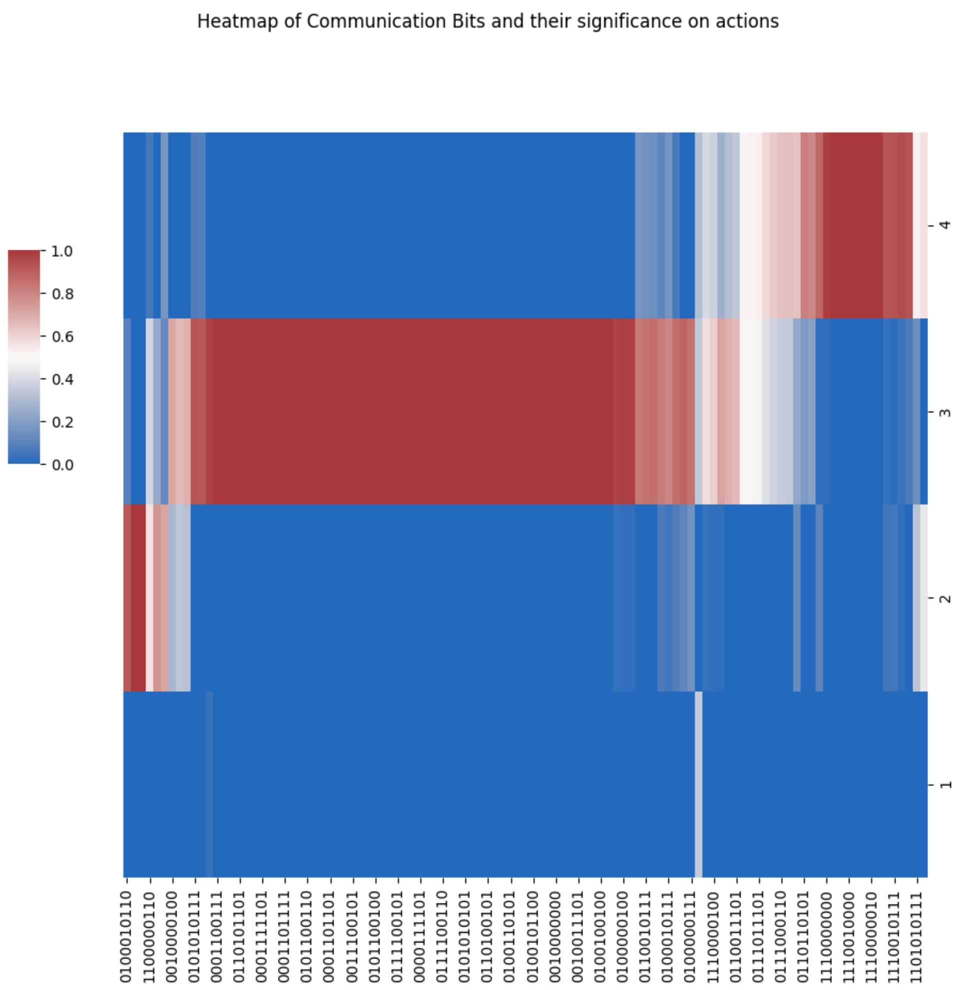


Figure 3: Heatmap depicting probability of picking an action given a bit sequence - 8 bits

The heatmap shows the bit sequences and the probability of it being associated with each of the actions. The heatmap shows the redundancy in the communication between the two agents, with multiple sequences being used for encoding a particular action. Running the model for 1000 episodes, there were no bit sequences that encoded for the action 0 which does not make any movement and the action 1 has only two bit sequences encoding it with a very low probability. An explanation for action 0 not being encoded by any of these bit sequences could be that the algorithm has not discovered its use yet as it could stop moving after reaching the landmark. Another reason could be that as the actions are discrete, the velocity of the agent is pre-defined and it may not be able to accurately place itself over the landmark and therefore never discovered using it to prevent movement. For action 1, there are very few sequences that are encoding for this action and this could be due to the relatively low usage of this action. The heatmap shows that a few bit sequences are followed by this action but they have a very low proportion compared to other. Action 2 has slightly more bit sequences with a higher probability of being associated with action 2 and this too could be related to the relatively lower use of this action.

With the current architecture and setup, a few more experiments are performed, studying the impact of varying the number of bits on the performance of the model and keeping all the other hyperparameters constant. In the analysis, 8 and 16 bits are chosen so that the resultant bit sequences could be easily inspected by converting it using UTF-8 or UTF-16 format, however most of the bit sequences used in these settings were not present in the character encoding standard. Table 1 shows the performance of the model with different number of bits, with 10 bits shown to have the highest average reward and the second lowest variance.

No. of Bits	Maximum number of sequences	Number of sequences used
5	32	18
8	256	31
10	1024	69
16	65536	233

Table 2: Available bandwidth and utilized

Learning Rate	Reward
Base model	-56.96 ± 50.97
0.001	-109.71 ± 81.23
0.0001	-53.25 ± 39.02

Table 3: Performance of language evolution

3.2 Analysis of stability of communication

One of the characteristics of a language is the constant linguistic shifts that take place in the structure and syntax of a language, with new words and meanings being added regularly to meet changing requirements of the society [4]. This experiment looks to test if the language used by the agents is constantly evolving in this setting and the parameters that are crucial for it. In order to check if the language changes, the model is trained for 200 more epochs after reaching a stable convergence, as shown in Section 3.1. Consequently, the changes in the bit sequences and their attributed actions are analysed. The learning rate is also varied as this is an important factor that controls the rate of change in the language. The learning rates used for this method are 0.001 and 0.0001 as 0.01 are used for training the base model and therefore a lower learning rate are used for subsequent training.

The analysis is performed on the 8 bit and 16 bit model and the 16 bit results are presented here with the 8 bit results presented in the Appendix. Table 4 shows the performance of models trained with the two learning rates and Table 8 shows the details about the language evolution, showing the number of sequences which are common and the number of sequences introduced after language evolution, the sequences not used in the new model and the number of sequences that had a different action than in the base model.

From the metrics, using a lower learning rate shows an increase in the number of common sequences in the 16 bit model. With a lower learning rate, the rate of change is much slower and therefore retains most of the original communication model. The data also shows an expansion in the 'vocabulary' with an increased training, with the base 16 bit model having a total of 233 sequences and all the models trained further had a higher number of total sequences.

3.3 Generalizability of communication to other tasks

A trivial property of language is that the same set of rules and sequences can be used in different situations if the content to be communicated is the same. For example, the instruction for 'move left' stays constant and does not change depending on the situation. This experiment looks to check if this also holds true for communication between agents. In order to undertake this analysis, the

Learning Rate	Total sequences	Common sequences	New sequences	Sequences missed	Common sequences with different actions
0.01	242	34	208	199	13
0.001	255	165	90	68	41
0.0001	259	198	61	35	12

Table 4: Language stability metrics - 16 bits

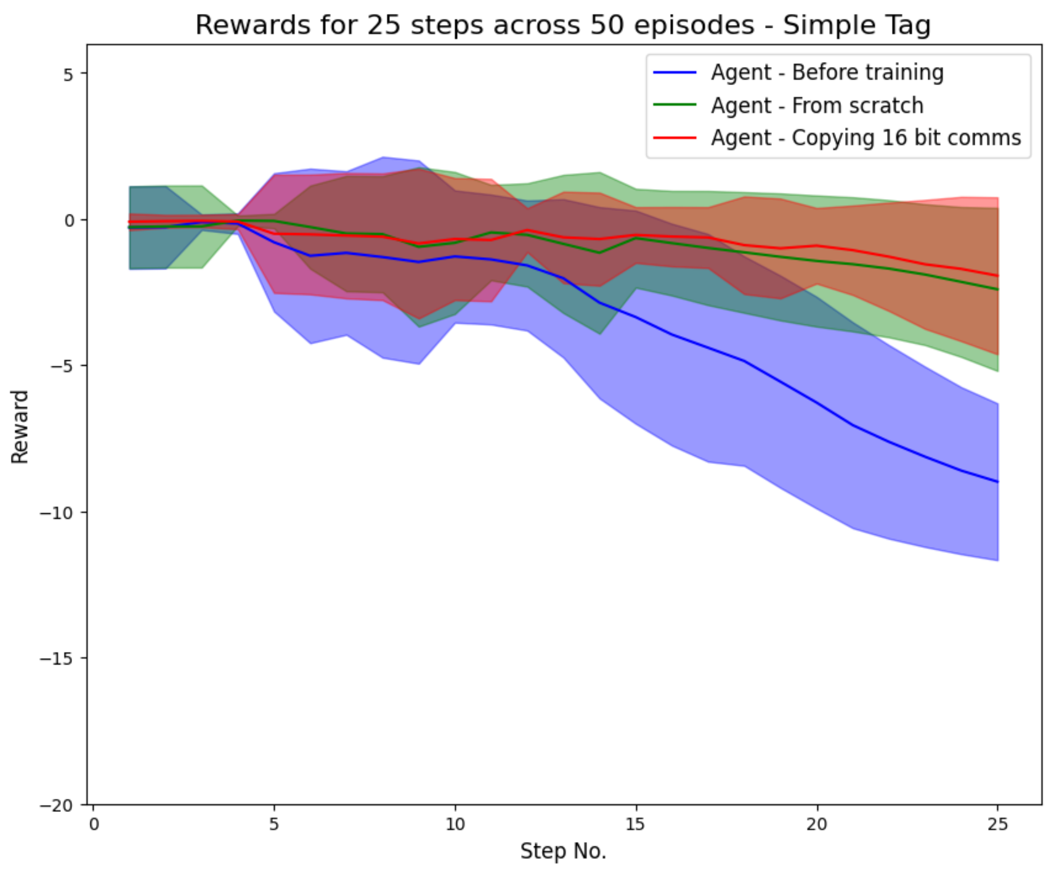


Figure 4: Reward comparison for agents in SimpleTag environment

agents are trained in the SimpleTag environment from PettingZoo [3], with the 16-bits base language model from Simple environment being used to initialize the models. A key characteristic of both these environments is that the action space is the same but the observation space and the objective of the environments is different. The SimpleTag environment consists of adversaries and agents, with the agents tasked to move to a pre-defined location while the adversaries are tasked to 'tag' the agent before it reaches the location. There are also obstacles present in the environment that both adversaries and agents have to navigate around. A full explanation of the environment is provided in the Appendix.

The agents and adversaries have a separate communication model and both are initialized with the 16-bit communication model from Simple environment. If the language that is designed by the agents is generalizable across environments, we would ideally see the agents and adversaries be able to perform well in the environment while not having major changes in bit sequences and their corresponding actions compared to the base model trained on Simple environment. From the experiment conducted in Section 3.2, it was evident that the learning rate plays a role in the magnitude of change in the bit sequences and their meanings. Therefore, this experiment uses a learning rate of 0.0001 to ensure the communication sequences are not significantly different due to the choice of learning rate. The model is trained for 500 epochs with a reduced exploration rate of 0.5 at the start of training which decays by $0.99 * e$ every epoch. Figure 4 shows the rewards achieved at each step in an episode across 50 episodes. The plot consists reward graphs for three methods, base 16-bit communication model without training, training from scratch and training using the 16-bit communication model from Simple environment.

The rewards achieved by using the pre-trained communication model closely matches the rewards achieved by training from scratch, indicating that using a pre-trained communication model does not

Table 5: Comparing average reward using pre-trained 16-bit communication model

Model	Reward
Base 16-bit model	-84.70 ± 45.66
Training model from scratch	-21.31 ± 14.91
Using pre-trained 16-bit model	-21.29 ± 22.77

Table 6: Comparison after masking bits and training for 100 epochs

Model	Reward
Initial Model	-53.85 ± 22.68
Using 0.01 initial learning rate	-159.36 ± 74.43
Using 0.0001 initial learning rate	-83.01 ± 68.21

deteriorate the performance of the model on another environment. Table 5 provides the cumulative reward achieved by the three methods, averaged across 50 episodes.

However, on closer analysis of the communication sequences, the sequences used by the two models are vastly different. The base model uses a total of 233 sequences and the model trained on SimpleTag environment uses 7211 sequences. On analysis, only 44 sequences are found to be in common, and only 26 of these denote the same action in both the models even though the actions for both the environments are the same.

One of the possible explanations for this behaviour is that as the environment is different, the architecture needs a more robust messages and this leads to the intermediate outputs in the encoder-decoder architecture to be modified to better suit the different environment. Due to the modification of the environment, the agent now has to avoid obstacles and the adversaries have to tag agents and this could lead to a modification in the messages and their meaning in order to adapt to the new environment and the changes, even if the action space remains the same.

3.4 Overcoming communication breakdown

Language can evolve to overcome restrictions like a bandwidth limit by making use of shorter representations to convey critical information. This experiment looks at the ability of the agents to evolve on imposing a barrier, in this case, reducing the number of bits available and imposing a communication breakdown. In order to test this, the agents are allowed to communicate using 16 bits of information and trained for 300 epochs. In order to represent a disturbance in communication, 8 of the 16 bits are masked with 0s and the agents are then trained to study if they are able to overcome this limitation and re-establish a language structure and regain the performance.

For this experiment, the base model is trained for 300 epochs with 16 bits. The modified architecture masks the last 8 bits of the messages and this is trained using varying learning rates for 100 epochs and their reward graphs are plotted. Table 6 shows the cumulative reward for an episode and Figure 5 shows the reward per state of an episode, both averaged across 50 episodes.

The metrics show that the method is able to achieve a partial convergence to a language but requires a very low initial learning rate in order to do so. Using a very high learning rate severely impacts the performance of the agents and would require further training but this would lead to a rewriting of the existing language and would not be an effective way of testing communication adaptability. Another key insight is the high variance in the rewards after the communication bits are masked compared to the base model. This could indicate that the model may not have completely overcome the communication mask and might require more training in order. In order to check if further training can lead to stabilization in the reward values and thus in the language, the model is trained for 300 epochs instead of 100 and the metrics are provided in Table 7 and the reward graphs in Figure 6

Training for 300 epochs improves the performance with initial learning rate of 0.01 as a higher learning rate could adapt better and discard the previous communication scheme. Training for more

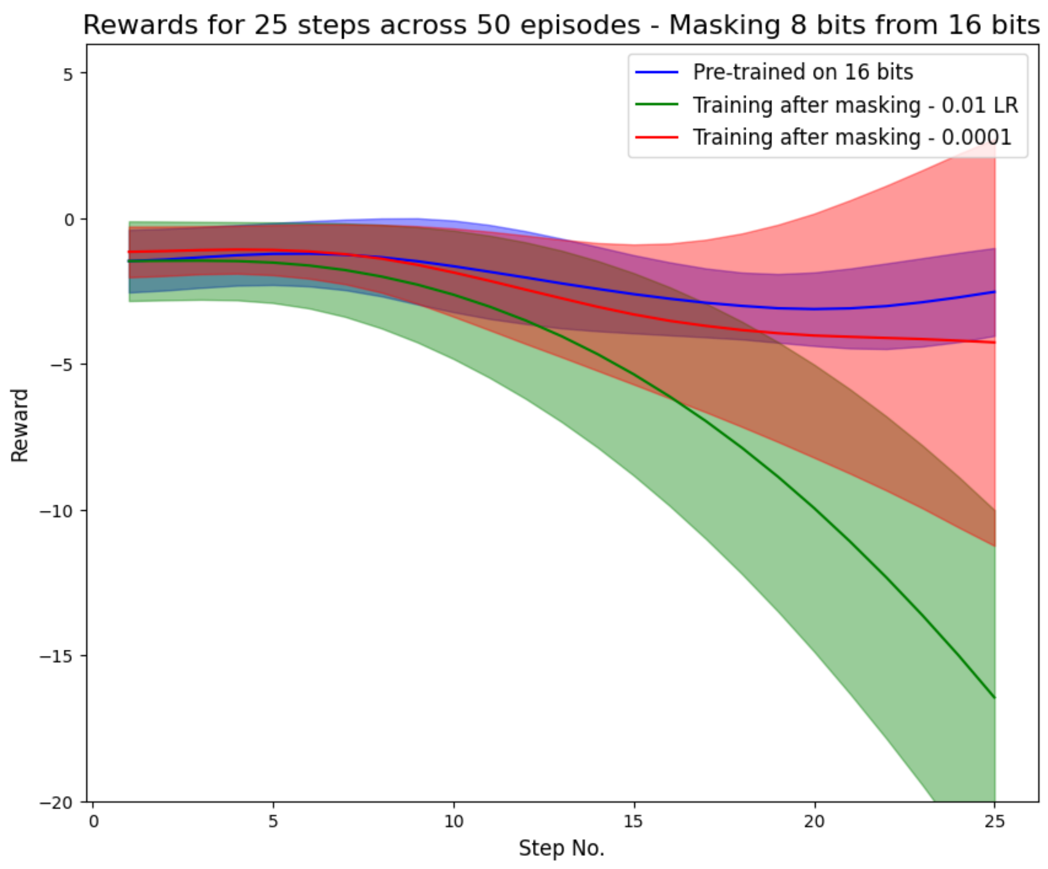


Figure 5: Reward graph for communication breakdown - 100 Epochs

Table 7: Comparison after masking bits and training for 300 epochs

Model	Reward
Initial Model	-53.85 ± 22.68
Using 0.01 initial learning rate	-84.96 ± 85.36
Using 0.0001 initial learning rate	-87.87 ± 74.99

epochs does not reduce the variance using 0.0001 learning rate, indicating that the agents are not able to completely recover from the communication breakdown.

3.5 Analysis of communication sequences

This experiment aims to inspect the sequences that are assigned to each action and better understand the differentiating factor between these sequences that makes them discernible by the agents. In order to achieve this, as the messages are binary sequences, the Structural Hamming Distance (SHD) can be a useful metric that would count the number of differences between a pair of messages. The idea with this is that the SHD of sequences assigned to the same action would be lower than the SHD of sequences assigned to different actions. Another way to explain this is that a pair of sequences with a lower SHD would be more likely to be indicative of the same action as they are more similar compared to a pair of sequences with a high SHD. In order to test this, the sequences from the SimpleTag environment experiment from Section 3.3 is used as it uses a larger spectrum of the available communication sequences. If a bit sequence was followed by a particular action for more than 50% of total occurrences, the bit sequence is assigned to the action. The communication sequences do not encode for action 3 with a high proportion and therefore are not present in this

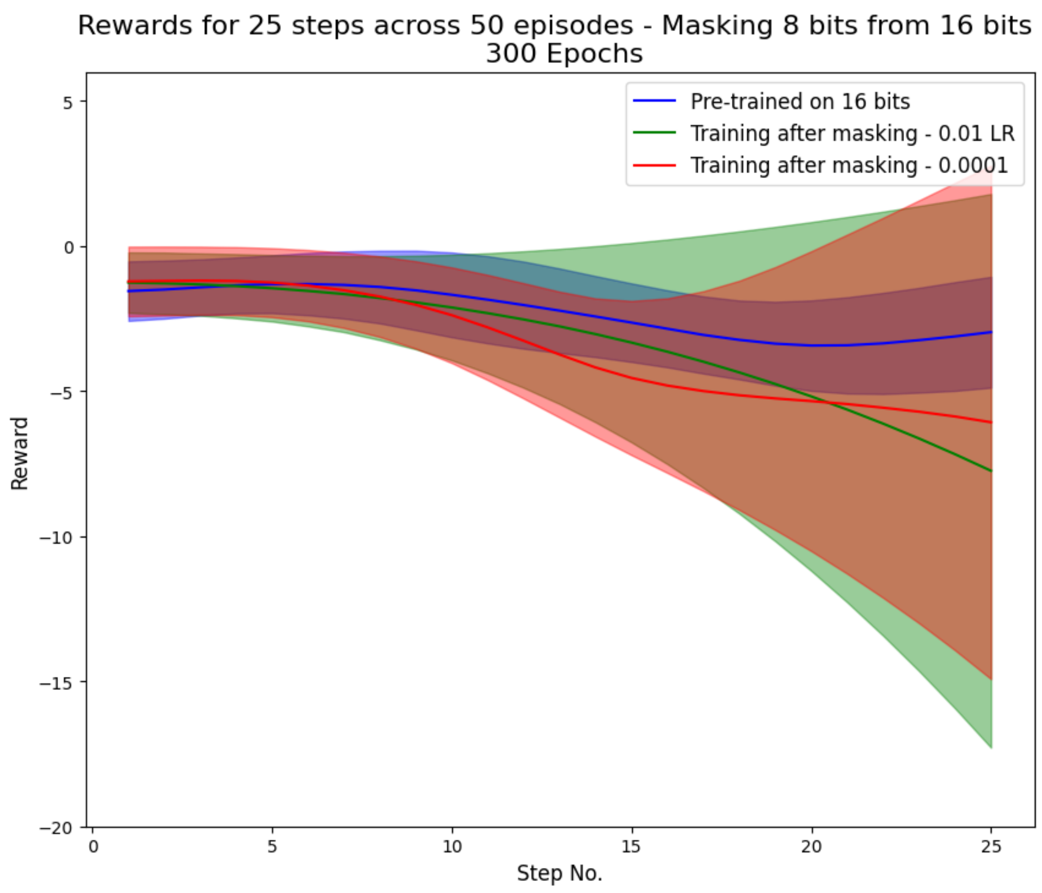


Figure 6: Reward graph for communication breakdown - 300 Epochs

analysis. These sequences are grouped according to the action they encode and the SHD between the groups and within each groups is computed. This can be easily visualized as a matrix heatmap, as shown in Figure 7, with the red regions indicating sub-sections of the same action and thus contain the intra-SHD metric. Areas outside these sections would contain SHD metrics of pairs of sequences which encode separate actions and thus provide the inter-SHD metric.

From the matrix, there is an underlying pattern that can be visualized, with the intra-SHD regions being darker, i.e. lower compared to the other regions. In order to confirm this, the SHD values are passed through a two-tailed t-test to check if the values are significantly different or are similar. Running the t-test gives a p-value of 0.016, supporting the alternate hypothesis that the two groups show significant difference. Further data relating to this are given in Table 9 and Table 10 in the Appendix.

4 Conclusion and Future Work

Through this work, the evolution of language between a pair of agents was achieved by promoting co-operation for mutual benefit. The architecture was constructed in an encoder-decoder format, with the introduction of binarization of the intermediate outputs before being passed on to the *Explorer* in order to discretize the number of possible sequences, making it similar to languages. Further analysis was undertaken on this by comparing the evolved language with characteristics of a real-world language, i.e., looking at stability of language in long term training, the application of the evolved language to other tasks, adaptability of the system to communication breakdown where part of the bits are missing or zeroed and analyse if the communication sequences are significantly different if they encode different actions. Using this architecture and the results of the analysis performed, it can be deduced that the evolved communication demonstrates a few of the characteristics of a language

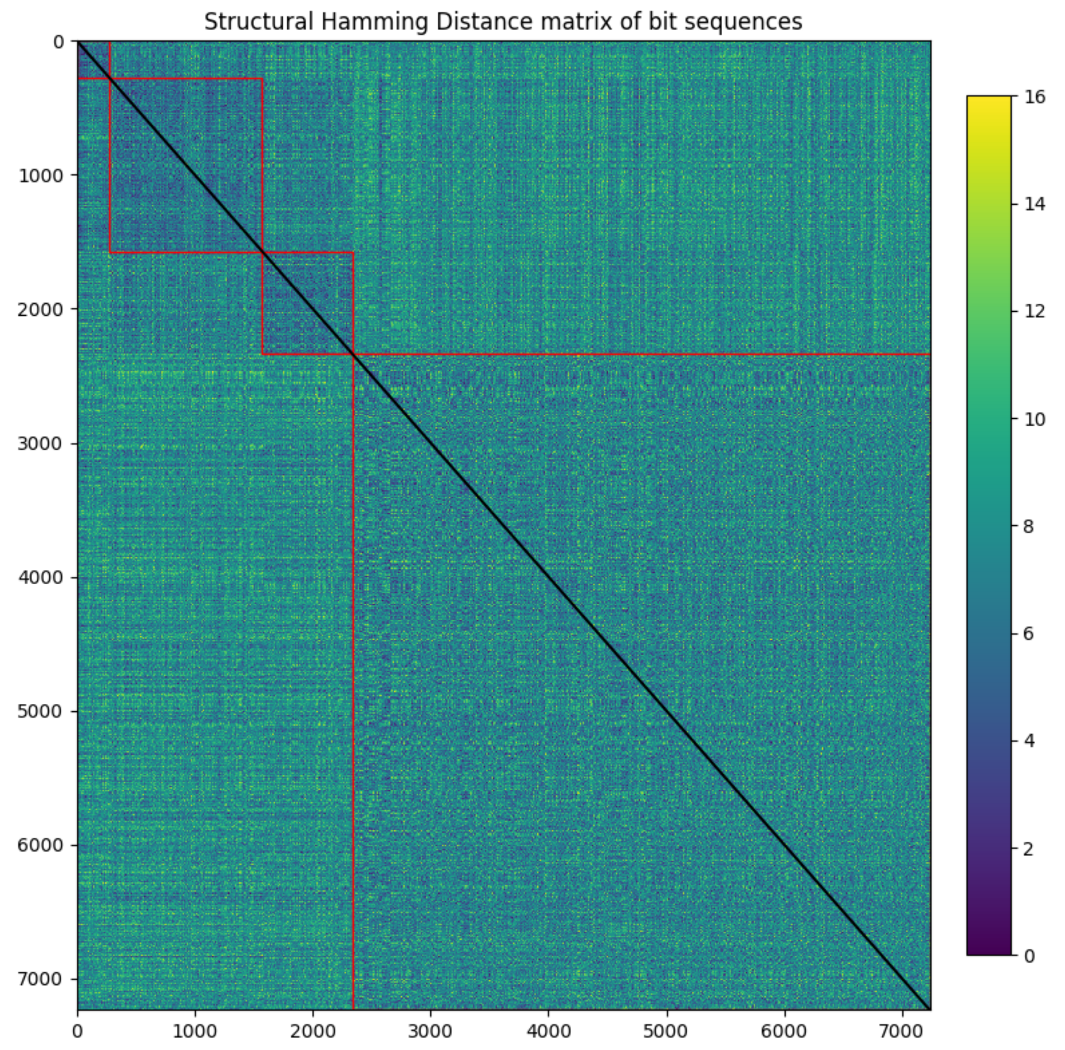


Figure 7: Matrix denoting the SHD between each pair of bit sequences. Regions within red lines indicate bit sequences of the same action.

that are redundancy, constant evolution of language and being able to adapt to a communication breakdown. However, it is unable to adapt an existing language for different tasks although the actions are the same. Overall, the architecture and the communication between the agents is shown to demonstrate many of the characteristics of a language.

The binarization of the intermediate output of the encoder-decoder model was also planned to provide easier interpretability by converting them to text using UTF, however many of the sequences were out of range of the coding standard. The communication bandwidth is also not completely utilized by the models, with an approximate 99% of it still being unused. A future work could look at making modifications to allow for better interpretability and improving the utilization of the available bandwidth. The architecture of the system could also be explored so that it could demonstrate generalizability to different environments and other characteristics which were not tested here.

References

[1] Serhii Havrylov and Ivan Titov. *Emergence of Language with Multi-agent Games: Learning to Communicate with Sequences of Symbols*. en. arXiv:1705.11192 [cs]. Nov. 2017. DOI: 10.48550/arXiv.1705.11192. URL: http://arxiv.org/abs/1705.11192 (visited on 02/19/2025). 1

[2] Volodymyr Mnih et al. *Playing Atari with Deep Reinforcement Learning*. arXiv:1312.5602 [cs]. Dec. 2013. DOI: 10.48550/arXiv.1312.5602. URL: http://arxiv.org/abs/1312.5602 (visited on 03/28/2025). 2

[3] J. K. Terry et al. *PettingZoo: Gym for Multi-Agent Reinforcement Learning*. arXiv:2009.14471 [cs]. Oct. 2021. DOI: 10.48550/arXiv.2009.14471. URL: http://arxiv.org/abs/2009.14471 (visited on 03/28/2025). 3, 6

[4] Anastasia Thanukos. "A Look at Linguistic Evolution". en. In: *Evolution: Education and Outreach* 1.3 (July 2008). Number: 3 Publisher: BioMed Central, pp. 281–286. ISSN: 1936-6434. DOI: 10.1007/s12052-008-0058-3. URL: https://evolution-outreach.biomedcentral.com/articles/10.1007/s12052-008-0058-3 (visited on 03/28/2025). 5

[5] Tobias J. Wieczorek et al. "A framework for the emergence and analysis of language in social learning agents". en. In: *Nature Communications* 15.1 (Aug. 2024). Publisher: Nature Publishing Group, p. 7590. ISSN: 2041-1723. DOI: 10.1038/s41467-024-51887-5. URL: https://www.nature.com/articles/s41467-024-51887-5 (visited on 03/16/2025). 1, 2

A Appendix

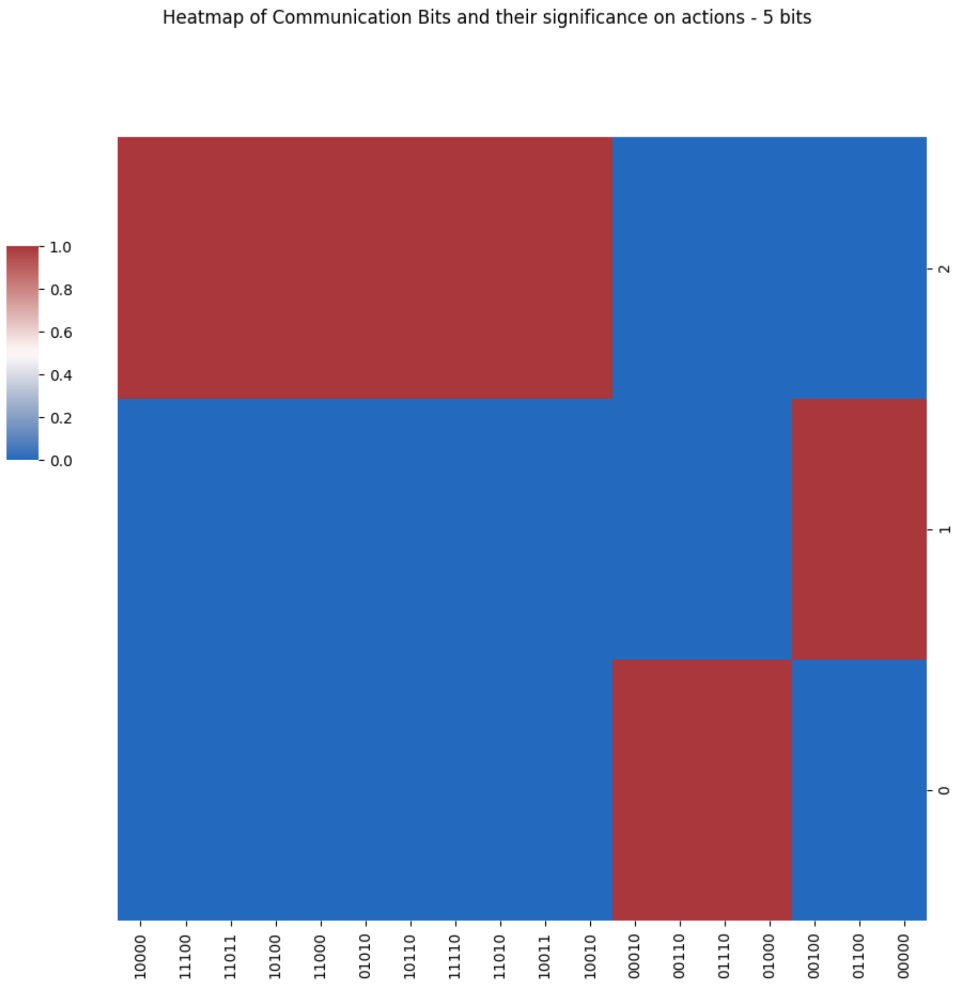


Figure 8: Heatmap depicting probability of picking an action given a bit sequence - 5 bits

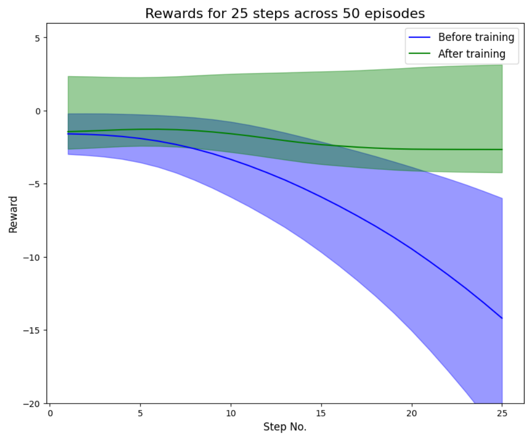
A.1 Environments used and their description

Simple and SimpleTag environments from the PettingZoo library are used for the experiments performed in this work. The Simple environment consists of a single agent and a landmark that it needs to reach. The agent has 5 actions available, which are Left, Right, Up, Down and None, and the observation space consists of the position of the landmark and the relative position of the landmark.

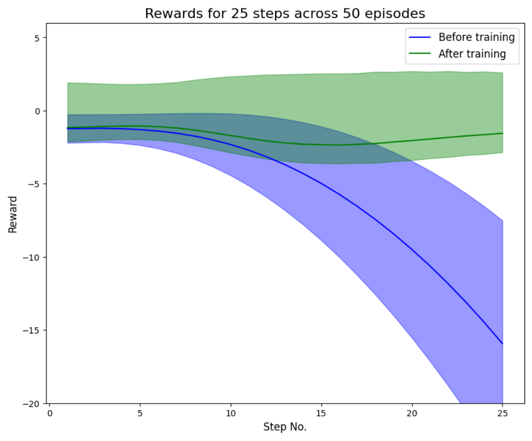
The SimpleTag environment is an extension of this environment and consists of one agent and multiple adversaries. The objective of the environment is for the agent to reach a landmark and the adversaries must 'tag' the agent before it reached the landmark. The action spaces for each of the participants is the same as in the Simple environment and the observation space of each participant is their velocity, position, landmark position and the position and velocities of the other participants in the same group (agent or adversary). Both of the environments reward the agents as negative of the euclidean distance between the landmark and the agent. The adversary in the SimpleTag environment are rewarded positively if they tag an agent or negatively if they fail to tag an agent.

A.2 Modelling communication

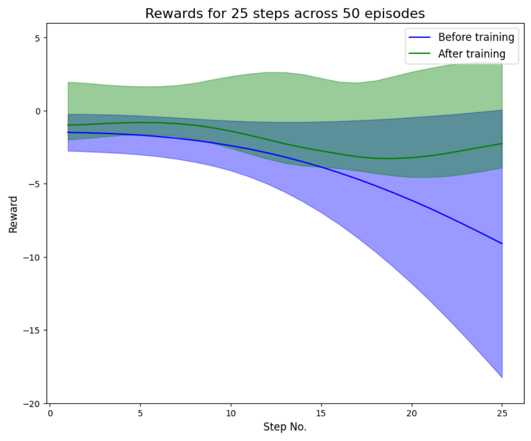
The reward graphs for 5, 10 and 16 bits is shown in Figure 9a, Figure 9b and Figure 9c respectively.



(a) Reward graph for 5 bit model



(b) Reward graph for 10 bit model



(c) Reward graph for 16 bit model

Learning Rate	**Total sequences**	**Common sequences**	**New sequences**	**Sequences missed**	**Common sequences with different actions**
0.01	28	15	13	18	3
0.001	33	23	10	10	7
0.0001	30	27	3	6	4

Table 8: Language stability metrics - 8 bits

A.3 Stability of communication

The base 8 bit model has a total of 33 sequences and the metrics relating to the training for stability is given in Table 8.

A.4 Analysis of communication sequences

Further information relating to the statistical test performed to check if the differences in SHD between actions and within actions are given in Table 9 and the t value and p value of the t-test are given in Table 10.

Group	Number of samples	Mean of samples
Intra-action	4	6.96
Inter-action	6	7.70

Table 9: Intra- and Inter- action statistics

Statistic	Value
t value	-3.05
p value	0.016

Table 10: Intra- and Inter- action statistics