Language Effects on Exact Equivalence Tasks

Satchel Grant, grantsrb@stanford.edu

May 1, 2023

Abstract

Recent work has demonstrated that LSTM recurrent networks are capable of performing exact equivalence tasks with and without the use of explicit count language (Grant & McClelland, 2022). This finding sheds light on an issue in the Psychological literature of whether a mechanistic counting procedure is necessary to give humans the ability to perform exact equivalence tasks. The authors also found, however, that providing count language as an auxiliary task gives the LSTMs advantages in learning speed. In this work, we expound upon these issues by first comparing the effects of count language as an auxiliary task vs using the count language as a "technology" for the policy network. We then use PCA to examine the models' latent space to find a warping effect in language models, and we find evidence that the models are using a count up, count down strategy to solve the tasks.

1 Introduction

It has been established in the Psychological literature that members of the Pirahã, an Amazonian tribe who's language lacks exact count words, are unable to reliably perform numeric equivalence tasks past the count of 3 or 4 (Gordon, 2004). Furthermore, it has been shown that educated, adult English speakers perform poorly at exact equivalence tasks when simultaneously performing a verbal task. Spatial tasks do not have the same hindering effect. (Frank, Fedorenko, & Gibson, 2008; Frank, Fedorenko, Lai, Saxe, & Gibson, 2011; Frank, Everett, Fedorenko, & Gibson, 2008).

These findings inspired a computational simulation of the Pirahã experiments using Long Short-Term Memory (LSTM) Recurrent Neural Networks (RNNs) (Hochreiter & Schmidhuber, 1997) under the influence of different counting languages (Grant & McClelland, 2022). They used a classification task to simulate the use of language in the model. Grant and McClelland (2022) found that the LSTMs could learn to perform exact equivalence tasks, without the use of language, purely through the training signal of mapping observations to discrete actions. They found that count words did, however, improve the rate at which the models learned the task—with the cost of overfitting to quantities within their training distribution. The authors did no theoretical or empirical exploration as to why the experimental conditions led to different model performances. The answer to these questions could address whether the LSTM models have any connection to the human mind. Furthermore, it could shed light on the discussion of how language interacts with numerical cognition.

In this work, we first build on the work of Grant and McClelland (2022) to compare and contrast using language purely as an auxiliary task—serving as a latent constraint (LC) (Mu, Liang, & Goodman, 2019; Luo, Sexton, & Love, 2021)—versus using language as a technology (LT)—potentially serving as an inductive bias to break the task down into multiple steps (Wei et al., 2022). More concretely, we train LSTM models to map a gray-scale image to 6 discrete actions using behavior cloning (Sammut, 2010). We include a categorical classification task, mapping the same gray-scale images to categorical quantities as an analogy for human count words. We note that the case of LT resembles chain-of-thought reasoning (Wei et al., 2022; Andreas, Klein, & Levine, 2017). See Methods for more detail. Furthermore, we use Principle Components Analysis (PCA) on the recurrent latent states of the LSTMs to address how the models are solving the exact equivalence tasks. We also explore why the models developed in (Grant & McClelland, 2022), that use LC, have a tendency to overfit to the quantities within the training distribution.

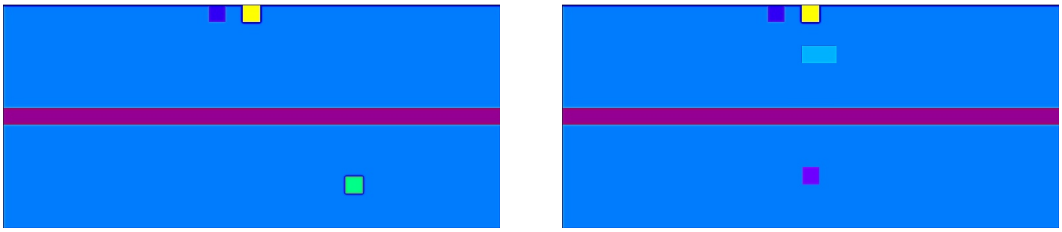
We find that the LT approach offers more robust advantages in training speed over the LC approach with a smaller amount of overfitting to quantities within its language. When looking at the latent representations, we find that the LC approach warps the latent space in such a way that the model cannot generalize. The LT approach has a similar warping effect, but to a far lesser degree. We also find evidence supporting the perspective that the LSTMs first count up to track target quantities, and then count down during the response phase of the tasks.

2 Related Work

A few publications have captured aspects of counting within Recurrent Neural Networks (RNNs). Di Nuovo and McClelland (2019) found that CNNs could count up to 5 fingers in an embodied robot setting. Fang, Zhou, Chen, and McClelland (2018) and Sabathiel, Mcclelland, and Solstad (2020) trained an LSTM to sequentially navigate to items in a display while enumerating the next count word in the sequence upon arriving at each object. In both of these works, however, the agent was never tasked with using the counting information for another task. This leaves the possibility that the model never learned numeric concepts, but rather learned a sequence of actions and a string of verbal symbols (Davidson, Eng, & Barner, 2012).

Many works have demonstrated the use of language to improve model generalization and learning robustness within the Machine Learning community. The most relevant publications for this work are (Andreas et al., 2017) who showed that a generated description of an image could be used as input to a decision making model to improve results over using the image directly. (Mu et al., 2019) showed that the language generation could improve results in some tasks purely as an auxiliary task. (Luo et al., 2021) followed up with a paper that focused on the regularizing effects of LC, using this to suggest a theory of how "language affects thought." And lastly, many works have demonstrated how tokens in the context window of a transformer can have a drastic impact on how the models solve a task. Some of these works include providing question answer pairs with explanations in context before the final question of interest (Lampinen et al., 2022); and training the model to produce a reasoning trace before answering the question, called "chain of thought prompting" (Wei et al., 2022; Zelikman, Wu, Mu, & Goodman, 2022). This list is far from exhaustive.

3 Environment and Features



(a) (b)

Figure 1: **a)** An observation during the demonstration phase. The yellow pixel is the agent, currently on top of the count button, the ending button is two spaces to the left of the agent, and the lower green pixel is a target item. **b)** An observation during the response phase. We see two response items just below the agent, and the indicator pixel below the dividing line.

The environment is a partially observable markov decision process (POMDP) (Zhang, 2010). The environment consists of a tuple (O, S, A, A^*, p_0, p) in which O is a set of observations in the form of a 15x31 grid of floating point values. The pixel values of the image correspond to different objects within the game. S is a set of game states, A is a set of actions { WAIT, LEFT, RIGHT, UP, DOWN, PRESS }, A^* is a set of optimal actions at each time step, $p_0(s_0)$ the probability of the initial state, and $p(s_{t+1} \mid a_t, s_t)$ is the probability of a transition from state s_t to s_{t+1} given the action a_t. Actions and language are represented as one hot encodings. The possible target quantities for a given trial ranges from 1-17. The possible count words learned by the agent range from 0-17.

Each trial of the environment consists of three phases: demonstration, response, and navigation. First is the demonstration phase in which the environment samples a target quantity proportionally to its inverse, $\frac{1}{n}$ where n is the target quantity. Then target items are displayed one by one, in the lower half of the visual grid, until n items have been displayed. The items disappear the time step after they were displayed, so that only one target item is displayed for any o_t. The agent cannot PRESS any buttons during the demonstration phase. At the end of the demonstration phase, a signal pixel appears in the lower half of the grid to indicate to the agent that the demonstration phase has ended. At this point, the response phase begins. In this phase, the player must PRESS the counting button the same number of times as the target quantity. Each press of the counting button causes a new visual response item to appear. The response items appear neatly in a row in the upper half of the grid. Once the agent has pressed the count button the correct number of times, it must navigate to and PRESS the Ending Button to end the game. The agent always spawns on

top of the counting button, the starting locations of both the counting and ending buttons are uniformly sampled from the top row of the grid at initialization.

Language labels differ depending on the phase of the game. During the demonstration phase, the labels correspond to the number of target items that have been displayed up until time t. During the response phase, the language labels correspond to the number of response items on the grid. During the navigation phase, no language learning takes place.

Each training epoch consists of 512 time steps over 128 different environments. At the beginning of each epoch, we collect new data tuples from the environments using the optimal actions A^*. This creates 65,536 data tuples per epoch, consisting of an observation, action, and word (o_t, a_t, and lang_t). The expected number of steps per trial is 22 (which is determined by the range of target quantities and their sampling statistics). This leads to approximately 2978 trials per epoch. To test the models, we rollout 25 trials for each target quantity from 1-27 using actions from the models. We record a trial as correct when the number of response items is equal to the target quantity.

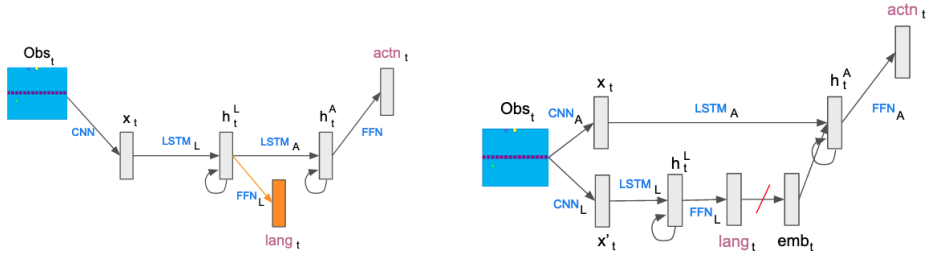


Figure 2: **a)** Language as a constraint (LC). The orange pathway denotes the auxiliary task of language prediction. **b)** Language as a technology (LT). The prediction lang_t is trained via softmax classification. The maximum argument of the language prediction vector is used to select an embedding, used as additional intput into LSTM_A. The red slash denotes a gradient stop.

4 Methods

We build off the architectures and training regimes used in Grant and McClelland (2022). We train models to map gray-scale pixels o_t to action classes a_t and count words lang_t. The loss is calculated as follows.

$$\ell(a, w) = \sum_{i=1}^{N} -\log\left(\frac{e^{a_{y_i}}}{\sum_j^{|A|} e^{a_j}}\right) - \log\left(\frac{e^{\text{lang}_{y_i}}}{\sum_j^{|L|} e^{\text{lang}_j}}\right)$$

Where a_j is the action prediction at index j and lang_j is the language prediction. y_i denotes the correct label for sample i. N is the number of samples in the training data. We use stochastic gradient descent and backpropagation through time to train the model. We backpropagate through all time steps of each trial.

The No-Language, baseline model consists of a two layer Convolutional Neural Network (CNN), using 9 and 18 channels, 0 padding, a stride of 1, and ReLU nonlinearities. The output of the CNN is reshaped and fed into $LSTM_L$. We use 24 units for the LSTM hidden states, $h_t \in R^{24}$. We then perform a layer normalization on h_t^L and use it as the input to $LSTM_A$. The output, h_t^A, is then mapped to an action prediction a_t using FFN_A. The LC model uses the same architecture as the No-Language model, except that it includes an auxiliary language prediction, mapping h_t^L to lang_t using FFN_L.

The LT model architecture uses a separate pathway for the language prediction. The output of CNN_L and CNN_A are fed into $LSTM_L$ and $LSTM_A$ respectively. Furthermore, the argmax of lang_t is used to select an embedding, $\text{emb}_t \in R^{24}$, which is used as an input into $LSTM_A$. Gradients are not propagated through this selection operation. See Figure 2b for more detail.

Each FFN consists of two layers with a hidden layer size of 72 and a ReLU nonlinearity, we train the models using RMSprop with no regularization using a learning rate of 5e-4, we train for a total of 95 epochs using the PyTorch optimization framework (Paszke et al., 2019). We train each model variant over 10 random seeds. At inference, we use a hard max to sample actions from the models, $\hat{a}_t = \arg\max_a f_a(o_t)$.

For the PCA analysis, we use h_t^A at each time step over 10 trials for each target quantity 1-27 from fully trained models. We then project each h_t^A vector into two dimensions. We refer readers to (Syms, 2008) for an in depth explanation of PCA. We used scipy to impelement PCA in code (McKinney, 2010).

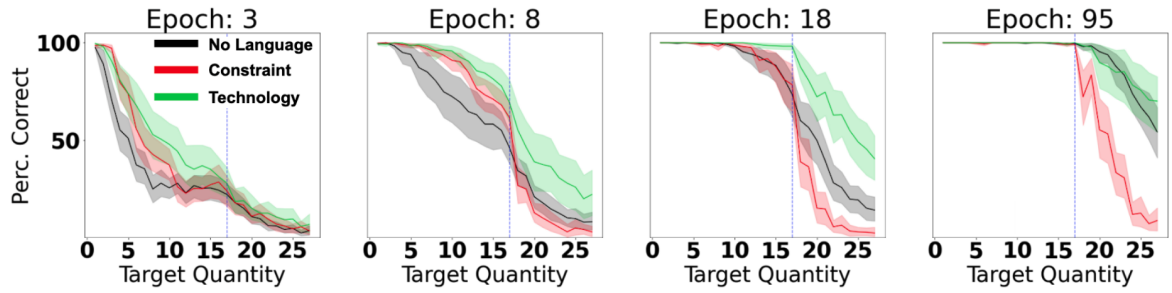


Figure 3: LC (Constraint) and LT (Technology) performance on unseen trials for different target quantities. To the left of the blue dashed line denotes target quantities (not trials) within the training distribution. The models have not seen quantities to the right of the blue dashed line during training.

5 Experiments, Results, and Discussion

5.1 Language as a Constraint vs Language as a Technology

We first look at the differences between the LC model and the LT model. The average testing accuracy for the LC models averaged over all 10 model seeds, over all epochs, was 73.17% ± 0.15 using SEM over model seeds to calculate error. The LT models by comparison had an accuracy of 86.35% ± 0.12. This indicates that the LT models are more robust over the entire training period. The LT model also beats the results of the No-Language baseline, 84.97% ± 0.11. From **Figure 3**, we can see that the LT models manage to generalize better to unseen target quantities over most stages of training. We also see that the LC models are far worse than the other models at generalizing to unseen quantities. We suspect that the reason why the LT models perform best is that the count words assist the $LSTM_A$ in breaking down the task into subtasks. However, we do not have hard evidence to defend this claim.

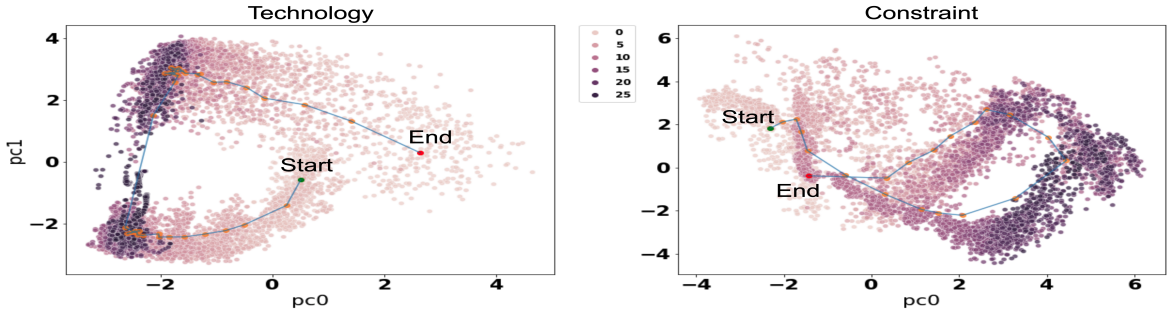


Figure 4: Projections of h_t^A in an LT model on the left vs an LC model on the right. We can see a warping effect in both models that is not present in the No-Language variant. Compare to **Figure 5**.

5.2 PCA Analysis

In **Figure 5**, we show the PCA projections of hidden states from a single model's $LSTM_A$. We can see a horse-shoe shape of the states, where the trajectories start in the lower left, move out to the right during the demonstration phase, then loop up at the beginning of the response phase, and finally move back to the left to end the response phase. If we were to interpret pc0 as an axis that encodes magnitude and pc1 as an axis that encodes phase, we would conclude that the model first counts up during the demonstration phase and then down during the response phase.

To explore this idea further, we include **Figure 6** which shows that the model is compressing the latent space at larger quantities during the demonstration phase (we note that is reminiscent of the logarithmically compressed numberline known to exist in humans (Dehaene, 2003)). States within the response phase are similarly compressed, although in reverse. We can see this in **Figure 6** which shows the state transition distances according to the difference between the target quantity and the number of response items. This phenomenon supports the claim that the model is using a count up, count down strategy as opposed to a count up, count up strategy. We believe the LSTMs are biased toward this strategy due to the difficulty of remembering a quantity over multiple time-steps (Van Houdt, Mosquera, & Nápoles, 2020).

Lastly, we can see in **Figure 4** that the latent space is warped when training with language. We can see a greater amount of warping in the LC models than the LT models. This explains the reason that the LC models tend to overfit to the quantities within the training distribution.

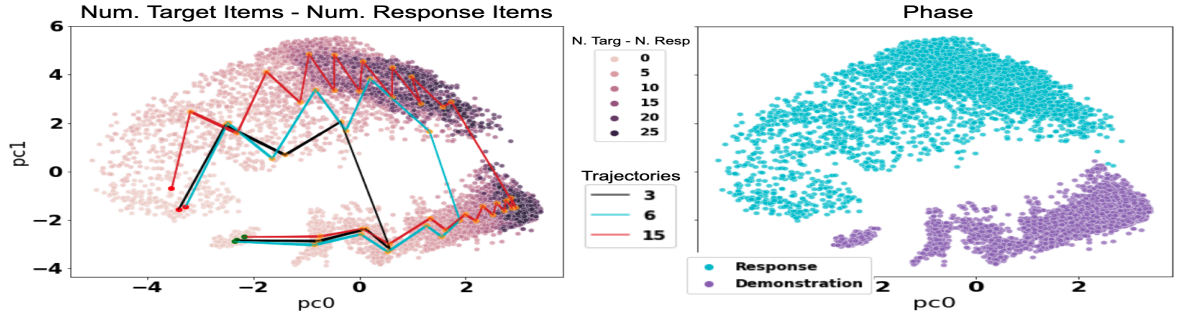


Figure 5: Both panels display the first two PCs of the h_t^A vectors from a No-Language model collected over 10 trials at each target quantity. For each point, the color on the left panel denotes the number of target items that have been displayed up until that point minus the number of response items that are currently displayed. The left panel displays example trajectories for trials with target quantities 3, 6, and 13 (arbitrarily selected). These example trials start at the green points and end at the red points. In the right panel, we can see the phase of the experiment. The navigation phase has been removed from the visual.

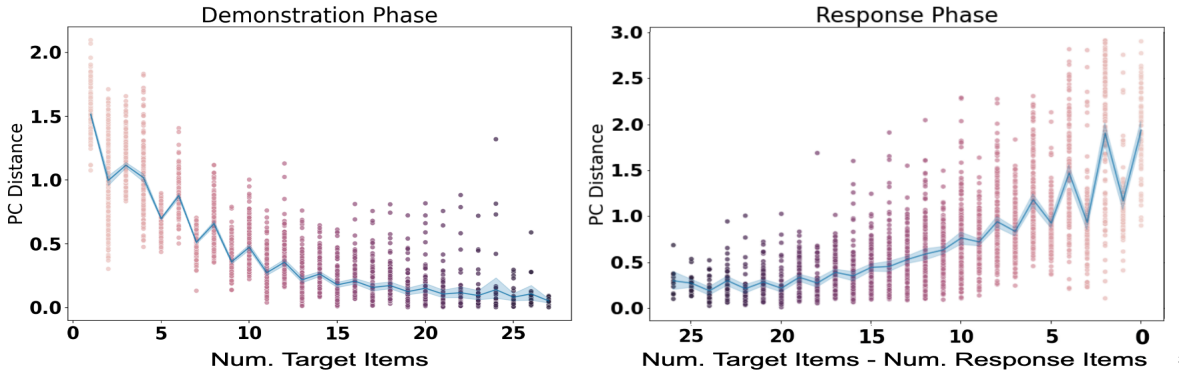


Figure 6: L2 distance between first 2 PCs of h_t^A and h_{t-1}^A along trial trajectories. The x-axis is the number of target items that have been displayed so far minus the number of response items. We see a compression of state space for larger quantities. This favors the view that the model first counts up during the demonstration phase and then counts down for the response phase before finishing the trial.

6 Conclusion

In this work, we demonstrated that within exact quantity tasks, the use of language as a technology improves model training speed and generalization over using language as a constraint. We also explored the latent representations of the models to find that the model compresses quantities of larger magnitude, and uses a count up, count down strategy to solve the exact equivalence tasks. Lastly, we showed that training with count words tends to warp the latent space of the models. This gives insight into why the LC models had a large amount of overfitting to in-distribution target quantities. This work adds to the debate about how best to use language in deep learning models, and this work adds to our collective understanding of neural representations of number.

7 Acknowledgments

Thank you to Jay McClelland for access to the CCN cluster, and for providing the direction that this work builds upon. Also, thank you to the instructors and TAs for making this a fun class.

References

Andreas, J., Klein, D., & Levine, S. (2017). Learning with latent language. *CoRR*, *abs/1711.00482*. Retrieved from http://arxiv.org/abs/1711.00482

Davidson, K., Eng, K., & Barner, D. (2012). Does learning to count involve a semantic induction? , 162–173.

Dehaene, S. (2003). The neural basis of the weber-fechner law: A logarithmic mental number line. *Trends in Cognitive Sciences*, *7*(4), 145–147. doi: 10.1016/S1364-6613(03)00055-X

Di Nuovo, A., & McClelland, J. L. (2019). Developing the knowledge of number digits in a child-like robot. *Nature Machine Intelligence*, *1*(12), 594–605. Retrieved from http://dx.doi.org/10.1038/s42256-019-0123-3 doi: 10.1038/s42256-019-0123-3

Fang, M., Zhou, Z., Chen, S., & McClelland, J. L. (2018). Can a recurrent neural network learn to count things? *Proceedings of the 40th Annual Conference of the Cognitive Science Society*, 360–365.

Frank, M. C., Everett, D. L., Fedorenko, E., & Gibson, E. (2008). Number as a cognitive technology: Evidence from pirahã language and cognition. *Cognition*, *108*(3), 819-824. doi: https://doi.org/10.1016/j.cognition.2008.04.007

Frank, M. C., Fedorenko, E., & Gibson, E. (2008). Language as a cognitive technology: English-speakers match like pirahã when you don't let them count..

Frank, M. C., Fedorenko, E., Lai, P., Saxe, R., & Gibson, E. (2011). Verbal interference suppresses exact numerical representation : online lexical encoding as an account of cross-linguistic differences in cognition..

Gordon, P. (2004). Numerical cognition without words: Evidence from Amazonia. *Science*, *306*(5695), 496–499. doi: 10.1126/science.1094492

Grant, S., & McClelland, J. (2022). Recurrent models of number as a cognitive technology. *preprint*. Retrieved from shorturl.at/eJOZ3

Hochreiter, S., & Schmidhuber, J. (1997, nov). Long short-term memory. *Neural Comput.*, *9*(8), 1735–1780. Retrieved from https://doi.org/10.1162/neco.1997.9.8.1735 doi: 10.1162/neco.1997.9.8.1735

Lampinen, A. K., Dasgupta, I., Chan, S. C. Y., Matthewson, K., Tessler, M. H., Creswell, A., ... Hill, F. (2022). Can language models learn from explanations in context? , 1–29. Retrieved from http://arxiv.org/abs/2204.02329

Luo, X., Sexton, N. J., & Love, B. C. (2021). A deep learning account of how language affects thought. *Language, Cognition and Neuroscience*, *0*(0), 1–10. Retrieved from https://doi.org/10.1080/23273798.2021.2001023 doi: 10.1080/23273798.2021.2001023

McKinney, W. (2010). Data structures for statistical computing in python. In S. van der Walt & J. Millman (Eds.), *Proceedings of the 9th python in science conference* (p. 51 - 56).

Mu, J., Liang, P., & Goodman, N. D. (2019). Shaping visual representations with language for few-shot classification. *CoRR*, *abs/1911.02683*. Retrieved from http://arxiv.org/abs/1911.02683

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., ... Chintala, S. (2019). Pytorch: An imperative style, high-performance deep learning library. , 8024–8035.

Sabathiel, S., Mcclelland, J. L., & Solstad, T. (2020). Emerging Representations for Counting in a Neural Network Agent Interacting with a Multimodal Environment. (7491).

Sammut, C. (2010). Behavioral cloning. In C. Sammut & G. I. Webb (Eds.), *Encyclopedia of machine learning* (pp. 93–97). Boston, MA: Springer US. Retrieved from https://doi.org/10.1007/978-0-387-30164-8$_6$9 doi: 10.1007/978-0-387-30164-8$_6$9

Syms, C. (2008). Principal components analysis. In S. E. Jørgensen & B. D. Fath (Eds.), *Encyclopedia of ecology* (p. 2940-2949). Oxford: Academic Press. Retrieved from https://www.sciencedirect.com/science/article/pii/B9780080454054005383 doi: https://doi.org/10.1016/B978-008045405-4.00538-3

Van Houdt, G., Mosquera, C., & Nápoles, G. (2020). A review on the long short-term memory model. *Artificial Intelligence Review*, *53*(8), 5929–5955. doi: 10.1007/s10462-020-09838-1

Wei, J., Wang, X., Schuurmans, D., Bosma, M., Chi, E. H., Le, Q., & Zhou, D. (2022). Chain of thought prompting elicits reasoning in large language models. *CoRR*, *abs/2201.11903*. Retrieved from https://arxiv.org/abs/2201.11903

Zelikman, E., Wu, Y., Mu, J., & Goodman, N. D. (2022). *Star: Bootstrapping reasoning with reasoning.* arXiv. Retrieved from https://arxiv.org/abs/2203.14465 doi: 10.48550/ARXIV.2203.14465

Zhang, H. (2010). Partially observable markov decision processes: A geometric technique and analysis. *Operations Research*, *58*(1), 214–228. Retrieved 2022-10-28, from http://www.jstor.org/stable/40605971