Report: Emergent communication in a summation game

Abstract

In this project, I implement and analyze a language emergence game in which the sender neural network is given two integers as input, $s_1, s_2 \in \{0, \dots N\}$, and the receiver neural network must output their sum, $s_1 + s_2$. For example, if the input to the sender is $(3, 5)$, the receiver output should be 8. The agents communicate with single-symbol messages. I conduct simulations for different values of N and different vocab sizes. Given that the vocab size is large enough, the agents manage to play the game successfully for all tested values of N ($\{20, 40, 80\}$). However, they can only generalize to novel combinations of input integers if the input space is large ($N = 80$) and if the training set covers a large proportion of that input space. Interestingly, the agents use a communication strategy where the sender encodes information about the input *sum*, rather than the two input *summands*. Mappings from one symbol to multiple input sums (i.e. polysemy) exist but are rare. Mappings from one input sum to multiple symbols (i.e. synonymy) exist, too, and increase with the vocab size. I discuss these results and present some ideas on how to improve the setup and scale to larger input spaces.

1 Setup

Two agents, a sender and a receiver, play a *summation game*. One round of the game proceeds as follows:

1. Two randomly selected summands, $\{(s_1, s_2) \mid s_i \in \mathbb{N}_N\}$ with $\mathbb{N}_N = \{0, ..., N\}$, are passed to the sender.
2. The sender generates a single-symbol message $m \in V$, where $V = \{0, \dots, |V| - 1\}$ is the vocabulary.
3. The receiver generates a sum $S \in \mathbb{N}_{2N}$.

The input data consists of all possible combinations of input summands, $s_1 \times s_2$, given a fixed value for N. These $|\mathbb{N}_N|^2$ samples are randomly split into a training and a test set at a ratio of 0.9 / 0.1. To increase the training set size each sample is repeated 50 times.

2 Architecture and optimization

The sender and the receiver are implemented as multi-layer perceptrons (MLPs). The sender neural network processes the concatenated one-hot encodings of the input summands and generates a discrete message using Gumbel-Softmax (GS) (Maddison, Mnih, & Teh, 2017; Jang, Gu, & Poole, 2017). The message is embedded and passed to the receiver neural network, which generates a probability distribution over possible sums. Although the GS method estimates symbol probabilities through a continuous approximation during training, I evaluate training and test performance for discrete communication.

I conducted a grid search, fixing $N = 20$ and $|V| = 200$, to determine working hyperparameters in a small-scale setup (see Appendix A). Sender and receiver MLP each have two hidden layers of size 128, and the symbol embeddings are of size 64. The initial GS temperature is 2.0 and decays at a rate of 0.995 every epoch, up to a fixed minimum of 1.35. The agents are trained with cross-entropy loss using Adam optimizer, learning rate 0.001, and batch size 32.

3 Experiments

I conduct experiments for different input spaces and vocab sizes. Aside from $N = 20$, which was used in the grid search, I test $N = 40$ and $N = 80$. The minimal vocab size required to communicate all possible sums with distinct symbols is $|V|_{min} = |\mathbb{N}_{2N}|$. I vary the vocab size using different factors of this value, $|V|_{min} \cdot \{1, 2, 4, 8\}$. For example, the minimal vocab size for $N = 20$ is $|V|_{min} = 41$ and the vocab sizes used for training are $|V| \in \{41, 82, 164, 328\}$. Training proceeds for up to 250 epochs, or until a training accuracy of 0.99 is reached. In the interest of time and resources, I conduct only one run per combination of input size and vocab size but multiple runs would help substantiate the results.

4 Results

4.1 Training performance

Figure 1.a shows the agents' performance on the training set. The left plot shows the accuracy achieved at the end of training, and the right plot the mean absolute error (MAE) between the true and the predicted output sums. Plots of the training accuracy over time can be found in Appendix B. While the accuracies lie at 0.85 ($N = 20$), 0.81 ($N = 40$), and 0.57 ($N = 80$) for the minimal vocab size, they increase to > 0.96 for all N if the vocab size is increased. The same pattern is reflected in the MAE scores, which drop down to less than 0.35 for the larger vocabularies. In short, high accuracy is only achieved when the vocab size exceeds the number of distinct input sums, $|V| > |\mathbb{N}_{2N}|$, and a factor of two is sufficient for all tested values of N.

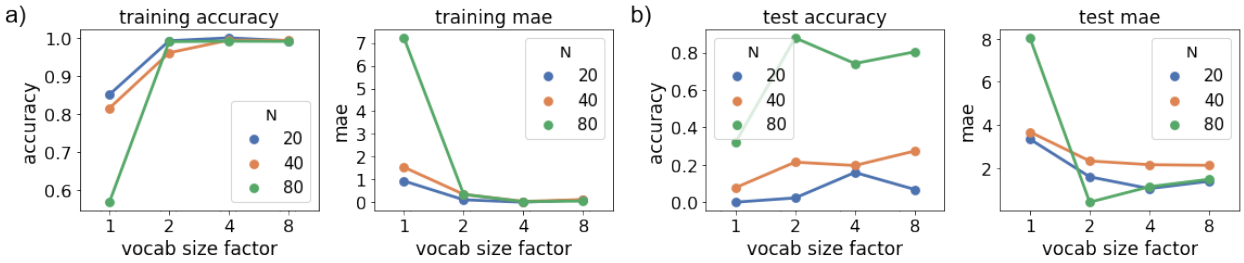

Figure 1: Accuracy and mean absolute error (MAE) on the training set (a) and the test set (b) for different input data ($N \in \{20, 40, 80\}$) and different vocab sizes ($|V|_{min} \cdot \{1, 2, 4, 8\}$).

4.2 Generalization

Figure 1.b shows accuracy (left) and MAE (right) on the test set. For $N = 20$, test accuracies range between 0.00–0.16, for $N = 40$ between 0.07–0.21, and for $N = 80$ between 0.27–0.88. Thus, the ability to generalize to novel combinations of input values largely depends on the size of the input space.

Yet, the MAEs between predicted and true sums reveal that the agents' generalization behavior is not random, even if test accuracies are low. The average MAEs for a *random* mapping between input sums and output sums lie at ~ 11.81 ($N = 20$), ~ 23.62 ($N = 40$), and ~ 46.66 ($N = 80$). Even for the minimal vocab size, the actual MAEs are far smaller (3.34, 3.67, and 8.04), indicating that the agents tend to map novel inputs to a value that lies in the vicinity of the true sum.

I study the effect of the training/test ratio on generalization using $N = 80$, and picking out one specific vocabulary size, $|V| = 644$. Figure 2 shows training and test accuracies when different proportions of the data are used for testing ($\{0.05, 0.1, 0.2, 0.3, 0.4, 0.5, 0.8\}$). While the training accuracy is not affected by the ratio, the test accuracy decreases with the training set size and lies close to chance performance for test sets containing $\geq 40\%$ of the data. The ability to generalize not only depends on the size of the input space but also on the proportion of samples covered by the training data.

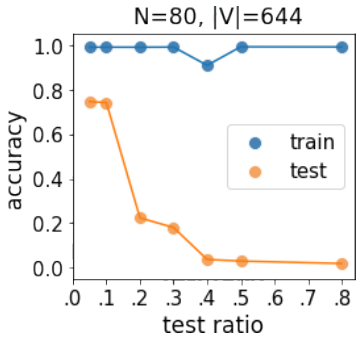

Figure 2: Accuracy on training and test set for different splits of the data, fixing $N = 80$ and $|V| = 644$.

4.3 Symbol use

I analyze the agents' symbol use focusing on those vocab sizes that lead to training accuracies > 0.95, so factors 2, 4, and 8. If you are interested in what an actual protocol, including synonymous and polysemous expressions, looks like for an example run, you can check out Appendix C.

4.3.1 Numbers of symbols used

Table 1 shows the numbers of symbols that are used at least once to describe the training samples, together with their proportion of the total vocab size in parentheses. For all values of N, more symbols are used if more symbols are available, without an obvious trend towards an increase or decrease in proportion.

		$\lvert V\rvert_{min}$ factor		
		2	4	8
N	20	58 (0.71)	86 (0.52)	149 (0.45)
	40	114 (0.70)	132 (0.41)	383 (0.59)
	80	190 (0.59)	472 (0.73)	938 (0.73)

Table 1: Number and proportion (in parantheses) of symbols used for different N and different vocab sizes.

4.3.2 Correspondence between messages and inputs

It seems that there are two main communication strategies the agents could use. Either the sender encodes information about the input values and the receiver generates their sum, or the sender encodes the sum directly. Encoding all distinct combinations of integers requires $|V| \geq |\mathbb{N}_N|^2$, whereas encoding distinct input sums only requires $|V| \geq |\mathbb{N}_{2N}|$. Above, we saw that successful communication is achieved at $|V| = 2 \cdot |\mathbb{N}_{2N}| \ll |\mathbb{N}_N|^2$ (with an even smaller active vocabulary), suggesting that the sender encodes the input sum.

I calculate the normalized mutual information between the messages and different input encodings, $\mathcal{NI}(M, I)$, to evaluate how much information is shared between messages and input sums versus messages and input summands.[1] The $\mathcal{NI}$ scores between messages and sums lie at 0.906–0.985 while the scores between messages and summands lie at 0.712–0.752. In conclusion, the correspondence between messages and sums is high and consistently constitutes the preferred encoding.

4.3.3 Synonymy and polysemy

I choose to quantify synonymy and polysemy using conditional entropies. The conditional entropy of inputs given messages, $\mathcal{H}(I \mid M)$, quantifies how much uncertainty about the input remains after learning the message and serves as a proxy for polysemy. The conditional entropy of messages given inputs, $\mathcal{H}(M \mid I)$, in turn, quantifies how much uncertainty about the message remains after knowing the input and serves as a proxy for synonymy. Rather than measuring whether symbols and inputs co-occur at all, these metrics take into account co-occurrence frequencies. E.g., synonymy will be higher if two symbols are used equally often to describe the same input than if one symbol is used much more frequently. Depending on the definition of synonymy and polysemy, one might prefer to ignore such frequency effects (see Appendix D for an alternative analysis).

Figure 3 shows the conditional entropies normalized by the respective marginal entropy. Polysemy is close to zero with respect to input sums, but hovers around 0.4 with respect to input summands. Different sums need to be encoded by different symbols, otherwise performance suffers, but different combinations of integers resulting in the same sum can be encoded by the same symbol. Synonymy is generally rather low for both input encodings and increases with the vocab size, matching the increase in used symbols observed above. A higher degree of synonymy might mean that more symbols are recruited to encode each sum, or that the agents use hybrid strategies of encoding input sums as well as input summands. As polysemy with respect to input summands does not decrease for larger vocab sizes, the former explanation seems to apply.

[1] The mutual information is normalized by the arithmetic mean of the two marginal entropies, $\mathcal{H}(I)$ and $\mathcal{H}(M)$.

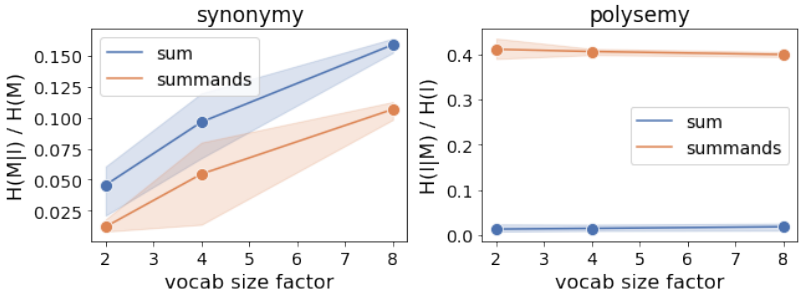

Figure 3: Normalized conditional entropy of messages given inputs (left) and inputs given messages (right). Results are averaged across $N \in \{20, 40, 80\}$. The shaded regions range from the minimal to the maximal value.

It is worth pointing out that, despite the low polysemy scores for input sums, polysemy does exist. On average, a symbol co-occurs (at least once) with 1.81 input sums (range 1.33–2.13). The low conditional entropies, however, indicate that these cases of polysemy are not systematic. Indeed, a symbol on average refers to the same input sum in 95.2% of its use cases (range 93.1%–99.1%). Considering only symbols that co-occur with more than one input sum, they still refer to the same input sum 91.0% of the time (range 86.8%–96.6%).

5 Discussion

5.1 Main findings

I trained a sender and a receiver agent on a *summation game*. The agents achieve high training accuracies if the vocab size exceeds—but not if it is equal to—the number of different sums to be communicated. Generalization to novel combinations of input integers improves with the size of the input space, and the proportion of the input space covered by the training data. Further analysis showed that the symbols encode information about the input sum, rather than the input summands. The number of different symbols used to describe the same input sum increases with the vocab size, while the number of different input sums encoded by the same symbol stays constantly low.

Some of these observations have been made before in other language emergence games. Running simulations for a reconstruction game, Chaabouni, Kharitonov, Bouchacourt, Dupoux, and Baroni (2020) found that the communication channel capacity must exceed the size of the input space for training to converge. In line with the presented results, the authors established that generalization ability strongly correlates with the input size, due to an increase in both the absolute number of distinct samples as well as the variety of seen inputs.

I used a random subset of the distinct inputs for testing. Hence, the agents must generalize to novel combinations of familiar summands. Other, more difficult test sets might come to mind. a) Holding out all samples with specific integer values at specific positions (e.g. all samples $(3, s_2)$ or $(s_1, 10)$, with $s_i \in \mathbb{N}_N$), to test whether the agents can generalize familiar integers to novel positions. b) Holding out all samples that add up to specific sums, to test whether the agents can combine the contributions of familiar integers into novel sums. I did not run these experiments. However, I think that a) will fail because the weights of the first hidden layer are not trained for the held-out values and b) will fail because neural networks have an *anti-mutual-exclusivity bias* (Gandhi & Lake, 2020), i.e. they will not map novel inputs onto outputs they have learned to avoid during training.

5.2 Limitations and possible improvements

I identified working parameters for the summation game using a grid search and these parameters surprisingly also scaled to larger values of N. The grid search suggests that it is particularly important to use agents with multiple layers.[2] Still, the training process is not very stable. This instability becomes apparent

[2]You can find a tabular representation of the grid search results in the repository

when using even larger N than reported here. It might help to more carefully tune the hyperparameters or to use other training methods than GS. Especially regimes where the receiver gradients are calculated via backpropagation and the sender gradients via REINFORCE (Williams, 1992) tend to be more stable (e.g., Chaabouni, Kharitonov, Dupoux, & Baroni, 2019). In general, artificial neurons have difficulties with the extrapolation of mathematical operations and different architectural changes have been proposed to overcome this problem (e.g, Trask et al., 2018), some of which might also be useful here.

References

Chaabouni, R., Kharitonov, E., Bouchacourt, D., Dupoux, E., & Baroni, M. (2020). Compositionality and generalization in emergent languages. In *Proceedings of the 58th annual meeting of the Association for Computational Linguistics (ACL)* (pp. 4427–4442).

Chaabouni, R., Kharitonov, E., Dupoux, E., & Baroni, M. (2019). Anti-efficient encoding in emergent communication. In *Advances in neural information processing systems* (Vol. 32).

Gandhi, K., & Lake, B. M. (2020). Mutual exclusivity as a challenge for deep neural networks. In *Advances in neural information processing systems* (Vol. 33, pp. 14182–14192).

Jang, E., Gu, S., & Poole, B. (2017). Categorical reparameterization with gumbel-softmax. In *5th international conference on learning representations (ICLR).*

Maddison, C. J., Mnih, A., & Teh, Y. W. (2017). The concrete distribution: A continuous relaxation of discrete random variables..

Trask, A., Hill, F., Reed, S. E., Rae, J., Dyer, C., & Blunsom, P. (2018). Neural arithmetic logic units. In *Advances in neural information processing systems (NeurIPS)* (Vol. 31).

Williams, R. J. (1992). Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*, *8*, 229–256.

A Hyperparameter search

I fixed certain hyperparameters to reduce the search space. All models were trained for $N = 20$ and $|V| = 200$ using Adam optimizer with learning rate 0.001 and batch size 32. Training proceeded for a maximum of 150 epochs, with a fixed early stopping accuracy of 0.99 on the training set. The minimal GS temperature was set to 0.75. The following parameter values were varied:

- number of MLP layers: $\{1, 2, 3\}$—in the case of 1 layer the input is directly mapped onto the output.
- hidden layer dimension: $\{64, 128, 256\}$
- symbol embedding layer dimension: always half of the hidden layer dimension
- GS temperature: $\{1, 1.5, 2.0\}$
- GS temperature decay (applied every epoch): $\{0.990, 0.995, 1.000\}$
- one-hot encoding of input summands: $\{\text{True}, \text{False}\}$

I selected the model with the highest training accuracy. The best model uses temperature 2.0 at decay rate 0.995 and reached accuracy > 0.99 after 77 epochs, so the temperature reached ~ 1.35. For the actual experiments, I extended the number of epochs to 250. To avoid potential instabilities at lower temperatures in later epochs, I increased the minimal temperature to 1.35.

The grid search results can be found in the corresponding folder in the github repository: `https://github.com/XeniaOhmer/summation_game/tree/main/grid_search`.

B Training accuracies

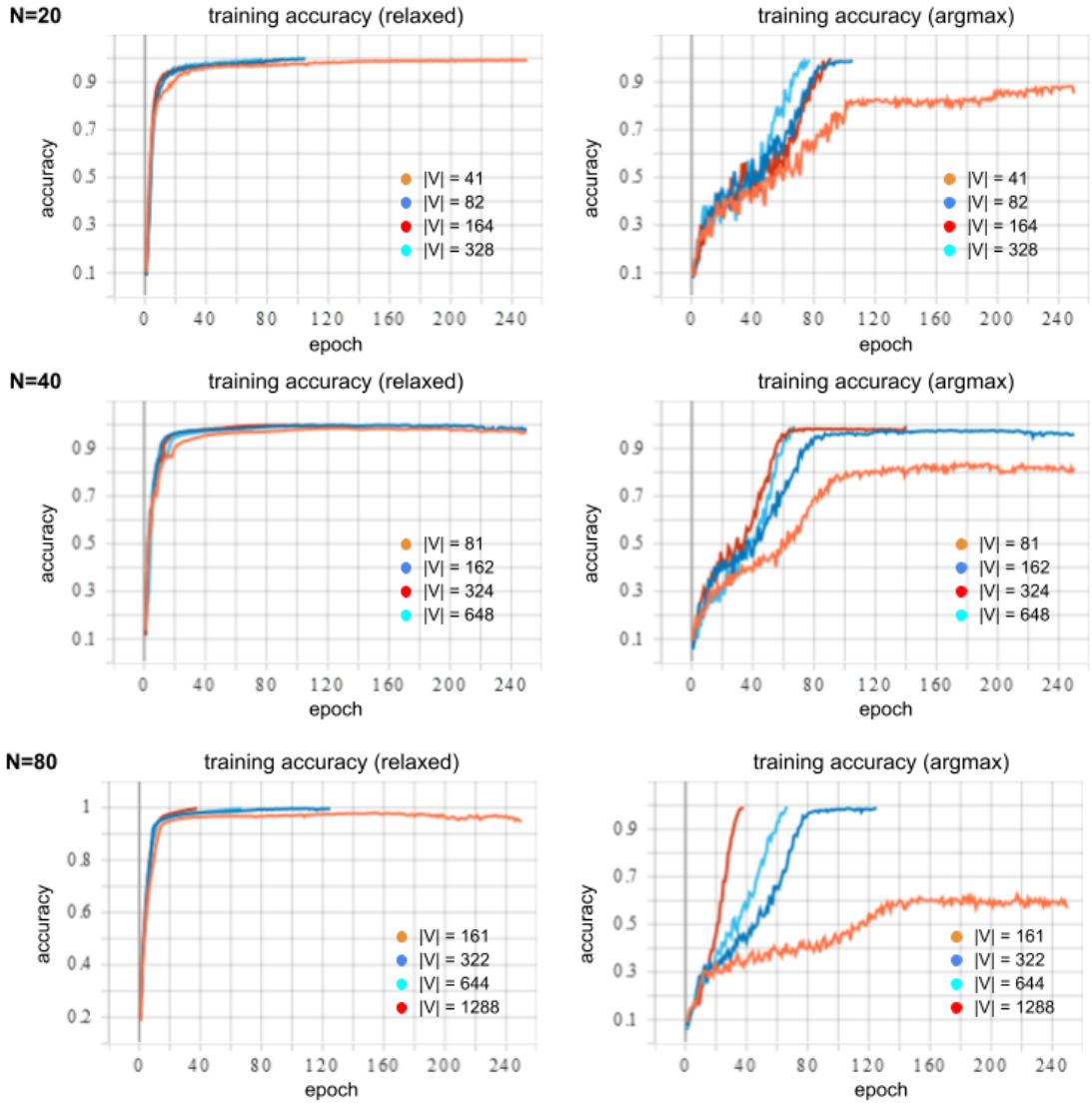

Figure 4: Accuracy for $N \in \{20, 40, 80\}$ and all vocab sizes over the course of training. The two columns show the training accuracy for the relaxed (left) and the true (right) categorical distribution over symbols. Note the reversed order of colors for $N = 80$.

C Example of a communication strategy

I selected the run with $N = 20$ and $|V| = 82$ as input size and vocab size are rather small—facilitating visualization—and training accuracy is high. Figure 5 visualizes the co-occurrences between symbols and input sums. The barplots show the number of symbols that co-occur with each input sum (top) and the number of input sums that co-occur with each symbol (bottom). The average number of symbols per sum is 2.07 and the average number of sums per symbol 1.47. Still, a symbol on average refers to the same input sum in 96.7% of its use cases, and an input sum is on average mapped to the same symbol in 92.2% of its occurrences.[3]

[3]While polysemy does not become systematic for larger vocab sizes, synonymy does. At a vocab size factor of 8, an input sum is on average mapped to the same symbol only 78.0% of the time.

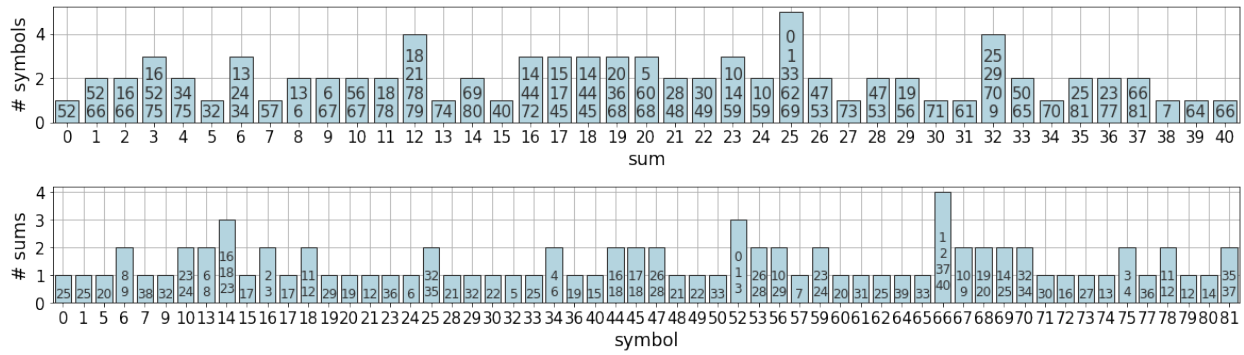

Figure 5: (Top) Synonymy: Barplot of the number of symbols that co-occur at least once with each input sum. The symbols that co-occur with each sum are used as bar labels. (Bottom) Polysemy: Barplot of the number of sums that co-occur at least once with each symbol. The sums that co-occur with each symbol are used as bar labels.

D Synonymy and polysemy — alternative analysis

Here, I quantify synonymy by calculating how many distinct symbols occur at least once per input, and polysemy by counting how many distinct inputs occur at least once per symbol. Figure 6 shows the results for different vocab size factors, values of N, and input encodings. The main results are very similar to those presented in section 4.3.3. As symbols encode information about the input sums, rather than the input summands, polysemy is very high for the latter and very low for the former. In addition, synonymy increases with the vocab size, for both input summands and input sums.

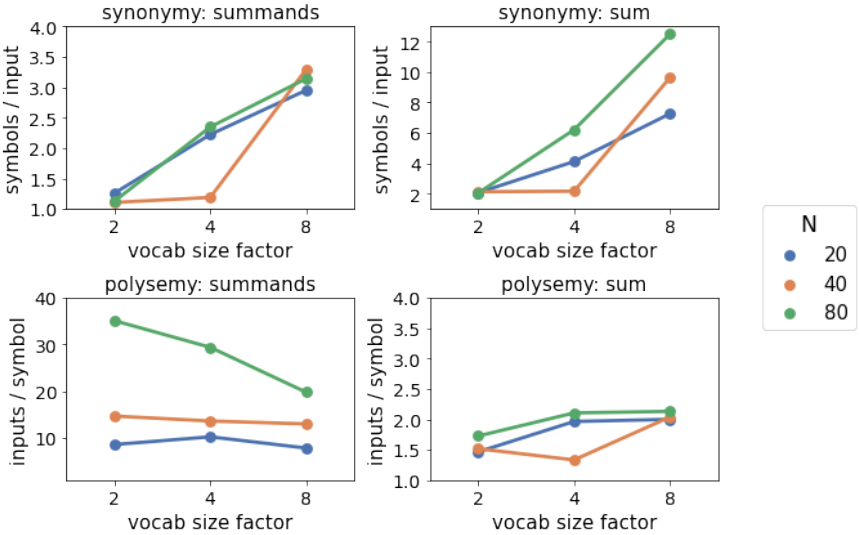

Figure 6: Synonymy (top row) and polysemy (bottom row) for different vocab sizes and different N. The scores are evaluated on the training data, either with respect to the input summands (left column) or the input sums (right column).