Generalized Schema Learning by Neural Networks

Catherine Chen (cathy.chen@princeton.edu)[*]

Qihong Lu (qihong.lu@princeton.edu)[†]

Andre Beukers (abeukers@princeton.edu)[†]

Chris Baldassano (chrisb@princeton.edu)[†]

Kenneth Norman (knorman@princeton.edu)[†]

[*]Princeton Computer Science Department, Princeton NJ 08540

[†]Princeton Neuroscience Institute and Princeton Psychology Department, Princeton NJ 08540

Abstract

Humans have schematic knowledge of how certain types of events unfold (e.g., restaurant meals) that can readily be generalized to new instances of those events. Here, we test whether neural networks can do this kind of generalized schema learning. We stochastically generate stories according to predefined rules and test networks' ability to answer questions about role fillers (e.g., who is the waiter). We find that networks trained on a small set of fillers can only generalize to stories using the same fillers. By training on a large number of fillers represented by random vectors, we allow networks to generalize to fillers they have never seen before. We find qualitative differences in learning ability between networks with different forms of memory, with some networks learning fewer categories of tasks than others. We further find distinct influences of task difficulty on learning order, and of training order on learning ability.

Keywords: neural networks; schema; generalization

Introduction

Humans acquire schematic knowledge about the structure of particular types of events (e.g., restaurant meals), which we can readily apply to new instances of those events (e.g., we may have never seen this particular waiter, but we can still predict how they will act) (Bower, Black, & Turner, 1979) A combination of long- and short-term memory enables this: long-term memory stores schemas created over many experiences, and short-term memory tracks information about the current schema instance.

Recurrent neural networks (RNNs) are a class of neural network architectures with weights that form directed cycles, forming feedback loops that allow the networks to maintain an internal state. These feedback loops allow the networks to maintain both long-term (through the standard weight updates of a neural network) and short-term (through the information stored in the internal state) memory. These networks have shown success on a wide range of tasks such as speech recognition (Graves, Mohamed, & Hinton, 2013) and language modeling (Mikolov & Zweig, 2012). In this work we examine their ability to acquire generalized event knowledge, by exposing the networks to procedurally-generated stories and testing them on their ability to answer questions about role fillers from the just-presented story. Prior work has shown that neural networks can do this when they are explicitly told what filler information to maintain (St. John & McClelland, 1990); here, we are particularly interested in whether networks can learn to maintain filler information on their own, and whether they can generalize to new fillers that they have not seen before.

Methods

We construct stories using Coffee Shop World, a generator that writes stories based on predefined rules. Coffee Shop World models stories as a graph in which nodes represent states of the story and edge weights represent transition probabilities between different states. Each state is a frame that includes fixed text and variable "roles", which are substituted with specific "fillers" in each instance of the story. For instance, an "Order_food" state might read:

[Subject] ordered a plate of [Food]

and in a specific instance of the story *[Subject]* and *[Food]* would be substituted with specific fillers, such as "Alice" and "sandwiches". Given a schema that defines states, transitions between them, and possible fillers for each role in the story, Coffee Shop World probabilistically generates stories that instantiate the schema.

We test networks' ability to perform role-filler binding, the task of associating abstract roles with concrete fillers. Given an input containing a story and a query specifying a role, the networks must return the corresponding filler. For instance, given the input

Alice ordered a plate of sandwiches ? QSubject

the network should return *Alice*.

In this experiment, we use the schema corresponding to the story graph in Figure 1 and the state definitions in Table 1. This schema contains six roles which correspond to the tasks *QDessert*, *QDrink*, *QEmcee*, *QPoet*, and *QSubject* (where "*QX*" denotes the task of identifying the filler corresponding to role *X*). Note that the *QSubject* task is the easiest, as the *Subject* always occurs as the second word in the story. Other roles do not always occur at a fixed location; for instance, the most consistent appearance of the *Friend* (in the "Sit_down" state) has three possible locations. The *QDessert* and *QEmcee* tasks are the hardest: the *Dessert* and *Emcee* do not occur in every story, and they do not have a fixed location even when they do occur in a story.

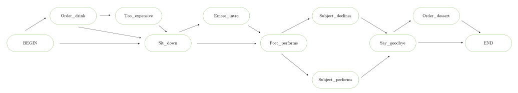


Figure 1: Story graph for role-filler binding experiments. Each edge indicates a possible transition. In our schema, for states with multiple outgoing transitions, each outgoing transition is equally likely.

State Name	**State Frame**
BEGIN	BEGIN [Subject]
Order_drink	Order_drink [Subject] [Drink]
Too_expensive	Too_expensive [Subject]
Sit_down	Sit_down [Subject] [Friend]
Emcee_intro	Emcee_intro [Emcee] [Poet]
Poet_performs	Poet_performs [Poet]
Subject_declines	Subject_declines [Subject]
Subject_performs	Subject_performs [Subject] [Friend]
Say_goodbye	Say_goodbye [Subject] [Friend]
Order_dessert	Order_dessert [Subject] [Dessert]
END	END [Subject]

Table 1: Story states for role-filler binding experiments. We provide the text of each state of the story, where the bracketed roles are substituted by specific fillers in each story.

We show tests with four neural network architectures with distinct forms of memory: a standard recurrent neural network (RNN), Long Short-Term Memory (LSTM) (an RNN with gates to control what the internal state stores, forgets, and displays to the rest of the network) (Hochreiter & Schmidhuber, 1997), Fast Weights (an RNN with a matrix of quickly changing "fast weights" that enable auto-associative memory) (J. Ba, Hinton, Mnih, Leibo, & Ionescu, 2016), and Differential Neural Computer (an RNN with an LSTM "controller" that learns to read to and write from an external buffer) (Graves et al., 2016). We use layer normalization for the RNN, LSTM, and Fast Weights architectures. This re-centers and re-scales the networks' layers and serves to stabilize the network dynamics (J. L. Ba, Kiros, & Hinton, 2016).

We represent each word of the story as a vector (using either one-hot or random Gaussian vectors for each experiment). At each time step we present one word to the network. After presenting all words in the input we determine the network's prediction by finding the word vector in the experiment's corpus with the closest cosine similarity to the network's output.

Results

Previously Seen Fillers

When networks are trained and tested on stories with fillers chosen from the same finite pool, networks can perform role-filler binding, as we show in Figure 2 for random embeddings.

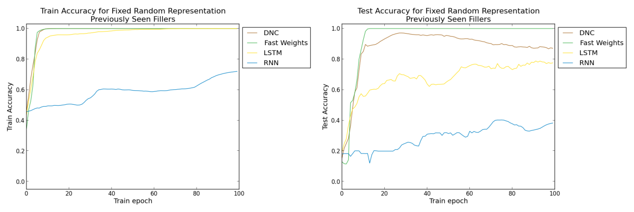


Figure 2: Role-filler binding succeeds with previously seen fillers. Networks learn to recall the filler fulfilling a specified role when the fillers used in stories are shared between the train and test sets.

Previously Unseen Fillers

If we train networks on stories with a small set of fillers, networks cannot generalize to previously unseen fillers, as we show in Figure 3 for random embeddings. For instance, if a network has never been trained on stories with the fillers *Chris* or *tacos* and is tested on the story

Chris ordered a plate of tacos ? QSubject

then it fails to retrieve the correct output, even when it has seen stories with the same frame during training. This happens even in experiments in which each input contains the query *QSubject*, and networks could simply learn to always return the second word of the story. In contrast, humans easily generalize to previously unseen fillers, successfully answering the input

Clkwef ordered a plate of talsk ? QSubject

even without ever having seen the words "Clkwef" or "talsk".

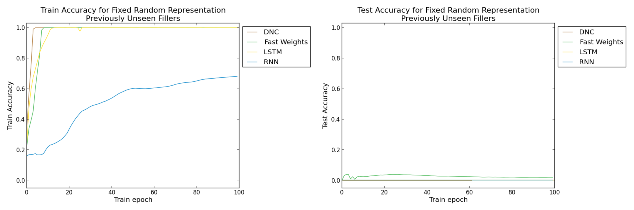
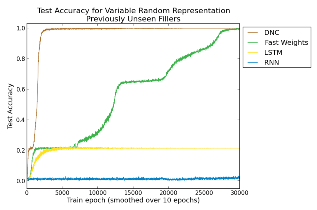


Figure 3: Role-filler binding fails on previously unseen fillers when the fillers in test set stories do not match those in train set stories.

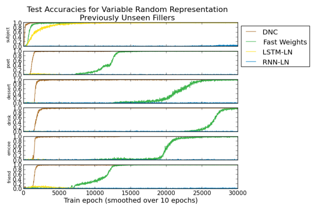
Previous work found that neural networks perform quite poorly when the distribution of the training set differs from that of the test set, which provides a possible explanation for the lack of generalization to unseen fillers (The Alan Turing Institute, n.d.). We address this problem by continually introducing new randomly generated fillers during training.

We designed a "variable random representation" training regime, in which we generate a new random vector for each input. For instance, the vector representing the word "Alice" is newly generated in each example received by the network. Under this regime, some networks successfully generalize to previously unseen fillers during test, as we show in Figure 4a. Note that using random vector representations allows the train and test fillers to come from the same distribution.

The accuracies of DNC and Fast Weights display a salient stepwise structure, and we show accuracies split by input query in Figure 4b. Some networks only learn to solve the easiest task (*QSubject*), while networks that learn to solve multiple tasks first learn the easiest task before continuing to learn more difficult tasks (such as *QPoet* or *QEmcee*).



(a) Overall accuracies



(b) Accuracies split by query

Figure 4: Role-filler binding succeeds on previously unseen fillers, using variable random fillers. When trained on stories that continually introduce previously unseen fillers, networks learn to generalize to previously unseen fillers. Task difficulty influences networks' ability to learn certain tasks and speed of learning.

Curriculum Learning

In single-task experiments (where all inputs contain the same task), the LSTM can learn the *QSubject* or *QPoet* task. However, in multi-task experiments (where inputs can contain different tasks), it only learns the *QSubject* task. Previous research has found that curriculum learning can accelerate learning (Bengio, Louradour, Collobert, & Weston, 2009). We find that it can also allow the network to learn additional tasks.

We constructed a curriculum training regime in which the LSTM first learns one task before being trained on both tasks. If the LSTM is trained on the *QPoet* task first it can learn both tasks, while if it is trained on the *QSubject* task first it only learns *QSubject task*. We show test accuracies for both curricula in Figure 5.

Discussion

Training on a small set of fillers makes networks unable to generalize beyond the fillers it has seen. Using random Gaussian embeddings and providing a large number of fillers – a new filler for each example – during training forces the network to generalize beyond a small set of fillers, and allows networks to successfully perform role-filler binding on previously unseen fillers.

Different forms of memory qualitatively affect performance: networks differ not only in their speed of learning, but also

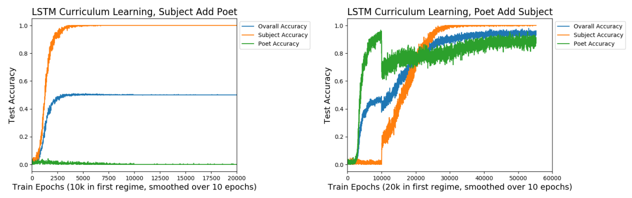


Figure 5: Order of learning affects success of role-filler binding. We train networks on a curriculum containing two regimes. In the first regime they receive only examples with a single query, and in the second regime they receive examples containing both queries. First training the network on the harder task allows the network to learn both tasks, while starting with the easier task blocks the network from learning the harder task.

in the tasks they are able to learn. When trained on inputs consisting of six queries, Fast Weights and DNC (biologically inspired by the existence of synaptic learning at different timescales and complementary learning systems) learn to answer multiple tasks where LSTM and RNN learn to answer only a single task. The single task learned by LSTM and RNN (*QSubject*) is the easiest task, and is also the task that Fast Weights and DNC learn first.

Saxe et al (Saxe, McClelland, & Ganguli, 2013) provide a possible explanation for this phenomena of stepwise accuracies. They show that a three-layer linear network given orthogonal input representations learns stronger input-output correlations more quickly, where singular values of the input-output correlation matrix denote input-output correlation strength. The different timescales of learning different tasks could explain the appearance of staggered characteristic learning curves in the split accuracy plots in Figure 4b.

Furthermore, the order of learning can affect networks' learning ability. First training the LSTM on the easier task or training on both tasks simultaneously causes the network to only learn the easier task. Perhaps the network latches onto a simpler method of solving the task, one which allows it to solve *QSubject* but not *QPoet*. First training the network on *QPoet* may force the network to find a more useful approach to role-filler binding, one which allows it to learn both tasks.

Our analysis of specific incorrect predictions shows patterns in networks' errors. While the standard RNN did not show patterns in networks' errors, the vast majority of the LSTM's mistakes occur when the network predicts the story's *Subject* rather than the filler requested by the query. In the Fast Weights architecture, errors also occur from providing the correct answer to the wrong query. Fast Weights' errors come from its incorrect prediction of a story's *Emcee* instead of its *Dessert* and vice versa. This suggests that these errors may come from failing to identify which task to perform (correctly answering queries in our experiments requires the network to learn to both (a) identify the task and (b) carry out the task).

Future work will analyze these architectures' use of memory using techniques akin to neuroimaging to better understand how they perform schema learning, and why they sometimes fail.

Acknowledgments

Funding was provided by a grant from Intel Labs and by a Multi-University Research Initiative grant to KAN (ONR/DoD N00014-17-1-2961). Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the Office of Naval Research or the U.S. Department of Defense.

References

Ba, J., Hinton, G., Mnih, V., Leibo, J. Z., & Ionescu, C. (2016, October). Using Fast Weights to Attend to the Recent Past. *arXiv:1610.06258 [cs, stat]*. Retrieved 2018-01-30, from `http://arxiv.org/abs/1610.06258` (arXiv: 1610.06258)

Ba, J. L., Kiros, J. R., & Hinton, G. E. (2016, July). Layer Normalization. *arXiv:1607.06450 [cs, stat]*. Retrieved 2018-02-02, from `http://arxiv.org/abs/1607.06450` (arXiv: 1607.06450)

Bengio, Y., Louradour, J., Collobert, R., & Weston, J. (2009). Curriculum Learning. In *Proceedings of the 26th Annual International Conference on Machine Learning* (pp. 41–48). New York, NY, USA: ACM. Retrieved 2018-04-26, from `http://doi.acm.org/10.1145/1553374.1553380` doi: 10.1145/1553374.1553380

Bower, G. H., Black, J. B., & Turner, T. J. (1979). Scripts in memory for text. *Cognitive psychology*, *11*(2), 177–220.

Graves, A., Mohamed, A. r., & Hinton, G. (2013, May). Speech recognition with deep recurrent neural networks. In *2013 IEEE International Conference on Acoustics, Speech and Signal Processing* (pp. 6645–6649). doi: 10.1109/ICASSP.2013.6638947

Graves, A., Wayne, G., Reynolds, M., Harley, T., Danihelka, I., Grabska-Barwiska, A., . . . Hassabis, D. (2016, October). Hybrid computing using a neural network with dynamic external memory. *Nature*, *538*(7626), 471–476. Retrieved from `http://dx.doi.org/10.1038/nature20101`

Hochreiter, S., & Schmidhuber, J. (1997, November). Long short-term memory. *Neural Computation*, *9*(8), 1735–1780.

Mikolov, T., & Zweig, G. (2012, December). Context dependent recurrent neural network language model. In *2012 IEEE Spoken Language Technology Workshop (SLT)* (pp. 234–239). doi: 10.1109/SLT.2012.6424228

Saxe, A. M., McClelland, J. L., & Ganguli, S. (2013, December). Exact solutions to the nonlinear dynamics of learning in deep linear neural networks. *arXiv:1312.6120 [cond-mat, q-bio, stat]*. Retrieved 2018-04-14, from `http://arxiv.org/abs/1312.6120` (arXiv: 1312.6120)

St. John, M. F., & McClelland, J. L. (1990). Learning and applying contextual constraints in sentence comprehension. *Artificial intelligence*, *46*(1-2), 217–257.

The Alan Turing Institute. (n.d.). *Recurrent Neural Networks and Models of Computation - Edward Grefenstette, DeepMind.* Retrieved 2018-02-28, from `https://www.youtube.com/watch?v=OWZ3mtzpn7g`