EMERGENCE OF LANGUAGE WITH MULTI-AGENT GAMES: LEARNING TO COMMUNICATE WITH SEQUENCES OF SYMBOLS

Serhii Havrylov, Ivan Titov
Institute for Logic, Language and Computation
University of Amsterdam
{s.havrylov,titov}@uva.nl

ABSTRACT

Learning to communicate through interaction, rather than relying on explicit supervision, is often considered a prerequisite for developing a general AI. We study a setting where two agents engage in playing a referential game and, from scratch, develop a communication protocol necessary to succeed in this game. We require that messages they exchange, both at train and test time, are in the form of a language (i.e. sequences of discrete symbols). As the ultimate goal is to ensure that communication is accomplished in natural language, we perform preliminary experiments where we inject prior information about natural language into our model and study properties of the resulting protocol.

1 INTRODUCTION

With the rapid advances in machine learning in recent years, the goal of enabling intelligent agents to communicate with each other and with humans is turning from a hot topic of philosophical debates into a practical engineering problem. It is believed that supervised learning alone is not going to provide a solution to this challenge (Mikolov et al., 2015). These, as well as other, considerations have motivated previous research into set-ups where agents invent a communication protocol which lets them succeed in a given collaborative task. For an extensive overview of earlier work in this area, we refer the reader to Kirby (2002). We continue this line of research and specifically consider a setting where the collaborative task is a game. Neural network models have been shown to be able to successfully induce a communication protocol for this setting (Lazaridou et al., 2016; Jorge et al., 2016; Foerster et al., 2016; Sukhbaatar et al., 2016). One important difference with these previous approaches is that we assume that messages exchanged between the agents are variable-length sequences of symbols rather than atomic categories (as in previous work). In our experiments, we focus on a *referential* game. The setting of the game is slightly different from that of Lazaridou et al. (2016). Its description can be found in the appendix 5.1.

2 MODEL

2.1 AGENTS' ARCHITECTURES

The sender and the receiver are implemented as LSTM networks (Hochreiter & Schmidhuber, 1997). Figure 1 shows the sketch of model architecture, where diamond-shaped, dashed and solid arrows represent sampling, copying and deterministic functions respectively. The inputs to the sender are the target image t and the special token <S>, which denotes the start of a message. Given these inputs, the sender generates next token w_i in a sequence by sampling from the categorical distribution $\texttt{Cat}(p_i^t)$, where $p_i^t = \texttt{softmax}(Wh_i^s + b)$. Here, h_i^s is the hidden state of sender's LSTM and can be calculated as $h_i^s = \texttt{LSTM}(h_{i-1}^s, w_{i-1})$[1]. In the first time step we have $h_0^s = f(t)$, where $f(t)$ is an affine transformation of image features extracted from a convolutional neural network (CNN). Message m_t is obtained by sequentially sampling until the maximum possible length L is reached or the special token <S> is generated.

[1]We omitted the cell state in the equation for brevity.

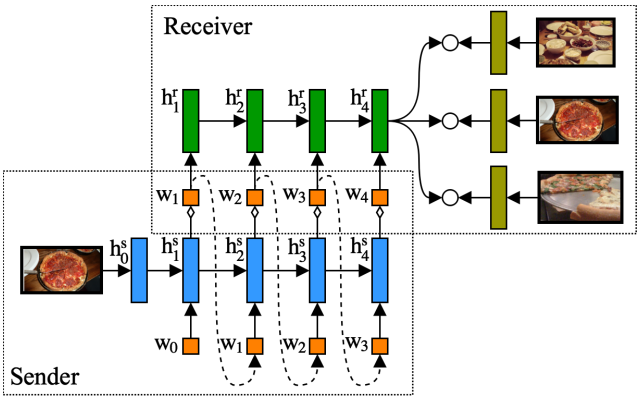

Figure 1: Architectures of sender and receiver

The inputs to the receiver are the generated message m_t and a set of images that contain the target image t and distracting images $\{d_k\}_{k=1}^K$. Receiver interpretation of message is represented by the last hidden state of the LSTM network h_l^r. The loss function for the whole system can be written as:

$$\mathcal{L}_{\phi,\theta}(t) = \sum_{k=1}^{K} \max[0, 1 + g(t)^T h_l^r - g(d_k)^T h_l^r] \tag{1}$$

Here $g(\cdot)$ is yet another affine transformation of image features extracted from the CNN. The energy function $g(\cdot)^T h_l^r$ can be used to define the probability distribution over a set of images. Communication between two agents is successful if the target image has the highest probability according to this distribution.

2.2 Straight-through Gumbel-softmax estimator

Generating message m_t requires sampling from categorical distributions over vocabulary, which makes backpropagating the error through the message impossible. It is tempting to formulate this game as a reinforcement learning problem. However, the number of possible messages[2] is proportional to $|V|^L$. Therefore, it seems unlikely that such model can be learned with naïve Monte Carlo methods. Also, in this setup, the receiver R_θ would correspond to the non-stationary environment in which sender S_ϕ acts, making the learning problem even more challenging. As a solution, we consider replacement of one-hot encoded symbols $w \in V$ with continuous relaxation $\tilde{w}$ obtained from the Gumbel-softmax distribution (Jang et al., 2016; Maddison et al., 2016). As a result of this relaxation the game becomes completely differentiable and can be trained using the backpropagation algorithm. Communicating with real values allows the sender to encode much more information into a message comparing to using a discrete one and is unrealistic if our ultimate goal is communication in natural language. To prevent this behaviour, we discretize $\tilde{w}$ back with $\arg\max$ in the forward pass. Nevertheless, we use continuous relaxation in the backward pass, effectively assuming $\frac{\partial L}{\partial w} \approx \frac{\partial L}{\partial \tilde{w}}$. This biased estimator is known as the straight-through Gumbel-softmax estimator (Jang et al., 2016; Bengio et al., 2013). As a result of applying this trick, there is no difference in using messages during training and testing stages, which contrasts with previous differentiable frameworks for learning communication protocols (Foerster et al., 2016; Sukhbaatar et al., 2016).

3 Experiments

3.1 Tabula rasa communication

We used the Microsoft COCO dataset (Chen et al., 2015) as a source of images. The MSCOCO 2014 validation set was used for evaluating the learned language. In our experiments images are represented by outputs of the `relu7` layer from the pretrained 16-layer VGG convolutional network (Simonyan & Zisserman, 2014).

[2]In our experiments $|V| = 10000$ and L is up to 14.

Table 1: Comparison of grounded protocol with natural language and artificial language.

Model	Comm. success (%)	Number of updates	Omission score
With KL regularization	52.51	11600	0.258
Without regularization	95.65	27600	0.193
Imaginet	52.51	16100	0.287

The model details can be found in appendix 5.2. Figure 2 shows the communication success rate as a function of the maximum message length L. Interestingly, the number of updates that are required to achieve training convergence decreases when we let the sender use longer messages. In other words, using longer sequences helps learn a communication protocol faster. This behaviour is slightly surprising as one could expect that it is harder to learn the protocol when the space of messages is larger. We also plot the perplexity of the encoder, it is relatively high and increasing with sentence length. This implies redundancy in the encodings: there exist multiple paraphrases that encode the same semantic content. For additional qualitative analysis of the learned language we refer reader to appendix 5.3.

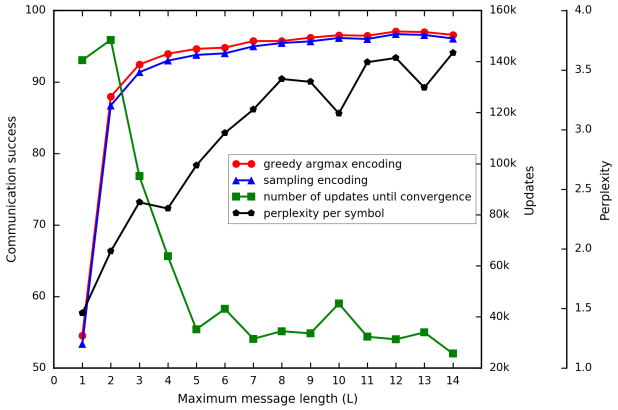

Figure 2: The performance and properties of learned protocols.

3.2 Grounding artificial language in natural language

As the ultimate goal is to ensure that communication is accomplished with a language that is understandable by humans, we should favour protocols which resemble, in some respects, a natural language. One possible solution is to use the Kullback-Leibler (KL) divergence regularization $D_{KL}\left(q_\phi(m|t)\|p_\omega(m)\right)$, from the estimated natural language model to the learned protocol. Details regarding language model could be found in appendix 5.2. This regularization provides indirect supervision by encouraging generated messages to have a high probability in natural language but at the same time maintaining high entropy for the communication protocol.

To get an estimate of communication success when using natural language we trained receiver with pairs of images and captions. This model is similar to Imaginet (Chrupała et al., 2015). Also, inspired by their analysis, we report the *omission score*. The score quantifies the change in the target image probability after removing the most important word in a sentence. Natural languages have content words that name objects and encode their qualities. One can expect that a protocol that distinguishes between content words and function words would have a higher omission score in comparison to a protocol that distributes information evenly across tokens. As Table 1 shows, the grounded language has the communication success rate similar to natural language. However, it has a slightly lower omission score. The model without regularization has the lowest omission score which probably means that symbols in the developed protocol have similar nature to characters or syllables rather than words.

4 Conclusion

We have shown that agents, modeled using neural networks, can successfully invent a language that consists of sequences of discrete tokens. Despite the common belief that it is hard to train such models, we proposed an efficient learning strategy which relies on the straight-through Gumbel-softmax estimator. We have performed basic analysis of the learned language and learning dynamics. We have also conducted preliminary experiments with the grounding learned language in natural language. Many open questions remain, so further investigation of this topic is needed.

ACKNOWLEDGMENTS

This project is supported by SAP ICN and ERC Starting Grant BroadSem (678254).

REFERENCES

Yoshua Bengio, Nicholas Léonard, and Aaron Courville. Estimating or propagating gradients through stochastic neurons for conditional computation. *arXiv preprint arXiv:1308.3432*, 2013.

Xinlei Chen, Hao Fang, Tsung-Yi Lin, Ramakrishna Vedantam, Saurabh Gupta, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco captions: Data collection and evaluation server. *arXiv preprint arXiv:1504.00325*, 2015.

Grzegorz Chrupała, Akos Kádár, and Afra Alishahi. Learning language through pictures. *arXiv preprint arXiv:1506.03694*, 2015.

Jakob Foerster, Yannis M Assael, Nando de Freitas, and Shimon Whiteson. Learning to communicate with deep multi-agent reinforcement learning. In *Advances in Neural Information Processing Systems*, pp. 2137–2145, 2016.

Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8): 1735–1780, 1997.

Eric Jang, Shixiang Gu, and Ben Poole. Categorical reparameterization with gumbel-softmax. *arXiv preprint arXiv:1611.01144*, 2016.

Emilio Jorge, Mikael Kågebäck, and Emil Gustavsson. Learning to play guess who? and inventing a grounded language as a consequence. *arXiv preprint arXiv:1611.03218*, 2016.

Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Simon Kirby. Natural language from artificial life. *Arificial Life*, 8:185–215, 2002.

Angeliki Lazaridou, Alexander Peysakhovich, and Marco Baroni. Multi-agent cooperation and the emergence of (natural) language. *arXiv preprint arXiv:1612.07182*, 2016.

Chris J Maddison, Andriy Mnih, and Yee Whye Teh. The concrete distribution: A continuous relaxation of discrete random variables. *arXiv preprint arXiv:1611.00712*, 2016.

Tomas Mikolov, Armand Joulin, and Marco Baroni. A roadmap towards machine intelligence. *arXiv preprint arXiv:1511.08130*, 2015.

Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

Sainbayar Sukhbaatar, Rob Fergus, et al. Learning multiagent communication with backpropagation. In *Advances in Neural Information Processing Systems*, pp. 2244–2252, 2016.

5 APPENDIX

5.1 REFERENTIAL GAME DESCRIPTION

1. There is a collection of images $\{i_n\}_{n=1}^{N}$ from which target image t is sampled as well as K distracting images $\{d_k\}_{k=1}^{K}$.
2. There are two agents: sender S_ϕ and receiver R_θ.
3. After seeing target image t, the sender has to come up with a message m_t, which is represented by a sequence of symbols from the vocabulary V of a size $|V|$. The maximum possible length of a sequence is L.
4. Given a message m_t and a set of images, which consists of distracting images and a target image, the goal of the receiver is to identify the target image correctly.

5.2 MODEL SPECIFICATION

We set the following hyperparameters without tuning: embedding dimensionality is 256, the dimensionality of LSTM layer is 512, vocabulary size is 10000, Gumbel-softmax distribution temperature is 1.0, the number of distracting images is 127, batch size is 128. We used Adam (Kingma & Ba, 2014) as an optimizer, with default hyperparameters and a learning rate of 0.001. We implemented language model $p_\omega(m)$ by using an LSTM recurrent neural network. Image captions of randomly selected (50%) images from the training set were used to estimate parameters of the language model. These images were not used for training the sender and the receiver.

5.3 QUALITATIVE ANALYSIS OF THE LEARNED LANGUAGE

To better understand the nature of the learned language, we inspected a small subset of sentences that were produced by the model with maximum possible message length equal to 5. Figure 3 shows some samples from the MSCOCO 2014 validation set that correspond to `(5747 * * * *)` code[3]. Images in this subset depict mainly animals. On the other hand, it seems that samples in figure 4 do not correspond to any predefined category. This suggests that word order is crucial in the developed language. Particularly, word `5747` on the first position encodes presence of an animal in the image. Considering Figure 5, we can conclude that adding word `5490` on the second position reduces possible options just to zebras, giraffes and sometimes horses. When we move token `5490` to the end of the message, we end up just with zebras in the images (Figure 6). Figure 7 shows that message `(5747 5747 7125 * *)` corresponds to a particular type of bears. This suggests that developed language implements some kind of hierarchical coding. This is interesting by itself because the model was not constrained explicitly to use any hierarchical encoding scheme. Figures 8, 9, 10 show similar behaviour for images in a food domain.

Figure 3: Images that correspond to `(5747 * * * *)` code.

Figure 4: Images that correspond to `(* * * 5747 *)` code.

Figure 5: Images that correspond to `(5747 5490 * * *)` code.

[3] * means any word from the vocabulary.

Figure 6: Images that correspond to `(5747 * * * 5490)` code.

Figure 7: Images that correspond to `(5747 5747 7125 * *)` code.

Figure 8: Images that correspond to `(5261 * * * *)` code.

Figure 9: Images that correspond to `(5261 2250 * * *)` code.

Figure 10: Images that correspond to `(5261 2250 5211 * *)` code.