Efficiency of transmitted information in simple communicative agents

Achraf Azize and Othman Gaizi

achraf.azize@polytechnique.edu, othman.gaizi@polytechnique.edu

1 Introduction

Studying basic properties of emerging languages in basic communication games between neural nets can help bridge the gap between natural languages and artificial ones. In this project, we focus on one these properties, namely the length distribution of the messages that two neural networks playing a simple signaling game come to associate to their inputs, in function of input frequency. We intend to review and compare some of the architectures developed in the literature: first we will try to reproduce the anti-efficient coding behaviour when using two standard agents (Chaabouni et al., 2019), then add the LazImp constraint from (Rita et al., 2020) to reproduce the Zipf Law of Abbreviation (ZLA) distribution in the emergent code. Finally, we propose a new communicative system (IntEmo) formed of an *interrupting listener*, who tries to interrupt the speaker as soon as he understands the message, and an *emotional speaker* who hates being interrupted.

2 Experimental framework

2.1 The Game

We explore the properties of emergent communication in the context of referential games where neural network agents, Speaker and Listener, have to cooperatively communicate in order to win the game.

Speaker network receives an input $i \in \mathcal{I}$ and generates a message m of maximum length `max_len`. The symbols of the message belong to a vocabulary $V = \{s_1, s_2, ..., s_{\texttt{voc\_size}-1}, \texttt{EOS}\}$ of size `voc_size` where `EOS` is the 'end of sentence' token indicating the end of Speaker's message. Listener network receives and consumes the message m. Based on this message, it outputs $\hat{i}$. The two agents are successful if Listener manages to guess the right input (i.e., $\hat{i} = i$).

In order to have a setting comparable to natural languages, inputs are drawn from $\mathcal{I}$ following a power-law distribution (Zipf, 2013), where $\mathcal{I}$ is composed of 100 one-hot vectors. Consequently, the probability of sampling the k^{th} most frequent input is: $\frac{1/k}{\sum_{j=1}^{100} 1/j}$. Also, experiments are done with `max_len` $= 30$ and `voc_size` $= 40$.

An emergent language consists of the input-message mapping. Each input $i \in \mathcal{I}$ fed to Speaker after successful communication is associated to its output message m. By $\mathcal{M}$, we define the set of messages m used by our agents after succeeding in the game.

2.2 Agents: Architecture and Loss

In our experiments, we compare different combinations of communicative agents:

- Standard agents: as a baseline, composed of Standard Speaker and Standard Listener;
- LazImpa agents composed of *Laz*y Speaker and *Impa*tient Listener.
- IntEmo agents composed of*Emotional* speaker and *Interrupting* listener;

2.2.1 Agents: Architectures

Standard Speaker. Standard Speaker is a single-layer LSTM which transforms the input i into a message m which is generated symbol by symbol with a downstream linear layer followed by a softmax to a categorical distribution over the vocabulary.

Standard Listener. Standard Listener is also a single-layer LSTM, which consumes the message m generated by the Speaker symbol by symbol until the `EOS` token is seen. The final hidden state is mapped to a categorical distribution $L(m)$ over the input indices

Lazy Speaker. Lazy Speaker has the same architecture as Standard Speaker. The 'Laziness' comes from a cost on the length of the message m directly applied to the loss.

Impatient Listener. Impatient Listener consists of a modified Standard Listener that, instead of guessing i after consuming the entire message $m = (m_0, ..., m_t)$, makes a prediction $\hat{i}_k$ for each symbol m_k from a categorical distribution $L(m_{:k})$ (Figure 1).

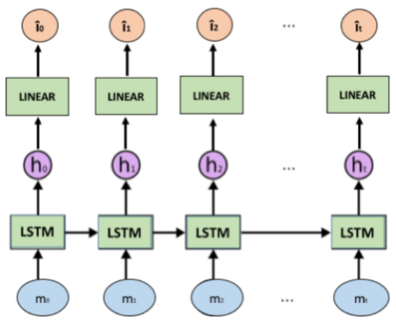

Figure 1: Impatient Listener architecture. The agent is composed of a single-layer LSTM cell and one shared linear layer followed by a softmax. It generates a prediction at each time step.

Interrupting Listener. Interrupting Listener uses the same intuition from the impatient listener, but rather than guessing $\hat{i}_k$ for each symbol m_k, the interrupting listener guesses $(\hat{i}_k, b_k)$, where b_k is binary (0/1): the first time $b_k = 1$, the listener 'interrupts' the speaker and endures a loss $\mathcal{L}_{std}(i, L(m_k))$ (Figure 2).
If all $b_k = 0$, we force $b_{EOS} = 1$.

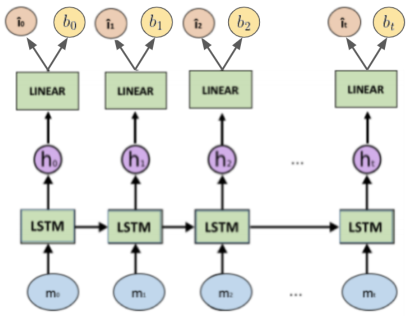

Figure 2: Interrupting Listener architecture. The agent is composed of a single-layer LSTM cell and two shared linear layers followed by a softmax. It generates a prediction and an interrupting signal at each time step.

Emotional Speaker. The emotional speaker has the same architecture as the standard speaker, but does not want to be interrupted. He wants the listener to consume all his message. This is achieved by adding a cost on the difference between the length of the message produced by the speaker and the length of the message actually consumed by the listener.

2.2.2 Agents: Losses

Standard loss $\mathcal{L}_{std}$. For Standard agents, we merely use the cross-entropy loss between the ground truth one-hot vector i and the output Categorical distribution of Listener $L(m)$.

LazImpa Loss $\mathcal{L}_{laz}$. LazImpa loss is composed of two parts that model 'Impatience' ($\mathcal{L}_{laz/L}$) and 'Laziness' ($\mathcal{L}_{laz/S}$), such that,

$$\mathcal{L}_{laz}(i, m, L(m)) = \mathcal{L}_{laz/L}(i, L(m)) + \mathcal{L}_{laz/S}(m). \quad (1)$$

On one hand, $\mathcal{L}_{laz/L}$ forces Impatient Listener to guess the right candidate as soon as possible when reading the message m and is defined as the mean cross-entropy loss between i and the intermediate distributions:

$$\mathcal{L}_{laz/L}(i, L(m)) = \frac{1}{t+1} \sum_{k=0}^{t} \mathcal{L}_{std}(i, L(m_{:k})), \quad (2)$$

On the other hand, $\mathcal{L}_{laz/S}$ consists of a penalty on message lengths. It can be defined with a fixed parameter α:

$$\mathcal{L}_{laz/S}(m) = \alpha |m| \quad (3)$$

or we can devise an adaptive version that consists of an exploration step during which the system explores long and discriminating messages and then, once it reaches good enough communication performance, a reduction step with a length cost penalty.

$$\mathcal{L}_{laz/S}(m) = \alpha(\text{acc})|m| \quad (4)$$

Such a schedule is shown in Figure 3.

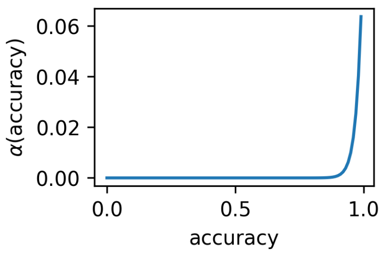

Figure 3: Scheduling of the regularization parameter α as a function of the accuracy. We distinguish two different regimes: the exploration and the reduction regimes.

Interrupting / Emotional loss $\mathcal{L}_{IntEmo}$. InterEmo loss is composed of two parts that model the 'interrupting' ($\mathcal{L}_{IntEmo/I}$)and 'not wanting to be interrupted' parts ($\mathcal{L}_{IntEmo/E}$) :

$$\begin{aligned}\mathcal{L}_{IntEmo}(i, m, L(m)) = \mathcal{L}_{IntEmo/I}(i, L(m)) \\ + \mathcal{L}_{IntEmo/E}(m, m_c).\end{aligned}$$

where $m_c = m_{consumed} = (m_1, ..., m_k)$ (such that k is the first time $b_k = 1$; remember b_{EOS} is always equal to 1) is the message consumed by the listener

The Interupting listener decides after each character consumed from the message whether or not to interrupt, and when he does ($b_k = 1$), he endures the loss:

$$\mathcal{L}_{IntEmo/I}(i, L(m)) = \mathcal{L}_{std}(i, L(m_k)) \qquad (5)$$

On the other hand, the emotional speaker does not want to be interrupted, if the listener decides to interrupt at k ($b_k = 1$), then the speaker has a loss:

$$\mathcal{L}_{IntEmo/E}(m, m_c) = \alpha(acc) * (|m| - |m_c|)^2 \qquad (6)$$

If the speaker does not want to be interrupted, one good way to achieve that is to produce concise messages.

2.3 Training

As messages are discrete-valued, the architecture is not directly differentiable. Hence, we use a hybrid optimization between REINFORCE for Speaker (Williams, 1992) and classic back-propagation for Listener (Schulman et al., 2015): in particular, we use a surrogate function whose gradient is as an unbiased estimate of the training task gradient (further details can be found in Rita et al., 2020).

3 Experiments

3.1 Experiment 1: Standard Speaker/ Standard Listener

As a first experiment, we intend to reproduce results of Chaabouni et al. (2020) where Standard Agents with no pressure towards brevity naturally develop anti-efficient encoding scheme. This can be seen in figure 4 where the emergent language is clearly not satisfying ZLA.

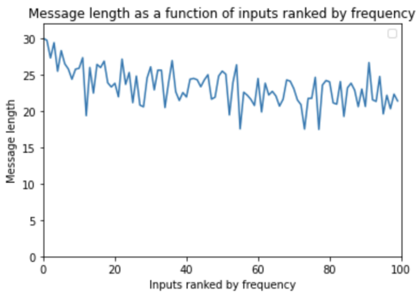

Figure 4: Messages length of Standard Agents

3.2 Experiment 2: LazImpa agent

Chaabouni et al. thus proposed to add a regularization term to the Speaker loss so as to imitate functional pressures in natural language toward effort minimization by penalizing long messages. It has been shown then that it does lead to ZLA-respecting emergent codes, but this approach can be taken further by adding a regularization term to the listener too as we've seen before. This results in a more efficient emerging code as displayed in figure 5.

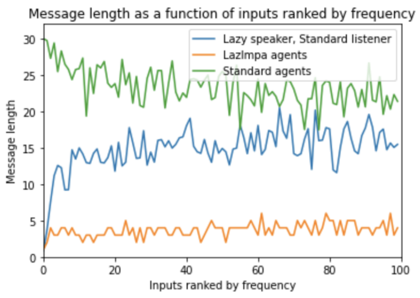

Figure 5: Messages length of Standard Agents versus Lazy speaker versus LazImpa

Let's recall the definition of the Impatient listener loss function: $\mathcal{L}_{laz/L}(i, L(m)) = \frac{1}{t+1}\sum_{k=0}^{t} \mathcal{L}_{std}(i, L(m_{:k}))$. Instead of the constant categorical distributions weights, we can design increasing weights so that the listener gets even more penalized as he takes longer to guess the correct output: $\mathcal{L}_{laz/L}(i, L(m)) = \sum_{k=0}^{t} \lambda^t \mathcal{L}_{std}(i, L(m_{:k}))$ (up to some normalizing factor). Results are displayed in figure 6: the emergent code with this new loss still respects ZLA, but it's far less efficient than previous LazImpa. We may need then to calibrate more finely the weights schedule before concluding on the impact of this approach.

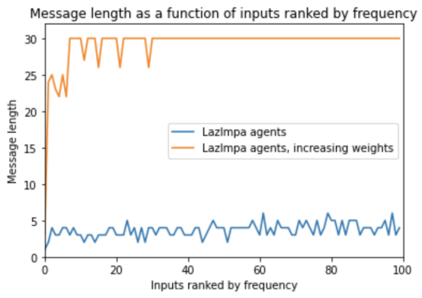

Figure 6: Messages length of LazImpa with increasing weights

3.3 Experiment 3: Comparisons

Finally, we wanted to perform comparisons between all possible agents combinations.
As expected (figure 7), combination of Lazy speaker and Impatient listener performs the best (and very close to natural languages). Surprisingly, Impatient listener alone with Standard speaker performs quite bad. It may be due to the fact that regularization of the listener alone is not sufficient to shrink messages length, compared to the most

explicit length cost of the speaker (or it may be due to unstable convergence, and further tests are required).

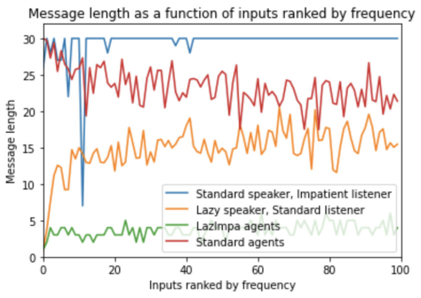

Figure 7: Messages length of different agents combinations

3.4 Experiment 4: IntEmo agent

For this experiment, we modified the code from LazImpa to have the IntEmo agent. To do so, we have first changed the receiver architecture, creating a "RnnReceiverInterrupting" class, by adding a second Linear Layer that outputs the binary interruption signal. The output now is a sequence of length $n\_features + 1$, with element at $n\_features + 1$ being a binary variable.
The second change is modifying the loss function. For the listener, we apply a mask to the output of the loss of the sequence to only retrieve the loss of the first time the listener interrupts. For the speaker, we retrieve from the mask the length of the message actually consumed to compute the emotional part of the loss. (To complete the implementation other modifications were also done like dump_interrupting and SenderInterruptingReceiverRnnReinforce to always take into consideration the interrupting signal.)

We first launched the training of IntEmo for 500 epochs with the loss as defined below and same parameters as Experiences 1,2 and 3. However, the agents didn't seem to learn, the loss was oscillating and the accuracy on the test dataset did not exceed 10%. After investigating our implementation, we discovered that the way we used to choose the first time the listener interrupted the speaker was blocking the flow of back-propagating gradients.

To overcome this implementation problem, we used a variation of the loss, where now we take the sum of all the losses where $b_k = 1$ (not only the first time):

$$\mathcal{L}_{IntEmo/I}(i, L(m)) = \sum_{k, b_k=1} \mathcal{L}_{std}(i, L(m_k)) \quad (7)$$

This can be seen as trying to interrupt the speaker many times while he is speaking.

After this modification, the agents started learning and the loss decreasing. However, and maybe due to some unlucky initialisation, the agents were stuck in a local minimum where the speaker was only producing messages of length 1, and the listener always interrupting after the first character. In this phase, the accuracy was around 0.2, since it's hard to code 100 digit inputs in a message of length 1. After around 200 epochs, the agents finally escape the local minimum and the accuracy increased drastically. However, after the end of the 500 epochs, the accuracy was still around 79%.
The experiment could not be considered successful. We believe that fine-tuning more the different hyper-parameters and training for more epochs would help the accuracy converge to 99%, like in the other experiments. Figure 8 shows the evolution of the length of inputs during learning. Since this was an unsuccessful run (0.79 accuracy) , we cannot fully analyse the results yet.

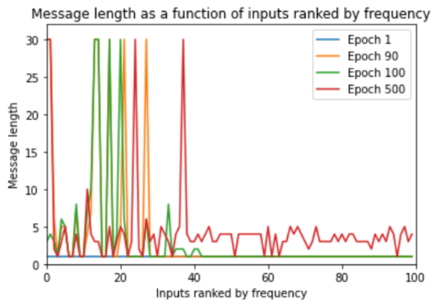

Figure 8: Messages length of the unsuccessful run of IntEmo

4 Conclusion and Future directions

In this project, we studied the effect of biases induced by architectures and loss functions in a simple communication scheme. We successfully reproduced the anti-efficient coding from standard agents, and ZLA-like distribution when using the LazImpa agents. We motivated a new agent 'IntEmo' with new loss function and architectures, but was unsuccessfully trained. More training epochs and hyperparameter tuning could have helped it converge. Another experiment for future directions that we would loved to try was to have the emotional speaker and interrupting listener co-evolve: at every step, the speaker generates a new character using the last character he generated but also the information on whether on not the listener interrupted or not. Many exciting properties could emerge from this scheme.

5 Group Work:

- Othmane: reproduced the experiments 1,2 and 3
- Achraf: designed the IntEmo architectures and

losses, implemented it using EGG in experiment 4

References

John Schulman, Nicolas Heess, Theophane Weber, and Pieter Abbeel. 2015. Gradient estimation using stochastic computation graphs. In *Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 2*, page 3528–3536, Cambridge, MA, USA. MIT Press.

Ronald J. Williams. 1992. Simple statistical gradientfollowing algorithms for connectionist reinforcement learning. *Machine Learning*, 8(3):229–256.

George Kingsley Zipf. 2013. *The psycho-biology of language: An introduction to dynamic philology.* Routledge.