Exploring Predictability Pressure in Referential Games for Language Emergence

Gerson Foks 12589845
University of Amsterdam
gerson.foks@gmail.com

Pieter Bouwman 11904488
University of Amsterdam
pieter.bouwman@student.uva.nl

Abstract

A referential game, in which two agents need to communicate do reach a common goal, is often used to study emergent languages. In this work we explore the effect of adding a predictor neural network next to the standard sender and receiver in a referential game. Making the training loss depend on the performance of this predictor imposes a pressure on the sender to generate messages that can be predicted well, which we expected to aid convergence and improve naturalness. Contrary to our expectations, applying this pressure does not make the emergent language more natural in terms of structuredness and instead hinders the convergence of the training.

1 Introduction

To study how languages evolve, we can run simulations of agents that need to communicate with each other and can develop their own language to do so. The languages that the agents develop to communicate with each other in these simulations, are referred to as emergent languages. The simulation studied in this work is called the Referential Game, in which a Receiver agent has to predict which pattern out of a number of input patterns is the pattern a Sender agent is seeing, purely based on a message received from the Sender (see 3 Approach). The emergent languages that evolve during these simulations rarely have the same properties as natural languages (e.g. [3], [6]).
The prominent challenge in the field of emergent languages is to make the emerged language more similar to natural languages, as this can give insight in how natural languages acquired their properties. To shape these languages to become more natural, pressures can be applied on how the agents can communicate during their development of the emergent language.

We propose a pressure that is intended to enforce structuredness of the emerged language through predictability of it, by training a Predictor neural network alongside the Sender and Receiver networks. We owe inspiration for this pressure from the predictive coding theory, that states that humans are continuously and subconsciously trying to predict what will happen next . We hypothesized that applying this Predictability Pressure (PP) will make the emerged language more natural in terms of structuredness and aids converge. Contrary to our hypothesis we found that:

1. There were no clear signs of improvement in syntactic structuredness when applying PP.
2. Performance on zero shot evaluation deteriorated when using PP, making it unlikely that PP increased the degree of compositionality of the language.
3. PP hindered convergence speed and stability.

Possible explanations for these discouraging results are in the game setup and the way the PP is implemented. We will elaborate on this the discussion. Furthermore we describe what experiments could be conducted to better understand this pressure.
The code used for our experiments is publically available [1]

2 Related Work

In the work of [3] the authors describe that natural language does not emerge naturally when agents are being trained to cooperate and communicate. They show that the training setup needs to be carefully designed to allow for languages to emerge that are more natural. [4] expands on this work by

[1] https://github.com/gersonfoks/NLP2EmergentLanguage

introducing different pressures that are designed to aid naturalness. They explore realistic pressures that are based on the principle of least effort and object constancy.
Other work alter the standard referential game setup to incorporate ideas that are based on the theory of mind. For example, [5] uses the idea of empathy to let the sender predict how the receiver is going to react on a given messages, which they show aids convergence speed when used in pair with the *REINFORCE* algorithm.
The inspiration of this paper came from an idea related to the theory of mind, namely the predictive coding theory (PCT). PCT states that *"... the brain is constantly involved in prediction-error minimization, that is, in minimizing the mismatch between internally-generated, top–down sensory signals and bottom–up sensory signals caused by the external environment."* [1]. Or, in other words, the brain is constantly trying to predict what comes next. To implement the idea of predictive coding theory, we add a Predictor module that learns to predict what the next symbol of the Sender will be for every timestep.

3 Approach

The simulation we study is a standard referential game, with a Sender, a Receiver, and when PP is active, a Predictor. The Sender gets to see an input pattern corresponding to a class. There are as many input patterns as classes, so there are no invariants to learn. Each input pattern is encoded as a concatenation of n one hot encodings of length m, representing n attributes that each can take on m different values. The Receiver sees three of those different classes of which one is the same as the Sender is seeing. The Receiver needs to predict which input pattern the Sender is seeing, from the message it sends. The Sender sends discrete messages which are chosen with the help of the gumbel-softmax trick [2].

In case that PP is active, a Predictor agent is added to the game that needs to predict each symbol that the Sender sends. The pressure on the system comes from the loss of the Predictor being added to the total loss as a weighted term. This means that optimizing the loss, involves the Sender to become more predictable for the Predictor.
The Predictor is implemented as a module that is separate from the Sender and Receiver. A

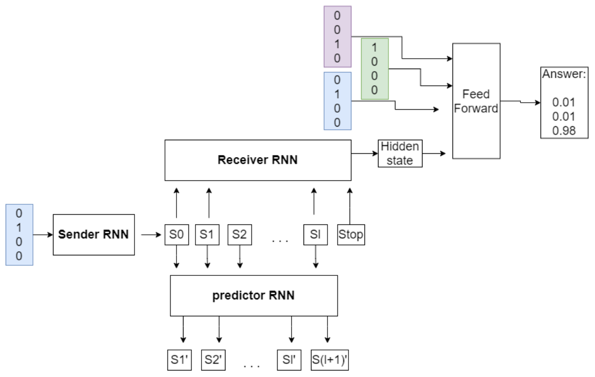

Figure 1: The setup

schematic overview is given in figure 1

Models A one layer Feed Forward Neural Network (FFNN) is used as a feature encoder of the input patterns. It feeds the Sender agent that is implemented as an LSTM. The Receiver is also an LSTM and gets as inputs the message generated by the Receiver, and the output of the FFNN feature encoder applied on the input patterns the Receiver has to choose from. The Predictor is also implemented as an separate LSTM and gets as input the message generated by the Sender and outputs a predicted message.

Loss function The loss function is a sum of two losses. The first term is the cross entropy between the output logits of the Receiver over the input patterns it can choose from and the correct index. The second term is the cross entropy between the message generated by the Sender, and the output logits predicted by the Predictor. The influence of the second term is modulated by a predictor loss weight, that is 0 when PP is not active.

4 Experiments & Hypotheses

To investigate whether applying PP results in emerged languages that are more structured, we conduct a series of experiments and analyse metrics that are indicative of structuredness. We consider two types of structure: syntactic structure, more word order structure, and semantic structure, as characterised by compositionality.

Metrics For measuring word order structure (or more general: syntactic structure) we use N-gram entropy for N=1 and N=2. For the degree of compositionality we use zero shot evaluation performance measured as accuracy on a held out dataset that is guaranteed to be compositionally constructable from the training set.

Experiments For all experiments the maximum number of symbols per message is set to 10, and the vocabulary size is 25. All three agents use an LSTM with a hidden dimension of 128. The FFNN feature encoder uses a hidden layer of size 128 and is pretrained until convergence by having the layer be part of a classifier that needs to classify the one-hot encodings to the corresponding class. The Adam optimizer is used for optimization with a learning rate of 0.0001.
The first experiment involves training a Sender and Receiver for establishing a baseline. As training data, the dataset described earlier is used with $n = 3$ attributes and $m = 4$ values each. The second experiment is identical to the first, apart from that now PP is active. In the third experiment, both experiments are conducted again, but for $n = 5$ and $m = 5$. For all experiments a batch size of 128 is used and the Receiver has to pick a pattern from 3 input patterns. All experiments where PP is active use are conducted for a predictor-loss-weight of 0.01 (PP ON-HIGH) and 0.0001 (PP ON-LOW).

Hypotheses If our proposed pressure has the expected influence on the language, this should be reflected in lower N-gram entropy, and higher performance on the held out set than a baseline that does not apply PP. If results indicate PP has the expected effects on the language, metrics such as message length, average number of distinct symbols per message will be inspected and qualitative evaluation will be performed, to ensure the perceived increase in structuredness is not due to shortcuts the models have found or pathological behavior such as only using a single symbol.

5 Results

Results do not confirm our hypotheses (see Figure 2 and Figure 3). While seeing no clear improvements when PP is active (either LOW or HIGH), we even see increased instability in some cases. This instability can for example be seen in the validation accuracy for HIGH PP, where only 1 out of 5 seeds did not perform merely on chance level (33% accuracy). For the experiments with more attributes and values ($n = 5$, $m = 5$), instability seemed to increase further as none of the 5 trained models achieved better performance than chance.
For zero shot evaluation on the held out set (see Figure 3), performance decreases as the pressure increases, and again high variability is visible in the PP ON experiments. What is notable is that the average message length is equal to the number of attributes in the data, 3. While this seems to point at a one-on-one relation between a used symbol and an attribute, the experiments with 5 attributes also had an average message length of 3. This shows that even if it would be the case that when $n = 3$, each symbol corresponds to an attribute, this does not hold in general.
In all experiments the Predictor reached an accuracy of around 0.15 or worse. As we will discuss later, this is performance poor, although it is hard to determine how poor it exactly is.

6 Conclusion & Discussion

We conclude that the proposed Predictability Pressure, for the experiments conducted in this research, did not improve naturalness in terms of word order structure, and in terms of generalization to a held out set that could eventually indicate compositionality, performance is even hurt. Furthermore applying the Predictability Pressure caused instability that is harmful for convergence stability. There are several possible reasons for why our results are not as expected. First of all, when using the Predictor, the game that is played becomes more complex for the Sender. This is because it not only needs to send an informative message but now also a message that is easily predictable for the Predictor. This increased complexity could explain why it is harder to find an optimum and thus explain why converge stability is hindered.

We therefore suspect that we may need a different training setup to make the training more stable and get better results with this pressure.

For example, parameter sharing between the receiver and the predictor could help to stabilize the training. Using parameter sharing, the Receiver gets more information, as its gradient signal is enriched by the gradient carrying information about predictability. We could also introduce a schedule for the weight of the pressure, in which we gradually increase the weight of the predictor, this could have the effect that the sender and receiver first solve the referential game and then improve their language further to be more easy predictable.

During our experiments we found that the accu-

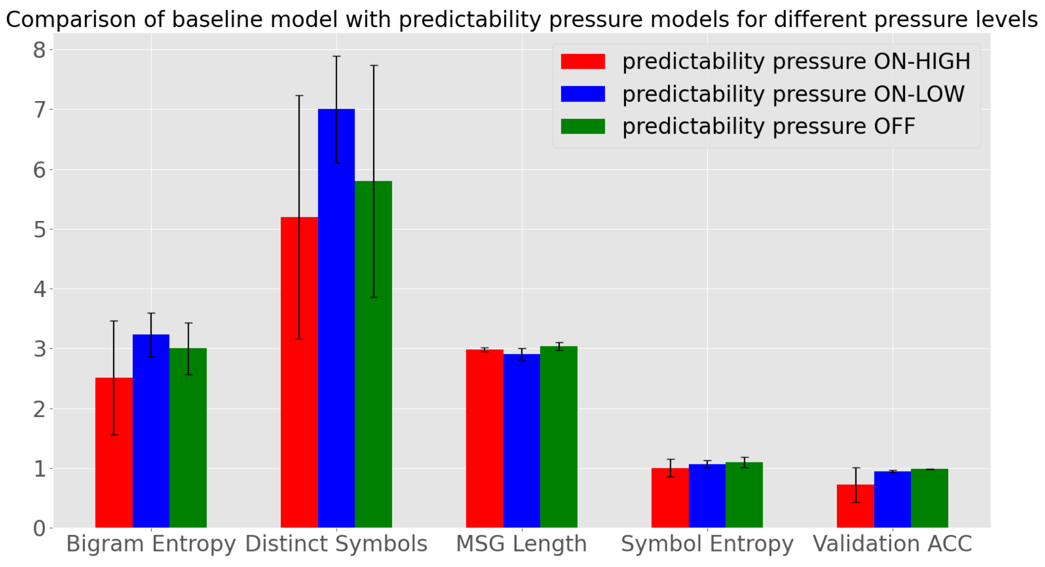

Figure 2: Comparing baseline model to predictability pressure models. For HIGH PP the predictor loss weight is 0.01, for LOW a weight of 0.0001. Results are averaged over 5 seeds.

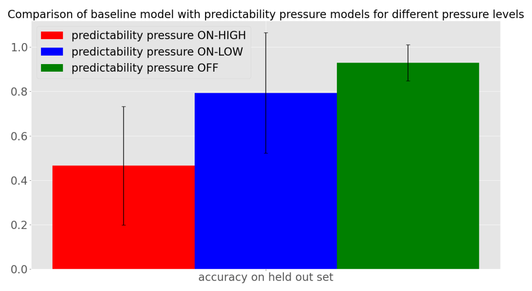

Figure 3: Zero shot evaluation performance for baseline and PP models. For HIGH PP the predictor loss weight is 0.01, for LOW a weight of 0.0001. Results averaged over 5 seeds.

racy of the predictor was rather poor. This could be due to the fact that the predicting task is inherit more difficult than it is for a sender and receiver to learn to communicate as the language keeps on evolving while the sender and receiver are improving. The predictor is therefore trained on a 'moving target', making it hard to train the predictor. Again, parameter sharing could be a solution to this as then the gradients of the predictor and receiver are aligned.

Lastly, we note that it is hard to determine what a good accuracy for the predictor is. For example if we have a fully compositional language for n=3, m=4 in which each symbol represents one attribute value we have that the best accuracy the predictor can have is $\frac{1}{m} = \frac{1}{4}$ as the predictor has $\frac{1}{4}$ chance of predicting what the value of the next attribute is. But when the language is not compositional it is hard to determine what a good accuracy is, as we do not know the probabilities of observing each subsequent symbol. This should be taken into account when analysing the performance of the predictor.

6.1 Future work

The study of emergent languages is a relatively new field and therefore lacks standards and benchmarks. During our work we found out that there are a lot of practical challenges when doing research into emerging languages, such as unstable training, the receiver remembering the training set and hard to interpret results. We suggest that work on creating benchmarks, tools to analyse emergent languages and tools to test and debug models is really valuable. A good start of this effort is [2].
Additionally, during our analyses we found that the game played may be too easy for the baseline as the baseline already finds a very good optimum with respect to the measurements with very little room for improvement. Thus once the point of having stable training is reached, it would be interesting to see how the pressure influences languages that emerge during more challenging tasks as this could result in a more clear differences between the baseline and the proposed methods.

[1] Paweł Gładziejewski. “Predictive coding and representationalism”. In: *Synthese* 193.2 (2016), pp. 559–582.

[2] Eugene Kharitonov et al. “EGG: a toolkit for research on Emergence of lanGuage in Games”. In: *arXiv preprint arXiv:1907.00852* (2019).

[3] Satwik Kottur et al. “Natural language does not emerge’naturally’in multi-agent dialog”. In: *arXiv preprint arXiv:1706.08502* (2017).

[4] Diana Rodrıguez Luna et al. “Internal and external pressures on language emergence: Least effort, object constancy and frequency”. In: *arXiv preprint arXiv:2004.03868* (2020).

[5] Marie Ossenkopf. “Enhancing Language Emergence through Empathy”. In: (2019).

[6] Oskar van der Wal et al. “The Grammar of Emergent Languages”. In: *arXiv preprint arXiv:2010.02069* (2020).