The Importance of Being Recurrent for Modeling Hierarchical Structure

Ke Tran[1] **Arianna Bisazza**[2] **Christof Monz**[1]
[1] Informatics Institute, University of Amsterdam
[2] Leiden Institute of Advanced Computer Science, Leiden University
{m.k.tran,c.monz}@uva.nl a.bisazza@liacs.leidenuniv.nl

Abstract

Recent work has shown that recurrent neural networks (RNNs) can implicitly capture and exploit hierarchical information when trained to solve common natural language processing tasks (Blevins et al., 2018) such as language modeling (Linzen et al., 2016; Gulordava et al., 2018) and neural machine translation (Shi et al., 2016). In contrast, the ability to model structured data with non-recurrent neural networks has received little attention despite their success in many NLP tasks (Gehring et al., 2017; Vaswani et al., 2017). In this work, we compare the two architectures—*recurrent* versus *non-recurrent*—with respect to their ability to model hierarchical structure and find that recurrency is indeed important for this purpose. The code and data used in our experiments is available at https://github.com/ketranm/fan_vs_rnn

1 Introduction

Recurrent neural networks (RNNs), in particular Long Short-Term Memory networks (LSTMs), have become a dominant tool in natural language processing. While LSTMs appear to be a natural choice for modeling sequential data, recently a class of non-recurrent models (Gehring et al., 2017; Vaswani et al., 2017) have shown competitive performance on sequence modeling. Gehring et al. (2017) propose a fully convolutional sequence-to-sequence model that achieves state-of-the-art performance in machine translation. Vaswani et al. (2017) introduce Transformer networks that do not use any convolution or recurrent connections while obtaining the best translation performance. These non-recurrent models are appealing due to their highly parallelizable computations on modern GPUs. But do they have the same ability to exploit hierarchical structures *implicitly* in comparison to RNNs? In this work, we provide a first answer to this question.

Our interest here is the ability of capturing hierarchical structure without being equipped with explicit structural representations (Bowman et al., 2015b; Tran et al., 2016; Linzen et al., 2016). We choose Transformer as a non-recurrent model to study in this paper. We refer to Transformer as Fully Attentional Network (FAN) to emphasize this characteristic. Our motivation to favor FANs over convolutional neural networks (CNNs) is that FANs always have full access to the sequence history, making them more suited for modeling long distance dependencies than CNNs. Additionally, FANs promise to be more interpretable than LSTMs by visualizing attention weights.

The rest of the paper is organized as follows: We first highlight the differences between the two architectures (§2) and introduce the two tasks (§3). Then we provide setup and results for each task (§4 and §5) and discuss our findings (§6).

2 FAN versus LSTM

Conceptually, FANs differ from LSTMs in the way they utilize the previous input to predict the next output. Figure 1 depicts the main difference in terms of computation when each model is making predictions. At time step t, a FAN can access information from all previous time steps *directly* with $\mathcal{O}(1)$ computational operations. FANs do so by employing a self-attention mechanism to compute the weighted average of all previous input representations. In contrast, LSTMs compress at each time step all previous information into a single vector *recursively* based on the current input and the previous compressed vector. By their definition, LSTMs require $\mathcal{O}(d)$ computational operations to access the information at time step $t - d$.

For the details of self-attention mechanics in FANs, we refer to the work of Vaswani et al. (2017). We now proceed to measure both models' ability to

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 4731–4736
Brussels, Belgium, October 31 - November 4, 2018. ©2018 Association for Computational Linguistics

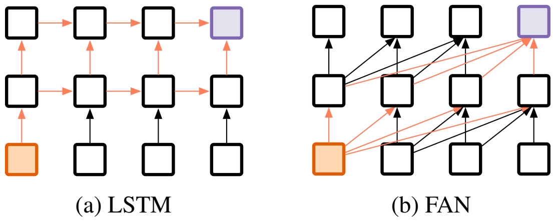


Figure 1: Diagram showing the main difference between a LSTM and a FAN. Purple boxes indicate the summarized vector at current time step t which is used to make prediction. Orange arrows indicate the information flow from a previous input to that vector.

learn hierarchical structure with a set of controlled experiments.

3 Tasks

We choose two tasks to study in this work: (1) subject-verb agreement, and (2) logical inference. The first task was proposed by Linzen et al. (2016) to test the ability of recurrent neural networks to capture syntactic dependencies in natural language. The second task was introduced by Bowman et al. (2015b) to compare tree-based recursive neural networks against sequence-based recurrent networks with respect to their ability to exploit hierarchical structures to make accurate inferences. The choice of tasks here is important to ensure that both models have to exploit hierarchical structural features (Jia and Liang, 2017).

4 Subject-Verb Agreement

Linzen et al. (2016) propose the task of predicting number agreement between subject and verb in naturally occurring English sentences as a proxy for the ability of LSTMs to capture hierarchical structure in natural language. We use the dataset provided by Linzen et al. (2016) and follow their experimental protocol of training each model using either (a) a general language model, i.e., next word prediction objective, and (b) an explicit supervision objective, i.e., predicting the number of the verb given its sentence history. Table 1 illustrates the training and testing conditions of the task.

Data: Following the original setting, we take 10% of the data for training, 1% for validation, and the rest for testing. The vocabulary consists of the 10k most frequent words, while the remaining words are replaced by their part-of-speech.

Table 1: Examples of training and test conditions for the two subject-verb agreement subtasks. The full input sentence is "The **keys** to the cabinet **are** on the table" where verb and subject are bold and intervening nouns are underlined.

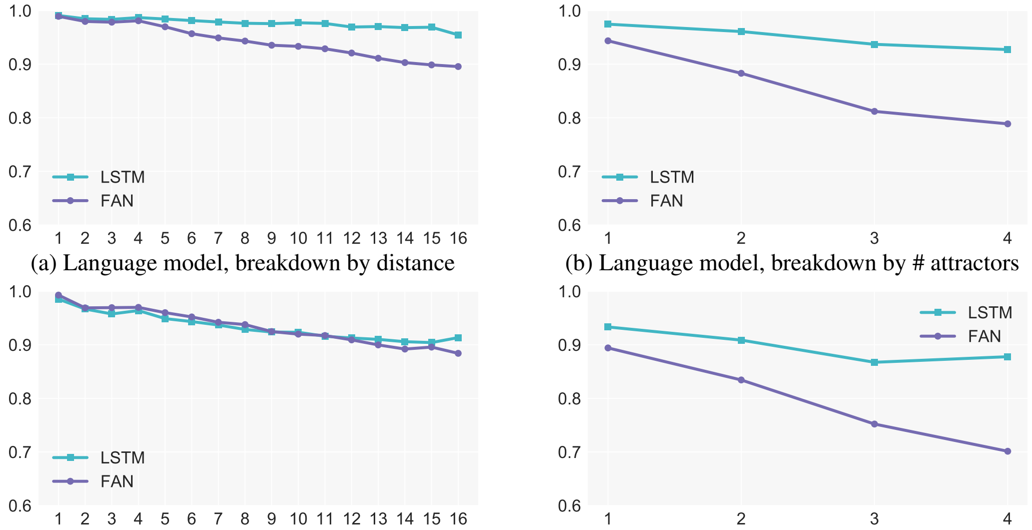
	Input	Train	Test
(a)	the keys to the cabinet	are	p(are) $>$ p(is)?
(b)	the keys to the cabinet	plural	plural/singular?

Hyperparameters: To allow for a fair comparison, we find the best configuration for each model by running a grid search over the following hyperparameters: number of layers in $\{2, 3, 4\}$, dropout rate in $\{0.2, 0.3, 0.5\}$, embedding size and number of hidden units in $\{128, 256, 512\}$, number of heads (for FAN) in $\{2, 4\}$, and learning rate in $\{0.00001, 0.0001, 0.001\}$. The weights of the word embeddings and output layer are shared (Inan et al., 2017; Press and Wolf, 2017). Models are optimized by Adam (Kingma and Ba, 2015).

We first assess whether the LSTM and FAN models trained with respect to the language model objective assign higher probabilities to the correctly inflected verbs. As shown in Figures 2a and 2b, both models achieve high accuracies for this task, but LSTMs consistently outperform FANs. Moreover, LSTMs are clearly more robust than FANs with respect to task difficulty, measured both in terms of word distance and number of agreement attractors[1] between subject and verb. Christiansen and Chater (2016); Cornish et al. (2017) have argued that human memory limitations give rise to important characteristics of natural language, including its hierarchical structure. Similarly, our experiments suggest that, by compressing the history into a single vector before making predictions, LSTMs are forced to better learn the input structure. On the other hand, despite having direct access to all words in their history, FANs are less capable of detecting the verb's subject. We note that the validation perplexities of the LSTM and FAN are 67.06 and 69.14, respectively.

Secondly, we evaluate FAN and LSTM models explicitly trained to predict the verb number (Figures 2c and 2d). Again, we observe that LSTMs consistently outperform FANs. This is a particularly interesting result since the self-attention mechanism in FANs connects two words in any po-

[1] Agreement attractors are intervening nouns with the opposite number from the subject.



(a) Language model, breakdown by distance

(b) Language model, breakdown by # attractors

(c) Number prediction, breakdown by distance

(d) Number prediction, breakdown by # attractors

Figure 2: Results of subject-verb agreement with different training objectives.

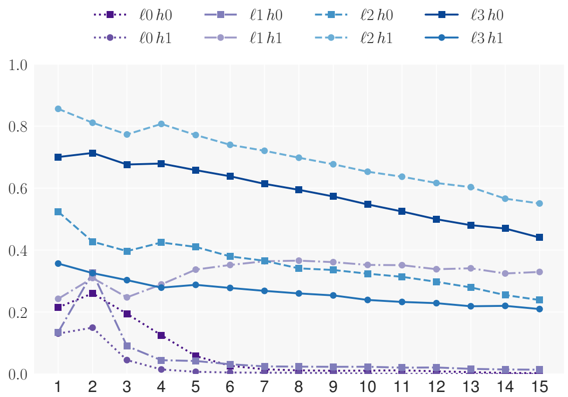


Figure 3: Proportion of times the subject is the most attended word by different heads at different layers ($\ell 3$ is the highest layer). Only cases where the model made a correct prediction are shown.

sition with a $\mathcal{O}(1)$ number of executed operations, whereas RNNs require more recurrent operations. Despite this apparent advantage of FANs, the performance gap between FANs and LSTMs increases with the distance and number of attractors.[2]

To gain further insights into our results, we examine the attention weights computed by FANs during verb-number prediction (supervised objective). Specifically, for each attention head at each layer of the FAN, we compute the percentage of times the subject is the most attended word among all words in the history. Figure 3 shows the results for all cases where the model made the correct prediction. While it is hard to interpret the exact role of attention for different heads and at different layers, we find that some of the attention heads at the higher layers ($\ell 2\ h1$, $\ell 3\ h0$) frequently point to the subject with an accuracy that decreases linearly with the distance between subject and verb.

5 Logical inference

In this task, we choose the artificial language introduced by Bowman et al. (2015b). The vocabulary of this language includes six word types {*a, b, c, d, e, f*} and three logical operators {*or*, *and*, *not*}. The task consists of predicting one of seven mutually exclusive logical relations that describe the relationship between a pair of sentences: entailment ($\sqsubset$, $\sqsupset$), equivalence ($\equiv$), exhaustive and non-exhaustive contradiction ($^\wedge$, $|$), and two types of semantic independence ($\#$, $\smile$). We generate 60,000 samples[3] with the number of logical operations ranging from 1 to 12. The train/dev/test dataset ratios are set to 0.8/0.1/0.1. Here are some samples of the training data:

(d (or f)) $\sqsupset$ (f (and a))
(d (and (c (or d)))) $\#$ (not f)
(not (d (or (f (or c))))) $\sqsubset$ (not (c (and (not d))))

Why artificial data? Despite the simplicity of the

[2] We note that our LSTM results are better than those in Linzen et al. (2016). Also surprising is that the language model objective yields higher accuracies than the number prediction objective. We believe this may be due to better model optimization and to the embedding-output layer weight sharing, but we leave a thorough investigation to future work.

[3] https://github.com/sleepinyourhat/vector-entailment

language, this task is not trivial. To correctly classify logical relations, the model must learn nested structures as well as the scope of logical operators. We verify the difficulty of the task by training three bag-of-words models followed by sum/average/max-pooling. The best of the three models achieve less than 59% accuracy on the logical inference versus 77% on the Stanford Natural Language Inference (SNLI) corpus (Bowman et al., 2015a). This shows that the SNLI task can be largely solved by exploiting shallow features without understanding the underlying linguistic structures, which has also been pointed out by recent work (Glockner et al., 2018; Gururangan et al., 2018).

Concurrently to our work Evans et al. (2018) proposed an alternative data set for logical inference and also found that a FAN model underperformed various other architectures including LSTMs.

5.1 Models

We follow the general architecture proposed in (Bowman et al., 2015b): Premise and hypothesis sentences are encoded by fixed-size vectors. These two vectors are then concatenated and fed to a 3-layer feed-forward neural network with ReLU non-linearities to perform 7-way classification of the logical relation.

The LSTM architecture used in this experiment is similar to that of Bowman et al. (2015b). We simply take the last hidden state of the top LSTM layer as a fixed-size vector representation of the sentence. Here, we use a 2-layer LSTM with skip connections. The FAN maps a sentence x of length n to $\mathbf{H} = [\mathbf{h}_1, \ldots, \mathbf{h}_n] \in \mathbb{R}^{d \times n}$. To obtain a fixed-size representation $\mathbf{z}$, we use a self-attention layer with two trainable queries $\mathbf{q}_1, \mathbf{q}_2 \in \mathbb{R}^{1 \times d}$:

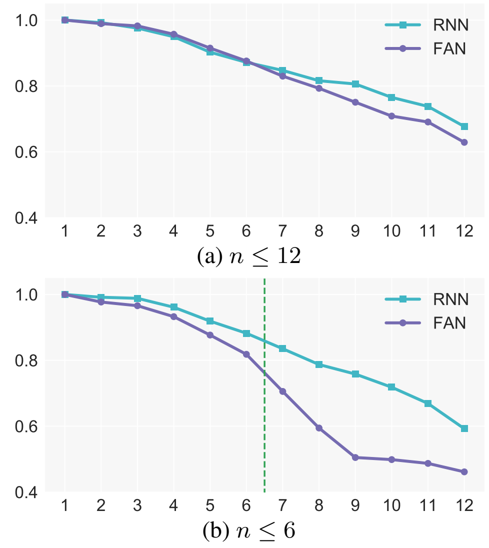
$$\mathbf{z}_i = \text{softmax}\left(\frac{\mathbf{q}_i\mathbf{H}}{\sqrt{d}}\right)\mathbf{H}^\top \quad i \in \{1, 2\}$$

$$\mathbf{z} = [\mathbf{z}_1, \mathbf{z}_2]$$

We find the best hyperparameters for each model by running a grid search as explained in §4.

5.2 Results

Following the experimental protocol of Bowman et al. (2015b), the data is divided into 13 bins based on the number of logical operators. Both FANs and LSTMs are trained on samples with at most n logical operators and tested on all bins. Figure 4 shows the result of the experiments with $n \leq 6$ and $n \leq 12$. We see that FANs and LSTMs perform similarly when trained on the whole dataset (Figure 4a). However when trained on a subset of the data (Figure 4b), LSTMs obtain better accuracies on similar examples ($n \leq 6$) and generalize better on longer examples ($6 < n \leq 12$).



(a) $n \leq 12$

(b) $n \leq 6$

Figure 4: Results of logical inference when training on all data (a) or only on samples with at most n logical operators (b).

6 Discussion and Conclusion

We have compared a recurrent architecture (LSTM) to a non-recurrent one (FAN) with respect to the ability of capturing the underlying hierarchical structure of sequential data. Our experiments show that LSTMs slightly but consistently outperform FANs. We found that LSTMs are notably more robust with respect to the presence of misleading features in the agreement task, whether trained with explicit supervision or with a general language model objective. Secondly, we found that LSTMs generalize better than FANs to longer sequences in a logical inference task. These findings suggest that recurrency is a key model property which should not be sacrificed for efficiency when hierarchical structure matters for the task.

This does not imply that LSTMs should always be preferred over non-recurrent architectures. In fact, both FAN- and CNN-based networks have proved to perform comparably or better than LSTM-based ones on a very complex task like machine translation (Gehring et al., 2017; Vaswani et al., 2017). Nevertheless, we believe that the ability of capturing hierarchical information in sequential data remains a fundamental need for building intelligent systems that can understand and process language. Thus we hope that our insights will be useful towards building the next generation of neural networks.

Acknowledgments

This research was funded in part by the Netherlands Organization for Scientific Research (NWO) under project numbers 639.022.213, 612.001.218, and 639.021.646.

References

Terra Blevins, Omer Levy, and Luke Zettlemoyer. 2018. Deep rnns encode soft hierarchical syntax. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 14–19, Melbourne, Australia. Association for Computational Linguistics.

Samuel R. Bowman, Gabor Angeli, Christopher Potts, and Christopher D. Manning. 2015a. A large annotated corpus for learning natural language inference. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 632–642, Lisbon, Portugal. Association for Computational Linguistics.

Samuel R. Bowman, Christopher D. Manning, and Christopher Potts. 2015b. Tree-structured composition in neural networks without tree-structured architectures. In *Proceedings of the 2015th International Conference on Cognitive Computation: Integrating Neural and Symbolic Approaches - Volume 1583*, COCO'15, pages 37–42.

Morten H. Christiansen and Nick Chater. 2016. The now-or-never bottleneck: A fundamental constraint on language. *Behavioral and Brain Sciences*, 39.

Hannah Cornish, Rick Dale, Simon Kirby, and Morten H Christiansen. 2017. Sequence memory constraints give rise to language-like structure through iterated learning. *PloS one*, 12(1):e0168532.

Richard Evans, David Saxton, David Amos, Pushmeet Kohli, and Edward Grefenstette. 2018. Can neural networks understand logical entailment? In *ICLR*.

Jonas Gehring, Michael Auli, David Grangier, Denis Yarats, and Yann N. Dauphin. 2017. Convolutional sequence to sequence learning. In *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 1243–1252, International Convention Centre, Sydney, Australia. PMLR.

Max Glockner, Vered Shwartz, and Yoav Goldberg. 2018. Breaking nli systems with sentences that require simple lexical inferences. In *The 56th Annual Meeting of the Association for Computational Linguistics (ACL)*, Melbourne, Australia.

Kristina Gulordava, Piotr Bojanowski, Edouard Grave, Tal Linzen, and Marco Baroni. 2018. Colorless green recurrent networks dream hierarchically. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1195–1205. Association for Computational Linguistics.

Suchin Gururangan, Swabha Swayamdipta, Omer Levy, Roy Schwartz, Samuel Bowman, and Noah A. Smith. 2018. Annotation artifacts in natural language inference data. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 107–112, New Orleans, Louisiana. Association for Computational Linguistics.

Hakan Inan, Khashayar Khosravi, and Richard Socher. 2017. Tying word vectors and word classifiers: A loss framework for language modeling. *ICLR*.

Robin Jia and Percy Liang. 2017. Adversarial examples for evaluating reading comprehension systems. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2021–2031, Copenhagen, Denmark. Association for Computational Linguistics.

Diederik Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *Proceedings of ICLR*, San Diego, CA, USA.

Tal Linzen, Emmanuel Dupoux, and Yoav Goldberg. 2016. Assessing the ability of lstms to learn syntax-sensitive dependencies. *Transactions of the Association for Computational Linguistics*, 4:521–535.

Ofir Press and Lior Wolf. 2017. Using the output embedding to improve language models. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, pages 157–163. Association for Computational Linguistics.

Xing Shi, Inkit Padhi, and Kevin Knight. 2016. Does string-based neural mt learn source syntax? In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1526–1534, Austin, Texas. Association for Computational Linguistics.

Ke Tran, Arianna Bisazza, and Christof Monz. 2016. Recurrent memory networks for language modeling. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 321–331, San Diego, California. Association for Computational Linguistics.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 6000–6010. Curran Associates, Inc.