Language Games

Mac O'Brien

September 20, 2022

Abstract

In this paper I introduce an approach to the computational study of language premised on the idea that language arises in the service of particular ends, and that the meaning of its constituent parts is always ultimately cached out in terms of behavior. To this end I train simple models on tasks that require the development of systems of communication and review their performance. I find that straightforward systems of communication can easily be learned through gradient descent, but that this simple neural network based approach is not sufficient to develop the more complicated modes of communication required for more difficult tasks. I also find that pre-training a model on one communication task can be useful when performing another.

1 Introduction: Meaning and Language

When we study natural language, the goal is usually to extract some sort of semantic meaning from a sequence of tokens. But what is semantic meaning in the first place? What are we saying when we say that a word or a phrase means something? One answer would be to say that a word has meaning insofar as it *stands for* or *corresponds to* an object in the world—but corresponds how? And what about words that don't seem to stand for any object at all?

One way of approaching this question draws from the notion of "language games" invoked by the philosopher Wittgenstein in his later writings. He uses the phrase to push away from an understanding of meaning as a strict correspondence between words and their objects, and towards a model of language as an activity interwoven with the various other activities of life, whose meaning arises from the "rules of the game" in which it is used.

From this perspective, the only way to fully understand the meaning of a word is to place it within the full context of the practical activities that language plays a part in. Since the expression language itself is an activity, much of the meaning of natural language can be captured by pure textual analysis, but to the extent that it excludes all other modes of activity, it likewise excludes part of the context that gives words meaning.

In what follows I've attempted to capture in a model the whole meaning of a language (though not any of ours), by treating it as a shared system of communication between agents, developed in pursuit of the particular ends placed on them by the rules of the game.

2 Methods

Each task was chosen to be theoretically solvable only if information was allowed to pass between agents. This means that a reliable solution to the task relies on information spread between the two agents, so that neither has direct access to enough data to solve the task on its own. This forces the models to learn systems of communication in order to succeed.

In order for the communication between agents to be learned fully by the model, the meaning intrinsic to the signals magnitude had to be stripped away as much as possible. To this end, the signals were forced into the form of "soft" one-hot embeddings, by multiplying the signal vector by a large constant (35) and then softmaxing it so that the largest element is pulled very close to one, and the rest towards zero. I couldn't fully reduce the signal to a one-hot embedding, because this would not be differentiable.

All models share a general architecture, with two linear layers for each agent, and the last element(s) of the output used as a signal to the other agent. In every model except the adversarial one, the communication process goes like this: Agent 1 gets its input and an empty communication space (zeros) and outputs its signal. Agent 2 takes in its input along with the signal just produced, and outputs its own signal alongside its final output. Agent 1 then takes its input again, this time with the signal from Agent 2, and produces its final output. Both outputs are then stacked and returned.

For the adversarial model, each agent instead outputs to a shared communication vector that is taken in as input by all agents at once. The iterative model differs from the others only insofar as it takes in the average signal up to the current time step.

As a control, the models' performances were compared to an analogous model with no communication allowed.

2.1 Target Matching

As a simple preliminary implementation, I chose the following simple task: pass in two one-hot encoded vectors (one to each agent), and output two numbers (one from each agent) that sum to one if both of the input vectors match any vector in a predetermined target list, and zero otherwise. The task satisfies the desired conditions, as neither agent can reliably know which output to choose without having information about the input received by the other agent. This task is also inherently cooperative, as the outputs are scored as a sum rather than individually. Theoretically, each agent should only need to pass one bit of information, signaling whether their own input matches or not.

2.2 Pair Matching

As a (slightly) more complicated task, I asked the model to return outputs that sum to one if the two one-hot encoded vectors passed to each agent matched each other, and zero otherwise.

2.3 Adversarial Match Checking

This task involved two pairs of agents with competing interests, all of whom contributed to a shared communication space. As in the other tasks, each agent gets a one-hot encoded input vector that only it has access to. The goal for one team was for the sum of outputs from every agent to equal one when any two vectors matched, and zero otherwise—the other team had precisely the opposite goal. The game is approximately fair for four agents when the size of the input vectors is 10, as there will be a match about 50 percent of the time, but the second team has a slight advantage. Each pair is optimized across both team members, with no access to the gradients involved in the other team's decision-making (allowing that sort of mind-reading converges to a draw every time.)

2.4 Iterative Counting

This was the only task to involve a temporal component. It went like this: at every time step, each of two agents was passed a one-hot encoded vector, and allowed to send and receive one signal. After the last time step, output from one agent was collected. The expected output was derived from the most common input vector across both agents in one of two ways: either as the direct one-hot representation of the vector, or as a binary classification of whether the vector appeared an even or odd number of times.

3 Results

The model was able to quickly perfect the target matching and pair matching tasks, with the minimum signal sizes theoretically possible (2 for the target matching, and equal to the input size for the pair matching.) As confirmed by the performance of the no-communication model, a learned system of communication was required to achieve these performances. These results are shown in Table 1.

On the iterative task, the model was able to perform 30 percent better than the non-communicative baseline when expected to output the most common vector, but, interestingly, performed worse than the baseline when asked to classify whether it appeared an even or odd number of times. A model trained first on the vector output, and then trained with another layer to perform the even/odd task was able to match but not surpass the model without communication. For the adversarial task, I compared models with

	Target Matching	Pair Matching	Iterative Counting (vector/classify)
No Signals	.058	.157	.800 / .252
Signals	0.0	0.0	.759 / .264
Improvement	100%	100%	30% / -4%

Table 1: The final losses of the trained models for each non-adversarial task with and without communication between agents. All loss values are calculated as the square of the difference between the actual and expected values. Since the vector version of the iterative task involves comparing more values, its loss will tend to be higher.

the same hyper-parameters, but varied the initial weights, either using a fresh initialization, or first training the model on the pair matching task. These results are shown in Table 2.

Because of the rules of the game, the second pair has a slight baseline advantage (seen in the no pre-training result,) but this advantage disappears when the first pair is allowed to train together beforehand. When the second pair is able to pre-train, but not the first, the gap widens even more.

	No Pre-training	Pre-trained A	Pre-trained B
loss_a	.290	.263	.378
loss_b	.270	.264	.250

Table 2: The final losses of the adversarial task for two models with the same hyper-parameters. Pair A has a slight advantage over pair B due to the nature of the task.

4 Conclusion

The results of the target matching and pair matching tasks show that gradient descent on simple neural networks is sufficient to develop systems of communication, so long as the task is relatively straightforward. With the addition of an iterative, temporal component, this method starts to fail, with the some (but not enough) useful information conveyed in the vector output version and none at all conveyed in the classification version. The classification task is probably more difficult because two sorts of information need to be conveyed at once: relating to which vector is most common, and to the parity of that vector. Learning just one of these forms of communication without the other does nothing for the final performance, so it is able to learn neither.

The adversarial task demonstrates that a pair of agents trained to already interpret one another's signals outperforms a pair without such training. It isn't clear from the experiment, however, if this pair learns to decode the signals of the other pair as they struggle to learn from scratch, or whether the advantage lies solely in the head start they get in developing a system of communication with each other.

Natural language is, of course, much more complicated than any of the systems of communication developed in the language games described above. But given the results, this shouldn't be surprising: as the complexity of the input and output spaces increased, so too did the sophistication of the system of communication needed to coordinate the appropriate activities. The practical context of natural language—the problems in response to which natural language asserts itself—is much broader and more varied than any simple game someone could come up with. The sequential task was perhaps closest, but even it didn't really require a grammar. Capturing the more complicated mechanisms of meaning therefore requires harder problems, and correspondingly sophisticated models. In the future, it would be interesting to see whether methods like self-attention, which have been extremely powerful tools for modeling language, could assist in the generation of new language.