Attention Is Not Enough

Joshua Miller[1] **(jmill233@vols.utk.edu)**
Department of Mathematics, University of Tennessee, Knoxville
Knoxville, TN 37904 USA

Shawheen Naderi[1] **(snaderi@csus.edu)**
Department of Mathematics, California State Univeristy, Sacramento
Sacramento, CA 95819, USA

Chaning B. Mullinax (cbm5d@mtmail.mtsu.edu)
Department of Computer Science, Middle Tennessee State University,
Murfreesboro, TN 37132 USA

Joshua L. Phillips (Joshua.Phillips@mtsu.edu)
Department of Computer Science, Middle Tennessee State University,
Murfreesboro, TN 37132 USA

Abstract

The human ability to generalize beyond interpolation, often called extrapolation or symbol-binding, is challenging to recreate with computational models. Biologically plausible models incorporating *indirection* mechanisms have demonstrated strong performance in this regard. Deep learning approaches such as Long Short-Term Memory (LSTM) and Transformers have shown varying degrees of success, but recent work has suggested that Transformers are capable of extrapolation as well. We evaluate the capabilities of the above approaches on a series of increasingly complex sentence-processing tasks to infer the capacity of each individual architecture to extrapolate sentential roles across novel word fillers. We confirm that the Transformer does possess superior abstraction capabilities compared to LSTM. However, what it does *not* possess is extrapolation capabilities, as evidenced by clear performance disparities on novel filler tasks as compared to working memory-based indirection models.

Keywords: Deep Neural Networks, Long Short Term Memory, Transformers, Indirection

Introduction

Generalization is the process of applying previously learned abilities and knowledge to novel experiences. It is an essential human faculty because no two experiences are exactly the same. One may have earlier experienced similar components of an event but the exact manner these components are arranged in, or even the components themselves, likely will not be identical. For instance, one is often able to infer the meaning of a sentence even if it contains a word outside of one's lexicon. While deep artificial neural networks (ANNs) are computational systems capable of some forms of generalization, challenges remain when generalizing given novel inputs. One important thread of research in this area focuses on *symbol processing*, or "the ability to represent information in the form of abstract variables that can be bound to arbitrary values", which plays an important role in one's ability to generalize using working memory (Kriete, Noelle, Cohen, & O'Reilly, 2013). In this paper, we build off of previous research regarding deep ANN models also reported to exercise this ability. We evaluate their capabilities on a series of increasingly complex sentence-processing tasks and are therefore able to infer the capacity of each individual architecture to extrapolate to novel inputs.

Background

Biologically Plausible Indirection

The interacting subsystems of the brain thought to underlie human working memory may allow for *functionality analogous to the concept of pointers from computer science* via a process called *indirection* (Kriete et al., 2013). Using indirection, abstract values located in one region of memory can store the "addresses" of arbitrary values located in another memory region. The prefrontal cortex (PFC) contains stripes of densely intraconnected neurons which are sparsely interconnected to one another (Elston, Benavides-Piccione, Elston, Manger, & Defelipe, 2011) and appears suited to active maintenance of memory traces. These stripes are strongly linked with the basal ganglia (BG) which is appears suited for controlling active memory gating/updating (Alexander, DeLong, & Strick, 1986). Consequently, Kriete et al. hypothesized that different PFC stripes may be employed to maintain pointer-like memory traces and thereby implement a form of indirection with the BG responsible for the gating/updating of information in PFC stripes and proposed a biologically plausible computational model with generalization capabilities that utilize indirection (Kriete et al., 2013).

In Kriete et al., the abstract variables are three possible roles in a three word sentence: agent, verb, and patient. The arbitrary values are known as fillers and are the concrete words which correspond to each role. For example, in the sentence "Tom Ate Food" the agent is "Tom", the verb is "Ate", and the patient is "Food". The function of the model is to take the previously discussed three word sentence and individually encode each role and filler into working mem-

[1] Equal contribution. Listing order is alphabetical by last name.

ory. Then, the model will be queried using the role, and the filler which was stored with the associated role will be recalled. The model was tested on three tasks, each requiring a different form of generalization performance:

- **Standard Generalization (SG)** – During training, every filler has been used in every role and all fillers have been in the same sequence together. For instance, if the fillers are "apple", "bat", and "cat" and the roles are their respective ordering in the sequence, during training the model may have been presented with "apple bat cat", "cat apple bat", and "bat cat apple". Thus, "apple", "bat", and "cat" had all been used in every role and had been used in the same sequence. During testing, unique combinations of role-filler pairs are employed, for example: "cat bat apple".
- **Spurious Anticorrelation (SA)** – During training, every filler has been used in every role but *not all* fillers have been in the same sequence together. For instance, if the fillers are "apple", "bat", "cat", and "dog" and the roles are their respective ordering in the sequence, during training the model may have been presented with "apple bat cat" and "cat bat dog". During testing, unique sequences of role-filler pairs are presented that have *not* been in the same sequence together, for example: "apple bat dog". In this approach, "apple", "bat", "cat", and "dog" had been trained in every role. However, while "apple" and "dog" had *not* been in the same sequence during training, they *were* together in the testing sequence.
- **Full Combinatorial (FC)** – During training, not all fillers have been used in every role. For instance, if the fillers are "apple", "bat", "cat", and "dog" and the roles are their respective ordering in the sequence, during training the model may have been presented with "dog bat cat", "dog apple cat", and "dog bat apple". During testing, a sequence of role-filler pairs is presented to the model where a filler is in a role it had not experienced during testing: for example, "apple bat cat". Thus, "apple" was tested in the first role but this filler had never been in the first role during training.

Since the model utilized one-hot encodings for the output layer response tokens (each output unit corresponding to one of the possible filler words), some previous exposure to all filler words was required via *pretraining* on a simple association task prior to any of the above experiments. These trials consisted of requiring the model to respond with the same word which was immediately provided as input. The Indirection model's performance on the SG, SA, and FC tasks described above was compared with a simple recurrent network (SRN), a working memory (WM) model, and a working memory model with the addition of output-gating mechanisms (WMO) but lacking indirection mechanisms. The generalization abilities of the Indirection model were found to surpass the SRN and WM on all three tasks, and performance was similar to the WMO model for the SG and SA tasks, but far superior on the FC task (Kriete et al., 2013).

Artificial Indirection Models

Since the indirection model by Kriete et al. was designed for biological plausibility, it contains many biologically plausible mechanisms which result in slower model training and testing compared to deep learning models which abstract away many of these details. Therefore, Jovanovich developed an artificial indirection model that employed Holographic Reduced Representations (HRRs) (Plate, 1995) to replace recurrent PFC layers and utilized temporal difference reinforcement learning, specifically SARSA, to replace Perceived Value Learned Value (PVLV) layers (O'Reilly, Frank, Hazy, & Watz, 2007). This model was tested on SG, SA, FC, but also on an additional task for the model to complete (Kriete et al., 2013; Jovanovich, 2017):

- **Novel Filler (NF)** – During training, some fillers are not observed in any role nor in any sequence. Some potential training examples might be: "apple bat cat", "bat apple cat", and "cat apple bat". During testing, the model is presented with "zoo bat cat". In this case, "zoo" is a *novel* filler since "zoo" is tested in the first role, but never observed in any roles during training.

Jovanovich's model performed approximately the same as the model from Kriete et al. on the three shared tasks (SG, SA, and FC) (Jovanovich, 2017). However, on the new task, NF, the performance was poor due to a *lack of pretraining*. While pretraining could have been leveraged to overcome this limitation, similarly to Kriete et al., it was seen as an unsatisfying way to meet the challenge of the NF task, since the goal is to assess how a model will perform when provided a word which it truly has never seen before. The model's internal representations were instead analyzed and the results indicated that it was holding onto and providing the correct information to the one-hot-encoded actor network for the task, but a solution to this dilemma was left for future work.

Long Short-Term Memory

Recurrent Neural Networks (RNNs) are helpful for processing sequential data (eg. text sequences) and may be viewed as an approximate form of working memory in deep learning research. They function by storing information from one time step to another through the use of hidden, internal states. This allows for an accumulation of information from the past states to impact current states (Karpathy, 2015). A problem that arises with RNNs are exploding and vanishing gradients. The former refers to the "large increase of the norm of the gradient during training" and the later to the gradient diminishing to zero (Pascanu, Mikolov, & Bengio, 2013). To surmount this challenge, a specific type of RNN, long short-term memory (LSTM) (Hochreiter & Urgen Schmidhuber, 1997), was developed. LSTMs deploy a mechanism of input, output, and forget gates that allow the model to more effectively remember useful information over time and greatly reduce the impact of the exploding and vanishing gradients problem (Goodfellow, Bengio, & Courville, 2016).

Novel Role-Filler Generalization

To partially overcome the pretraining limitations observed by Jovanovich, Mullinax utilized LSTM-based models to construct word embeddings which could be passed to an artificial indirection model with three memory stripes (Mullinax, 2020). Mullinax also replaced the SARSA algorithm, used by Jovanovich, with Q-Learning to operate the input and output gates that determined how a filler would be stored. However, HRRs were still used to encode role information. The model was constructed in a nested fashion where an *outer* LSTM (OL) encoder-decoder was used to learn word embeddings from character-level tokens, and an *inner* artificial indirection (IND) model would utilize these learned embeddings in place of the HRRs used for fillers in prior work (Cho et al., 2014; Sutskever, Vinyals, & Le, 2014).

Encoder-decoder models built from LSTM layers demonstrate standard generalization, SG, capabilities. Compared with the works above, this approach allowed for pretraining of the OL component in a more realistic manner and one that could fulfill the requirements of the NF task at the sentential-level. For example, pretraining the outer model on the fillers "boy" and "cat" would allow the outer model to successfully encode and decode the filler "bat". This manner of pretraining is analagous to our learning experience of spoken language as well, where the individual letters above correspond to individual phonemes. Even novel words can be properly encoded and decoded by this *outer*, OL, component, so long as they consist of letters (or phonemes) with which the model is already familiar. Therefore, word-level embeddings constructed by the outer component, OL, were more flexible than the one-hot or HRR encodings used in the prior approaches, leaving the more difficult task of examining the meaning of the novel filler in its sentential role up to the *inner* component, IND. Thus, standard generalization is used to encode whole word representation for fillers (OL), but these representations can then be referenced using indirection to avoid confusion when processed through the inner model (IND). Therefore, we refer to this approach as the OL/IND model.

The OL/IND model was compared to a nested model which used an outer LSTM (OL) encoder-decoder and inner LSTM (IL) encoder-decoder: OL/IL (see Figure 2). Critically, the outer component for both of these models was the same and was pretrained as explained above. However, the inner component of the OL/IL model was an encoder-decoder model constructed using LSTM layers, meaning it lacked any indirection capabilities and is the deep learning-equivalent of the WMO model employed by Kriete et al. above. Both models were trained and tested with three, five letter-long, fillers (agent, role, and patient) using all four task types described above: SG, SA, FC, and NF. The performance of the models were evaluated by comparing letter-level and word-level accuracy (see eqns. (2)–(3)). Letter-level accuracy measured the percentage of correct letters in the correct positions, and word-level accuracy quantified the percentage of correct words produced in the correct positions. The latter metric necessitates that each entire words must be spelled correctly to qualify as an accurate response, but letter-level accuracy can discern partially correct responses. The results indicated that the OL/IND model could learn to generalize perfectly across all four tasks using the realistic pretraining regime described above. However, the OL/IL model only performed well on the SG and SA tasks. Given prior research, it was not surprising that the OL/IL model failed to learn the NF task well, but this model actually performed even *worse on FC*. This was a somewhat surprising result, and suggests that some form of representational interference is preventing the OL/IL model from performing well on the FC task as prior work with one-hot and HRR encodings had suggested. Even though the OL/IND model showed better performance after training, it's reliance on reinforcement learning meant significantly longer training times.

Transformers

Transformers (T) are a neural network architecture first introduced in a 2017 paper titled "Attention Is All You Need" (Vaswani et al., 2017). One remarkable part of the Transformer architecture is its self-attention mechanism which allows the network to focus on relevant input features by weighing the relations of distinct components of the input against one-another. By comparing all tokens to all other tokens, the model is better able to understand long-range dependencies. This is because, unlike SRNs and LSTMs, Transformers can process entire sequences of inputs simultaneously which precludes any bias due to sequential processing and allows for faster, feed-forward training. Empirically, Transformers have performed better than other models on tasks such as English-to-German translation (Vaswani et al., 2017).

Emergent Symbol Binding Network

Since traditional deep neural networks have difficulties inferring rules from high dimensional data, Webb et al. developed a model known as Emergent Symbol Binding Network (ESBN) that can perform a simple form of indirection (Webb, Sinha, & Cohen, 2021). ESBN was trained and tested on four tasks that involved learning abstract rules from images: same/different discrimination, relational match-to-sample, distribution-of-three, and identity rules (Webb et al., 2021). Several deep learning architectures, including most importantly the Transformer, were compared with the performance of ESBN. The only clear advantage that ESBN had over other approaches was that it did not require as many training examples to learn the tasks. While this is clearly advantageous, these results suggest that the Transformer architecture is sufficient for performing indirection and symbol binding given sufficient data. However, it is not clear whether this was truly the case, or if the tasks were simply not challenging enough to delineate differences in performance between the tested architectures.

Methods

With the recent development and subsequent success of Transformers, the task still stands to explore their strengths as well as their shortcomings. While the Transformer architecture has passed several abstraction-based challenges (Vaswani et al., 2017) and indirection-based challenges (Webb et al., 2021), previous indirection work involving the SG, SA, FC, and NF data sets (Jovanovich, 2017; Mullinax, 2020) suggests that *these tasks may be better suited to discriminate between the capabilities of the LSTM, Transformer, and IND approaches*. Additionally, Transformer models can be viewed as a replacement for RNN/SRN/LSTM components, and therefore either the inner and/or outer LSTM components in prior models. By replacing the LSTM components of the OL/IL models with transformer components (T), we hope to observe what advantages the Transformer architecture might provide. This provides a framework for developing and testing five different combinations of inner/outer components: OL/IND, OL/IL, OL/IT, OT/IL and OT/IT. All outer components can be pretrained as described above, and all inner components can then be trained/tested across the four tasks using the embeddings produced by the outer components. An example of the encoding/decoding process is illustrated in Figure 1. In this way, we hope to expose any generalization distinctions to be made among the different models.

Data Sets

The goal for all models described in the subsequent sections is the reading and reproduction of three-word sentences fed into each of the models. There are two principal classes of data sets on which the models are trained. First is the *pretraining corpus* which was the same across all models and tasks; it consists of ten thousand 5 character-long lowercase ASCII words or *fillers*. Another 10,000 word-long *pretesting corpus* of different fillers was used for testing (SG constraints as described in the Background section). All *outer* model components are trained and tested on the two corpi, respectively, until reaching 100% accuracy on the pretesting corpus. Component weights are fixed after training for these (outer) components, so that they are only used to either embed each separate word in a sentence for presentation to the inner model component, or to decode the word representations produced by the output layers of the inner model. The second class of data sets employed are the training/testing materials for the *inner* model components which differ depending on the task: SG, SA, FC or NF. Every *training set* for each of the four tasks consists of two hundred three-word *sentences*, each sentence being composed of words from the *pretesting corpus* above. Therefore, for remainder of this work, we refer to the *pretesting corpus* as simply the *corpus* since the pretraining corpus is technically not used for any of the inner component tasks. This procedure therefore insists on using words that the outer component has never seen before to train the inner components. The *testing set* for each task also consists of words from the corpus, but obeying the rules of the respective task. Note that an important distinction occurs in the NF task: some fillers from the corpus are never observed during model training, but are used during testing.

Models and Training Parameters

All models were created using Tensorflow/Keras [`ver. 2.5.0`]. Additionally, all models were developed with both a coupled and decoupled version to either provide teacher-forcing (for training) or remove teacher-forcing (for testing), respectively. When encoding and decoding tokens, the components (both outer and inner) are supplied with a start and stop token, allowing for potentially variable word/sentence lengths; though in this work each word was always composed of five letters (for the outer components) and each sentence was always composed of three words (for the inner components). The hidden states of the encoder are passed to initialize the states of the decoder. While training, a coupled version of these components is used, and during testing the decoupled version of the component is used wherein the hidden states are passed through separate input layers to allow separation between the encoder and decoder for testing *without teacher forcing*. Instead of decoupling encoder-decoder components, masking is another common method for removing teacher forcing and allowing for variable length inputs. However, our pretraining approach requires the outer components to be decoupled into a separate encoder and decoder for embedding and decoding, respectively. Therefore, we utilized consistent modeling efforts across both inner and outer components for parsimony.

The first model developed was the nested inner/outer LSTM (OL/IL) model which used an LSTM module to encode and decode fillers from the corpus into vector representations and then another LSTM to take three filler vectors and amalgamate them into a representation for the entire sentence. These are then decoded back into human-readable characters by the outer component decoder. Key hyperparameters involved are the dimension size of the OL into which each filler is encoded, the dimension size of the IL wherein the three filler vectors are combined into a vector representation of the entire sentence. This model was then trained and fitted according to the parameters shown in Table 1. The Adam optimizer was used and updated using mean squared error (MSE) and binary crossentropy (BCE) loss functions for the embedding and start/stop token layers, respectively. The training parameters for this model are given in Table 1.

The second model developed consisted of an outer LSTM model wrapped around an inner Transformer (OL/IT). Both the inner and outer components performed identical functions as in the OL/IL model. In addition to replacing the IL with the IT, the necessary addition of a position embedding layer was added between the OL and before the transformer block. Details in the training regimen and model constructions are found in Table 1, and as above the Adam optimizer with MSE and BCE loss functions for the embedding and start/stop token layers, respectively, were used to train.

In the third model, the roles are reversed and a Transformer is used as the outer model which surrounds an inner LSTM (OT/IL). The OT's encoder and decoder utilizes a masked position embedding prior to the transformer block itself. Unlike the previous models, this was trained using the Nadam optimizer with MSE and BCE loss functions for the embedding and start/stop token layers, respectively. Further details of the model and training scheme are shown in Table 1.

The last model developed was the nested Transformer model (OT/IT). Transformer blocks, each proceeded by a masked position embedding layer, are used to encode and decode both the corpus and the vector embeddings created by the outer encoder. Training used the Adam optimizer updated using MSE and BCE loss functions for the embedding and start/stop token layers, respectively. Further details of the model and training scheme are shown in Table 1.

A fifth model, used for state-of-the-art comparison, was the outer LSTM and inner Indirection model (OL/IND) developed by (Mullinax, 2020) and described in the Background section above. For further details on the model's construction and training/testing regimen see (Mullinax, 2020).

Table 1: **Key Model and Training Parameters**

Name	Value	Description
OL/IL		
Outer size	100	Dimension size for corpus embeddings
Hidden size	300	Dimension size for embedding of sentence
α	0.001	Learning rate for Adam optimizer
n	1600	Number of epochs
k	100	Batch size
OL/IL - Int. Embed.		
Outer size	64 or 256	Dimension size for corpus embeddings
Hidden size	300	Dimension size for embedding of sentence
α	0.001	Learning rate for Adam optimizer
n	1600	Number of epochs
k	100	Batch size
OL/IT		
Outer size	100	Dimension size for corpus embeddings
Num_heads	4	Number of attention heads in each transformer block
ff_dim	4	Hidden layer size in feed forward network in transformer
rate	0.1	Dropout rate for transformer
α	0.001	Learning rate for Adam optimizer
n	1600	Number of epochs
k	100	Batch size
OT/IL		
Outer size	300	Dimension size for corpus embeddings
Hidden size	300	Dimension size for embedding of sentence
Num_heads	4	Number of attention heads in each transformer block
ff_dim	4	Hidden layer size in feed forward network in transformer
rate	0.1	Dropout rate for transformer
α	0.001	Learning rate for Nadam optimizer
n	40	Number of epochs
k	25	Batch size
OT/IT		
Outer size	300	Dimension size for corpus embeddings
Outer Num_heads	32	Number of attentions heads in outer transformer block
Inner Num_heads	4	Number of attention heads in inner transformer block
embed_dim	128	Embedding size for each token
Inner ff_dim	32	Hidden layer size in feed forward network in inner transformer
rate	0.1	Dropout rate for both inner and outer transformers
α	0.001	Learning rate for Adam optimizer
n	250	Number of epochs
k	50	Batch size

Training, Testing and Evaluation

Before evaluating performance, a hyperparameter search for each model was performed manually until a set of values yielded consistent and high-performing results for that model. Certain key hyperparameters are shown in Table 1. Models were then formally evaluated by training and testing them ten times using the optimal hyperparameters found previously.

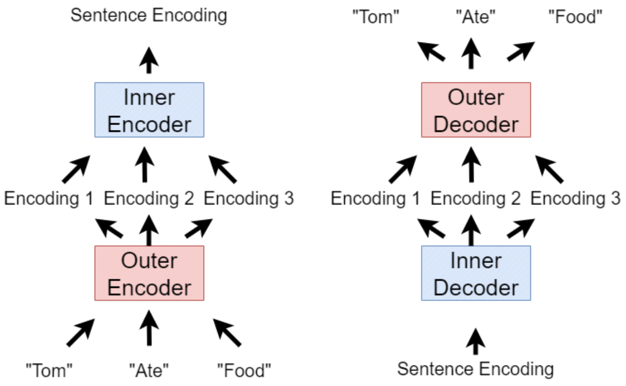

Figure 1: The figure above shows how the Inner/Outer Encoder/Decoder operate. A sentence like "Tom Ate Food" is passed to: 1) the Outer Encoder where each word is independently encoded as a series of letter token and 2) sent into the Inner Encoder which generates an encoding for the entire sentence as a series of word embeddings, and 3) this sentence encoding is then sent to the Inner Decoder which decodes the sentence into independent word embeddings, and 3) the word embeddings are sent to the Outer Decoder which decodes each one into the corresponding words (series of letters).

We employed letter-level accuracy and word-level accuracy to quantify how exactly the model was reproducing the test sentences it was fed. We defined a function $IS\_EQ(x_1, x_2)$ to compare words and characters from the model and testing data set as follows:

$$IS\_EQ(x_1, x_2) = \begin{cases} 1 & x_1 = x_2 \\ 0 & \text{otherwise} \end{cases} \tag{1}$$

Since the testing data sets are one hundred sentences long, word and letter accuracies are as follows:

$$WORD\_ACC = \frac{1}{100 \cdot 3} \Sigma_1^{100} \Sigma_1^3 IS\_EQ(\text{model's word,test word}) \tag{2}$$

$$LETTER\_ACC = \frac{1}{100 \cdot 3 \cdot 5} \Sigma_1^{100} \Sigma_1^3 \Sigma_1^5 IS\_EQ(\text{model's letter,test letter}) \tag{3}$$

Over the ten training runs, the word and letter-level accuracies for each model were obtained and the mean (+/- 1.96 standard errors) results were plotted for comparison with one another.

Results

According to the methodology described above, we trained and tested the four models we developed–OL/IL, OL/IT, OT/IL, and OT/IT–on the four tasks–SG, SA, FC and NF. Additionally we also include the results from the OL/IND model developed by (Mullinax, 2020). The results from the five models tested on the four tasks are shown in Figure 2, sorted according to the task they were tested on. One of the most apparent trends in the data is the performance gap between the SG/SA and FC/NF tasks. All models perform the SG and SA tasks with nearly 100% letter and word accuracy;

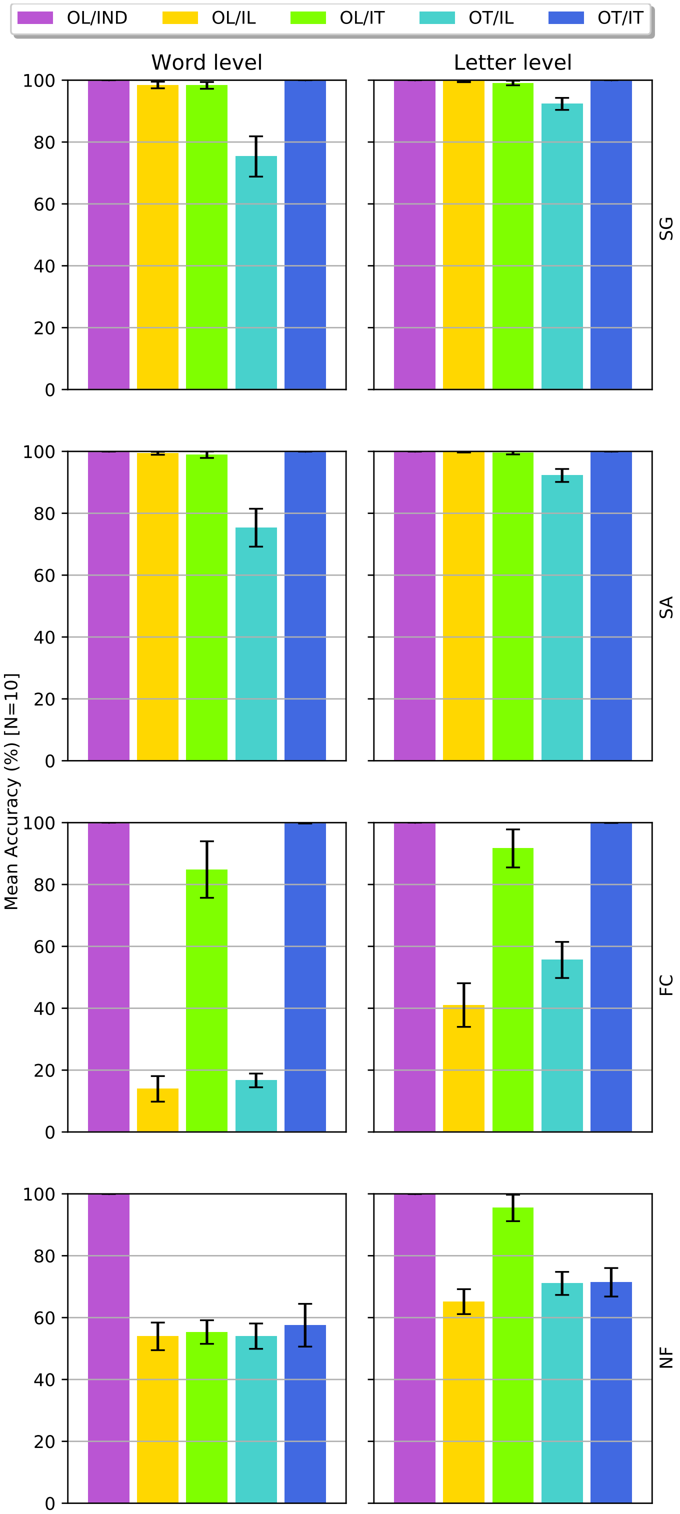

Figure 2: Plots of the letter-level and word level accuracies of the five models examined in this work. Formulas for these accuracies metrics are given in eqns. (2)–(3). All models were trained and tested N=10 times to obtain the mean (and +/- 1.96 standard errors shown by the black bars). Models are composed of inner (I) and outer (O) components: LSTM (L), Transformer (T), or indirection (IND).

however, as we move to the FC and NF tasks, the letter and word accuracies drop off significantly, most notably in the OL/IL model. The OL/IL model performed to over 90% accuracy on SG and SA for both world-level and letter-level accuracy. However, OL/IL model performance was poor on FC and NF (<40% and <65%, respectively). The OL/IL model did noticeably better on letter-level accuracy than word-level accuracy in all of these cases, and the study concludes that the model produces the *best matching* fillers experienced during training in these cases rather than the expected test fillers. Using a Transformer as the outer model (OT/IL) did not significantly increase performance: in fact, for the SG and SA tasks, the OL/IL model outperforms OT/IL, suggesting that the representations created by the OT may be, in a sense, confusing to the inner IL. However, seemingly resilient across all tasks at the letter-level were the OL/IT and OT/IT models, only showing poor word-level accuracy on the NF task. All of these models, however, were ultimately out-performed by the outer LSTM and inner indirection/working memory model (OL/IND), which scored nearly 100% across all tasks (see Figure 2). These results indicate that the Transformer better approximates indirection mechanisms than LSTM, but is not all that is needed to sufficiently handle novel fillers.

Conclusion

This work set out to evaluate certain neural network architectures' indirection and symbol binding capabilities through testing on a series of increasingly difficult sentence-processing tasks. Additionally, we examined possible synergy across the LSTM and Transformer architectures by creating nested models where the outer and inner components were interchangeable. As demonstrated in Figure 2, we see that models involving inner transformers (OL/IT, OT/IT) out-performed the model composed entirely of LSTMs (OL/IL). When employing transformers in conjunction with LSTMs, we found care must be taken as the OT/IL model performed far worse than the OL/IL model in the fairly easy SG and SA tasks. Further research into the relationships and synergies between the outer and inner encoders/decoders could help elucidate why in the OT/IL model under-performed compared to other model combinations but LSTMs were observed to be more sensitive to representational complexity than Transformers. Based on the OT/IT's 100% scores on the SG, SA and FC tasks, we confirm that the Transformer does possess superior abstraction capabilities compared to LSTM. However, what it does not possess is indirection or symbol binding capabilities, as evidenced by the clear disparity between the OL/IND model and the four others in the NF task. However, long training times are required for OL/IND models since they use reinforcement instead of supervised learning. More research is clearly needed to elucidate why the Transformer fails to perform indirection; yet it is certain that more than attention is needed to understand the unknown.

Acknowledgments

We thank three anonymous reviewers for their helpful feedback in improving the clarity of the manuscript. This work was funded by NSF Grant #1757493 (Computational Modeling and Simulation in Applied Science) at Middle Tennessee State University.

References

Alexander, G. E., DeLong, M. R., & Strick, P. L. (1986). Parallel organization of functionally segregated circuits linking basal ganglia and cortex. *Annual Review of Neuroscience*, *9*, 357-381. doi: 10.1146/annurev.ne.09.030186.002041

Cho, K., van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., & Bengio, Y. (2014). Learning phrase representations using RNN encoder–decoder for statistical machine translation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)* (pp. 1724–1734). doi: 10.3115/v1/D14-1179

Elston, G. N., Benavides-Piccione, R., Elston, A., Manger, P. R., & Defelipe, J. (2011, 2). Pyramidal cells in prefrontal cortex of primates: Marked differences in neuronal structure among species. *Frontiers in Neuroanatomy*, 1-17. doi: 10.3389/fnana.2011.00002

Goodfellow, I. J., Bengio, Y., & Courville, A. (2016). *Deep learning*. Cambridge, MA, USA: MIT Press. (http://www.deeplearningbook.org)

Hochreiter, S., & Urgen Schmidhuber, J. (1997). Long Short-Term Memory. *Neural Computation*, *9*(8), 1735–1780. doi: 10.1162/neco.1997.9.8.1735

Jovanovich, M. P. (2017). *Biologically inspired task abstraction and generalization models of working memory* Middle Tennessee State University. Retrieved from http://jewlscholar.mtsu.edu/xmlui/handle/mtsu/5561

Karpathy, A. (2015). The unreasonable effectiveness of recurrent neural networks. Retrieved from http://karpathy.github.io/2015/05/21/rnn-effectiveness

Kriete, T., Noelle, D. C., Cohen, J. D., & O'Reilly, R. C. (2013). Indirection and symbol-like processing in the prefrontal cortex and basal ganglia. *Proceedings of the National Academy of Sciences of the United States of America*, *110*. doi: 10.1073/pnas.1303547110

Mullinax, C. B. (2020). *Novel Role Filler Generalization for Recurrent Neural Networks Using Working Memory-Based Indirection* Middle Tennessee State University. Retrieved from https://jewlscholar.mtsu.edu/items/e90deed2-fc9e-4eb1-8305-5b4747bc4394

O'Reilly, R. C., Frank, M. J., Hazy, T. E., & Watz, B. (2007). *PVLV: The Primary Value and Learned Value Pavlovian Learning Algorithm.* (Vol. 121) (No. 1). American Psychological Association. doi: 10.1037/0735-7044.121.1.31

Pascanu, R., Mikolov, T., & Bengio, Y. (2013). On the difficulty of training recurrent neural networks. In *Proceedings of the 30th international conference on international conference on machine learning* (p. III–1310–III–1318). JMLR.org.

Plate, T. A. (1995). Holographic reduced representations. *IEEE Transactions on Neural Networks*, *6*(3), 623–641. doi: 10.1109/72.377968

Sutskever, I., Vinyals, O., & Le, Q. V. (2014). Sequence to sequence learning with neural networks. In *Proceedings of the 27th international conference on neural information processing systems - volume 2* (p. 3104–3112). Cambridge, MA, USA: MIT Press.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., ... Polosukhin, I. (2017). Attention is all you need. In *Proceedings of the 31st international conference on neural information processing systems* (p. 6000–6010). Red Hook, NY, USA: Curran Associates Inc.

Webb, T. W., Sinha, I., & Cohen, J. (2021). Emergent symbols through binding in external memory. In *Proceedings of the 9th international conference on learning representations.* (https://arxiv.org/abs/2012.14601)