One Attention Head Is All You Need for Sorting Fixed-Length Lists[1]

Mateusz Bagiński **Gabin Kolly**

Abstract

We trained a single layer attention-only transformer to sort a fixed-length list of non-repeating tokens. It turned out to implement an operation consisting of looking into the unsorted list and searching for tokens that are greater than the current token, giving greater weight to the ones that are closer in the ordering. This attention pattern was clearest in transformers with one attention head, whereas increasing the number of heads led to development of more complex algorithms. We further explored how the same task was accomplished by zero-layer models as well as how varying list length, vocabulary size, and model complexity impacts the results.

Keywords: Mechanistic interpretability, ML safety

1. Introduction

The field of Mechanistic Interpretability (MI) is a discipline concerned with analyzing behaviors and capabilities of neural networks, (most notably those based on transformer architecture, Vaswani et al., 2017) and explaining their behavior in terms of low-level, learned algorithms. While capability research using transformer-based neural networks is progressing very rapidly, our understanding of how these systems achieve their performance and what they learn in the course of training is lagging behind.

We trained transformers to sort lists of integers, and then tried to understand how they accomplished this task. It turned out that a simple attention-only transformer with one layer and one attention head is sufficient to get perfect or nearly perfect accuracy. We analyzed how the model performs on different variants of the task, varying list length, vocabulary size, and possibility of repeating tokens in the list. We used mechanistic interpretability tools to analyze the cognition learned by the model. Finally, we trained smaller (zero-layer "transformers") and bigger (more attention heads, attention layers, and with MLPs) models in order to compare their performance and behavior with our base model.

2. Methods

[1] Research conducted at the Apart Research Alignment Jam #4 (Mechanistic Interpretability), 2023 (see https://itch.io/jam/mechint)

We trained simple transformer-based language models to sort fixed-length lists of tokens. The sequences used to train and test the models consisted of the special START token, the unsorted list of tokens, the special MID token, and the sorted list of tokens, in that exact order.

For the loss function, we used mean cross-entropy, calculated only for the tokens and logits at later positions corresponding to the sorted list. All models were trained until the point of convergence (defined as having achieved perfect accuracy on the held out validation set for the last 10 checkpoints) or for 20000 epochs (each epoch corresponding to one batch of training data) if no convergence occurred. We used the TransformerLens[2] library and Pytorch. Jupyter notebooks with training procedures and saved states of trained models are available in the GitHub repository associated with that project.[3]

We conducted several training runs, varying hyperparameters, such as size of vocabulary (number of "normal" tokens), list length, and model architecture. After training, we analyzed the model's performance, focusing on how it reasoned on examples it got right and the ones it got wrong. We used the CircuitsVis library[4] to analyze attention patterns.

3. Results

In this section, we describe the training runs and results. Models and Jupyter notebooks corresponding to each, can be found in the GitHub repository.

3.1. One-head, attention-only transformer

First, we trained a simple attention-only transformer with one attention head in its only attention layer to sort lists of 5 non-repeating tokens out of vocabulary of size 66 (64 normal tokens plus 2 special START and MID tokens). It achieved 100% accuracy on validation and training data. The attention head learned the following algorithm. Until the MID token, its attention patterns seemed mostly random, which should be expected, since only logits from the MID token onwards influenced the loss function. The MID token attended to the smallest token in the unsorted sequence and copied it to the following position. Attention on each token after the MID token was directed to the tokens in the unsorted list, which were bigger[5] than the current token. Attention strength was anticorrelated with distance in the ordering. Attention on the final token appeared to be distributed mostly randomly, probably because its logits were not used in the loss function. Henceforth we call the attention head displaying this pattern of attention, the **ordering head**, since it can be interpreted as representing some kind of total order over a finite set of tokens.

We performed several experiments of corrupting the sequence, by removing special tokens or injecting them somewhere inside the unsorted sequence. Removing the START token significantly decreased performance, often leading to duplicating some of the tokens, whereas injecting it made it attend selectively to tokens in the unsorted sequence

[2] https://github.com/neelnanda-io/TransformerLens
[3] https://github.com/MatthewBaggins/one-attention-head-is-all-you-need
[4] https://github.com/alan-cooney/CircuitsVis
[5] In this paper, when we refer to a token being "bigger" or "smaller" than some other token, we mean it in reference to the ordering, we want the model to learn: (0, 1, 2, …, D_VOCAB)

with strength proportional to token cardinality.[6] On the other hand, removing the MID token did not hinder the model's performance. Feeding a sequence consisting only of the START token and the unsorted tokens to the model to continue resulted in it copying one of the usual tokens and using it as the MID token, with attention pattern likewise typical for the MID token, i.e. attending to the smallest token in the unsorted sequence. Injecting the MID token later in the sequence, on the other hand, led to its distributing its attention among the unsorted tokens anti-proportionally to their size, but not exclusively to the smallest token.

To confirm generalizability of our results, we trained the model on longer sequences and greater vocabularies, which required scaling up the dimensions of the model's residual stream and attention head. Although we could not get the model to achieve similar 100% accuracy (most likely due to greater difficulty of learning total ordering over a greater set of tokens) it easily achieved about 98% or 99% of accuracy on the test set. The results described for shorter lists and smaller vocabularies were reproduced.

We also trained a model on data with lists containing some repeating tokens with similar results. The main difference was that some attention was paid to the same tokens as the previous token. This was modulated by positional embeddings: the attention from tokens at the end of the list is smaller than the tokens at the beginning, even when they have the same values. More details and examples are available in the notebooks in our Github repository.

3.2. Degenerate transformer

Further, we trained a small 0-layer transformer. Since training models of that size does not take much time, we trained it on many combinations of list length and vocabulary size. Moreover, unlike for the 1-layer transformer described in the previous section, we measured predictive accuracy per token, rather than per entire sorted list. Figure 2 presents the results, showing that accuracy increases with list length and decreases with vocabulary size.

Exploration of the trained model's behavior revealed that it developed the strategy of predicting the token that was one of the closest successors of the previous one in the learned ordering. For example, it predicted that token 1 would be followed by 2, 3, or 4. How "far" in the ordering the model "looked" to sample predictions depended on vocabulary size and list length. The exception was the predictions following the final token which were more diverse and the only constraint on them was to be greater than the previous token. In order to get information that it is at the end of a sequence, it must use positional encoding. The optimal strategy when you only have access to the previous value in the list is to predict the previous value plus 1, since statistically this is what limits the least the rest of the sequence, and so what is the most common. The optimal strategy was only adopted when we tried with a small vocabulary.

Analysis of the "internal lookup table" obtained by multiplying embedding and unembedding matrices parallels these observations. Figure 3 presents one obtained from

[6] By "token cardinality", we mean it in the sense of being bigger or smaller (see the previous footnote).

the model trained on the greatest list length and vocabulary size and figure 4 presents one for a smaller vocabulary size and list length. Clearly, when the previous token is X, the model assigns the greatest amount of probability mass to tokens X+1, X+2, X+3, etc., with probability mass slowly decreasing for bigger tokens.

Enabling repetition hindered the model's performance (compare figures 2 and 6). Internal lookup tables of two 0-layer models trained with setups differing only in whether repetition was allowed or not, differed in that the one trained on lists with allowed repetition assigned non-trivial probability to continuing by predicting itself.

3.3. Bigger models

Increasing the number of layers in a transformer enables learning of methods of composing heads in circuits capable of executing sophisticated algorithms (Elhage et al., 2021). Adding non-linear transformations inherent in MLPs further increases the model's capabilities. Thus, we might expect bigger models to develop more sophisticated methods of sorting lists.

While detailed investigation of big transformers' behavior is beyond the scope of this project, we tried increasing the number of attention layers and attention heads in the model in order to see whether better bigger models also develop ordering heads. Attention heads in single-layer multi-head attention-only transformers to some extent attended in the way described previously for ordering heads. Models with more heads in their layer often exhibited deviation from the pattern. Increasing the number of layers while keeping the number of heads per layer the same, had a similar effect. Adding an MLP, on the other hand, did not inhibit development of ordering heads.

4. Discussion and Conclusion

We have shown that a single-layer, single-head, attention-only transformer can learn to sort fixed-length lists of non-repeating elements with perfect or nearly perfect performance. To do so, it develops what we called an ordering head, which is an attention head that, given any particular token in the sorted list being produced, attends to tokens in the unsorted list, which are greater than itself with attention strength proportional to how close they are in the ordering. Enabling repeating elements in the list modifies the behavior of the ordering head such that it attends not only to tokens that are greater than the previous token, but also to its own prior instances. Zero-layer models, which do not have access to the attention mechanism, learn to rely on statistical information they encountered in the training data about which tokens are likely to follow the previous token as well as (most likely) positional encoding.

Elhage et al. (2021) pioneered in training attention-only transformers on natural language data and explaining their behavior in a mechanistic manner. They found that 0-layer attention-only transformers (consisting primarily of embedding and unembedding matrices) learn bigram statistics, i.e., given any particular token, they predict the token that has followed it most often in the training data. This is similar to what we found when we trained the same kind of models on sorting fixed-length lists without. The models predicted the next token by incrementing the current one by a small number (Section 3.2).

This can be conceptualized as a lookup table, where every token is assigned a context-independent probability distribution about tokens that are going to follow it.

In the same paper, Elhage et al. report that 1-layer attention-only transformers use a combination of bigram statistics and skip-trigrams, where an attention head uses not only the information about the previous token but also some token in the past. Again, we found similar results for ordering heads in analogous models trained on our task. Given any token, the ordering head "selects" among the tokens from the unsorted list and selects the most likely successor.

Our experiment is a case study in how simple (toy), transformer-based language models can use attention mechanisms to accomplish a simple task, such as sorting fixed-length lists of tokens, which, as far as we know, has not been investigated so far. Although single case studies most often are not groundbreaking and do not offer great theoretical insights, such marginal discoveries are likely to lead at some point to discovering laws governing intelligent behavior emerging in neural networks, similarly to how studying many species makes it possible to uncover laws governing evolution and general, recurring phenomena, such as insular dwarfism and gigantism. While our discovery was initially made using a very simple attention-only model with one attention head, our initial explorations suggest that bigger models (including the ones with MLPs) learn a similar algorithm.

5. Future Directions

Our Jupyter notebooks are available in the GitHub repository to anybody who would like to take a closer look at our experiments and analyses or use them as a starting point or inspiration for their own research. Here we detail some limitations of our work that could be addressed in future research and some ways to extend it.

We did not investigate positional encoding in depth. In the no-repetition case, the model does not need positional information to guess the right answer, so we could train a model without positional encoding to see if it can learn sorting fixed-length lists and how its performance compares to that of models with positional encoding.

Although we used lists of different lengths to train the models, each model was only trained on lists of one particular length, which probably made the task simpler. An interesting direction would be to see how training one model on lists of differing lengths changes the results. On the other hand, the algorithm learned by our models should be easily scalable to variable lengths, so we predict that this modification would not have a huge impact.

Increasing the number of attention heads and attention layers led the model to develop more sophisticated methods of sorting the list, which deviated from the simple ordering head pattern. This opens an interesting line of research. Although MLPs did not inhibit development of the ordering head, they may enter into complex interactions with attention in bigger models.

Finally, large language models, such as GPT-3 (Brown, 2020) or PaLM (Chowdhery, 2022) are known to be able to execute many tasks given in natural language, including sorting lists. While analysis of LLMs' attention patterns is much more complicated than in the case of toy models trained on one simple task, it would be interesting whether they learn similar algorithms for sorting lists.

6. References

Brown, T.B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., Agarwal, S., Herbert-Voss, A., Krueger, G., Henighan, T.J., Child, R., Ramesh, A., Ziegler, D.M., Wu, J., Winter, C., Hesse, C., Chen, M., Sigler, E., Litwin, M., Gray, S., Chess, B., Clark, J., Berner, C., McCandlish, S., Radford, A., Sutskever, I., & Amodei, D. (2020). Language Models are Few-Shot Learners. *ArXiv, abs/2005.14165*.

Chowdhery, A., Narang, S., Devlin, J., Bosma, M., Mishra, G., Roberts, A., Barham, P., Chung, H.W., Sutton, C., Gehrmann, S., Schuh, P., Shi, K., Tsvyashchenko, S., Maynez, J., Rao, A., Barnes, P., Tay, Y., Shazeer, N.M., Prabhakaran, V., Reif, E., Du, N., Hutchinson, B.C., Pope, R., Bradbury, J., Austin, J., Isard, M., Gur-Ari, G., Yin, P., Duke, T., Levskaya, A., Ghemawat, S., Dev, S., Michalewski, H., García, X., Misra, V., Robinson, K., Fedus, L., Zhou, D., Ippolito, D., Luan, D., Lim, H., Zoph, B., Spiridonov, A., Sepassi, R., Dohan, D., Agrawal, S., Omernick, M., Dai, A.M., Pillai, T.S., Pellat, M., Lewkowycz, A., Moreira, E., Child, R., Polozov, O., Lee, K., Zhou, Z., Wang, X., Saeta, B., Díaz, M., Firat, O., Catasta, M., Wei, J., Meier-Hellstern, K.S., Eck, D., Dean, J., Petrov, S., & Fiedel, N. (2022). PaLM: Scaling Language Modeling with Pathways. *ArXiv, abs/2204.02311*.

Elhage, N., Nanda, N., Olsson, C., Henighan, T., Joseph, N., Mann, B., Askell, A., Bai, Y., Chen, A., Conerly, T., DasSarma, N., Drain, D., Ganguli, D., Hatfield-Dodds, Z., Hernandez, D., Jones, A., Kernion, J., Lovitt, L., Ndousse, K., Amodei, D., Brown, T., Clark, J., Kaplan, J., McCandlish, S., and Olah, C. A mathematical framework for transformer circuits. https://transformer-circuits.pub/2021/framework/index.html, December 2021.

Vaswani, A., Shazeer, N.M., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, L., & Polosukhin, I. (2017). Attention is All you Need. *ArXiv, abs/1706.03762*.

7. Plots and Figures

Figure 1. Attention patterns for the Q-K circuit of the one-layer one-head attention-only transformer. The MID token (the rightmost column) attends to tokens with strength anti-correlated with their cardinality. The START token (the second-to-rightmost column) attends seemingly randomly. Normal tokens (other columns) attend primarily to their closest successors, although it is most concentrated for the smallest and the biggest tokens, whereas for tokens of intermediate cardinality, attention pattern is much more spread out. Moreover, there are some outliers, where a small token receives excessive attention from a bigger token.

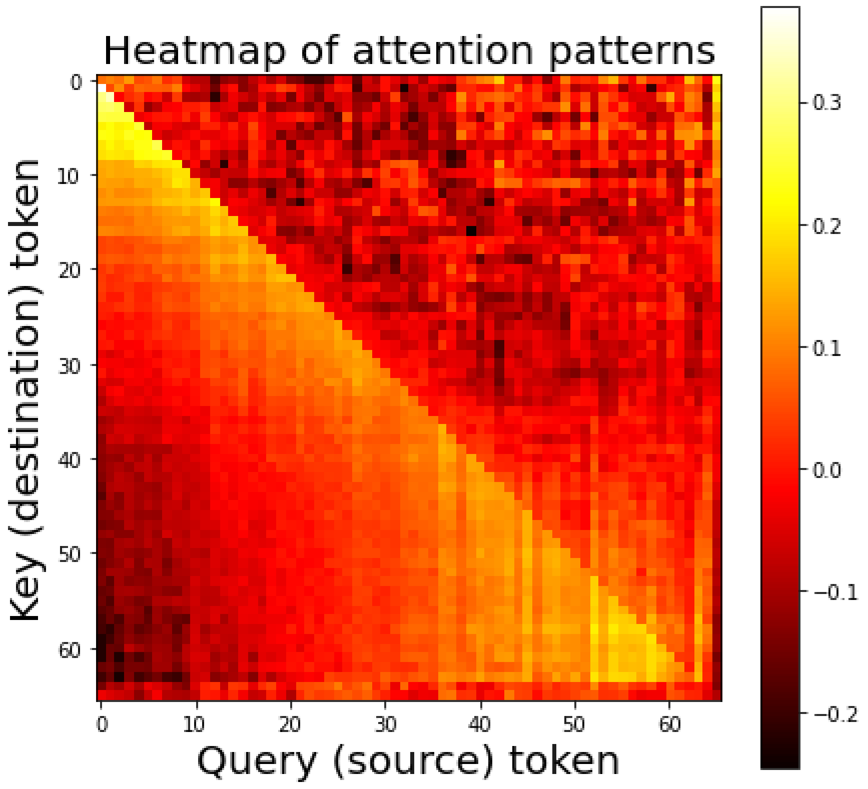
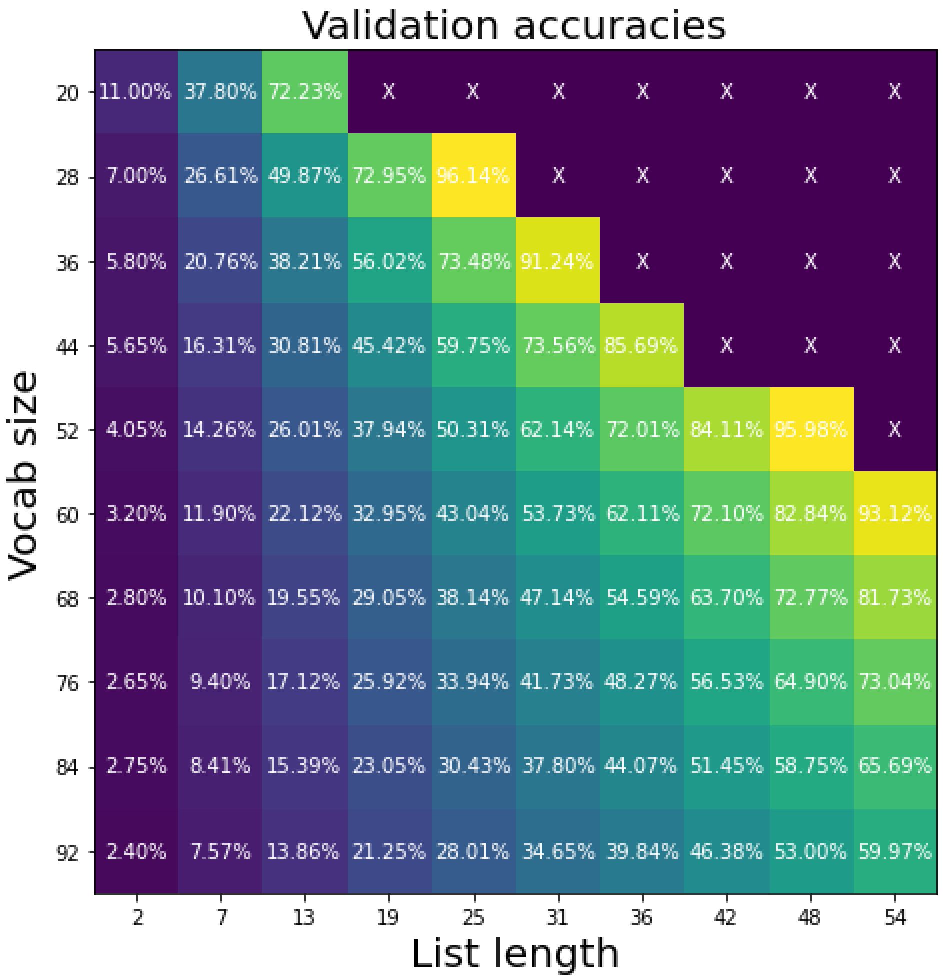


Figure 2. Final validation accuracies (per token) achieved by zero-layer transformer on particular list length and vocabulary sizes on lists of non-repeating tokens. Xs mark combinations of hyperparameters where list length was smaller than the number of normal tokens and thus a list of that length without repetitions could not be produced.



Vocab size \ List length	2	7	13	19	25	31	36	42	48	54
20	11.00%	37.80%	72.23%	X	X	X	X	X	X	X
28	7.00%	26.61%	49.87%	72.95%	96.14%	X	X	X	X	X
36	5.80%	20.76%	38.21%	56.02%	73.48%	91.24%	X	X	X	X
44	5.65%	16.31%	30.81%	45.42%	59.75%	73.56%	85.69%	X	X	X
52	4.05%	14.26%	26.01%	37.94%	50.31%	62.14%	72.01%	84.11%	95.98%	X
60	3.20%	11.90%	22.12%	32.95%	43.04%	53.73%	62.11%	72.10%	82.84%	93.12%
68	2.80%	10.10%	19.55%	29.05%	38.14%	47.14%	54.59%	63.70%	72.77%	81.73%
76	2.65%	9.40%	17.12%	25.92%	33.94%	41.73%	48.27%	56.53%	64.90%	73.04%
84	2.75%	8.41%	15.39%	23.05%	30.43%	37.80%	44.07%	51.45%	58.75%	65.69%
92	2.40%	7.57%	13.86%	21.25%	28.01%	34.65%	39.84%	46.38%	53.00%	59.97%

Figure 3. Probabilistic lookup table for the 0-layer model trained on greatest vocabulary size (92) and list length (54) without repetition. Clearly, on each token the greatest attention is paid to its closest successors, decreasing with distance.

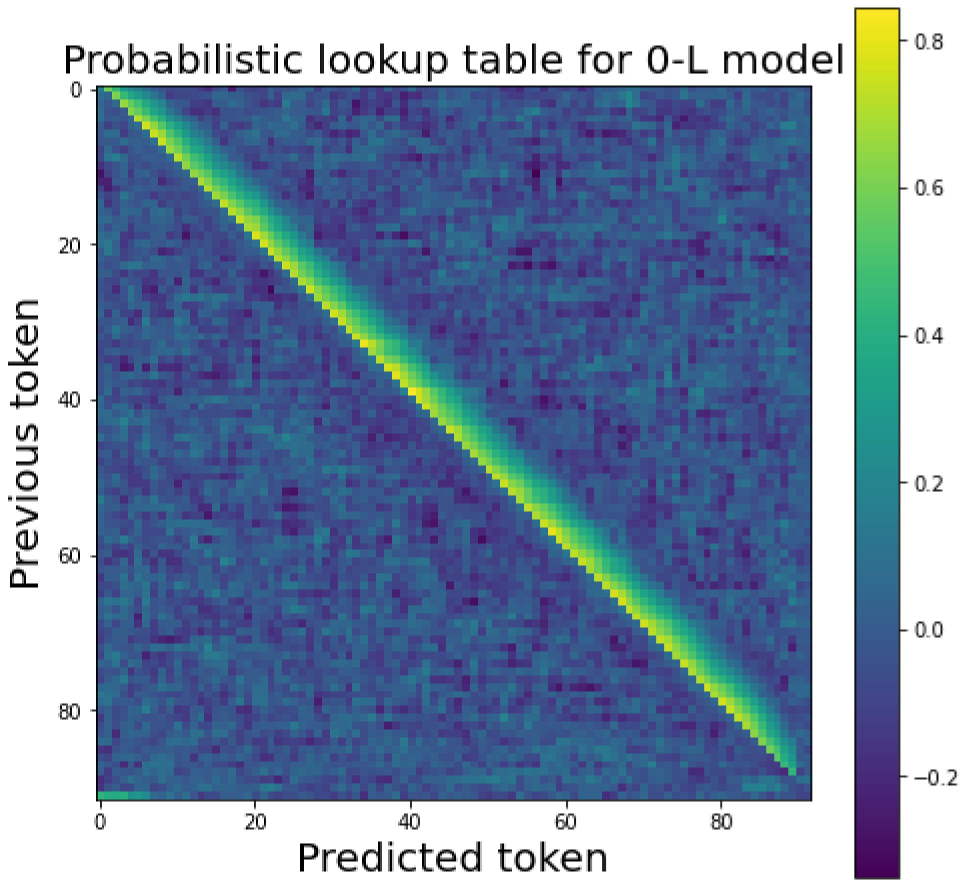


Figure 4. Probabilistic lookup table for the 0-layer model trained on vocabulary size 28 and list length 25 without repetition.

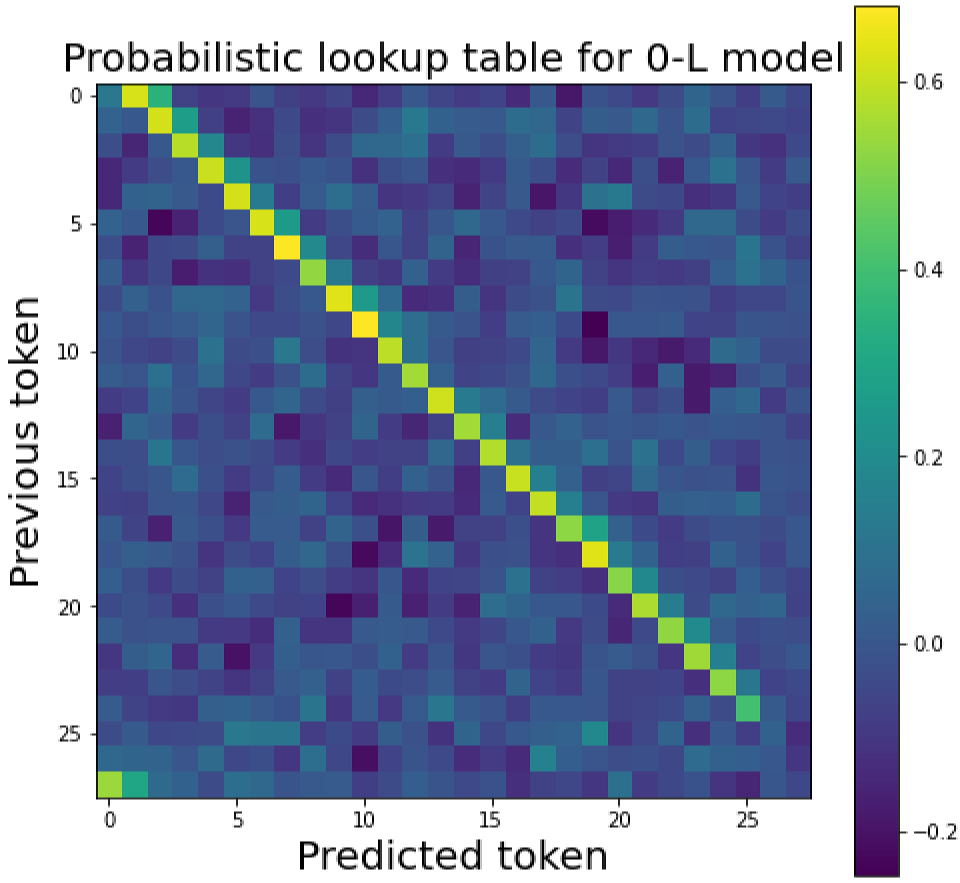


Figure 5. Statistics about token prediction for the 0-layer model trained on vocabulary size 28 and list length 25 without repetition. The results parallel those from figure 4.

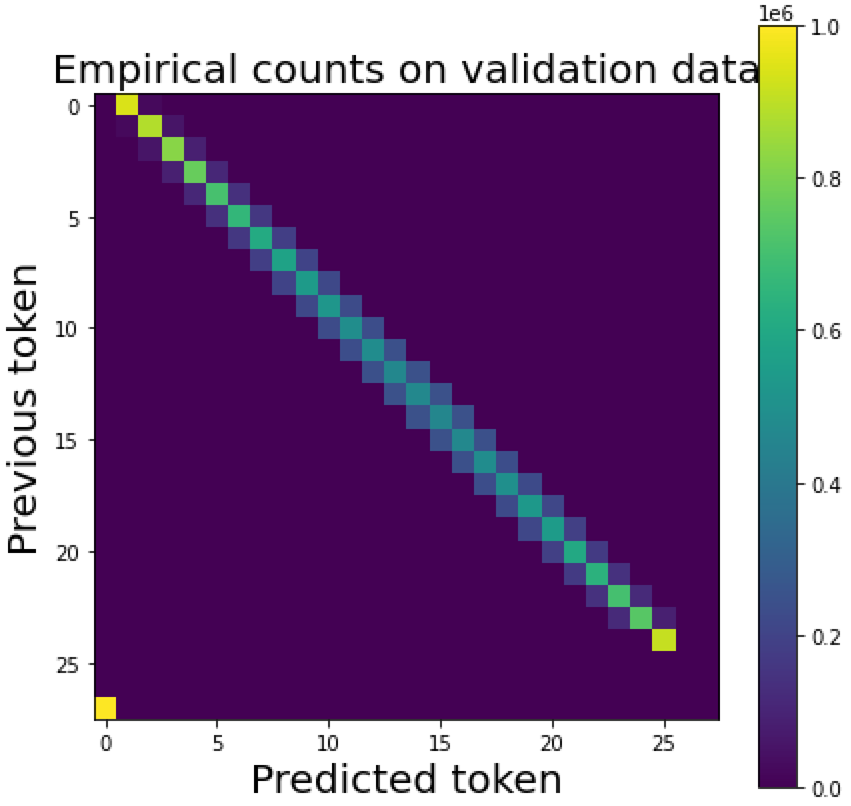
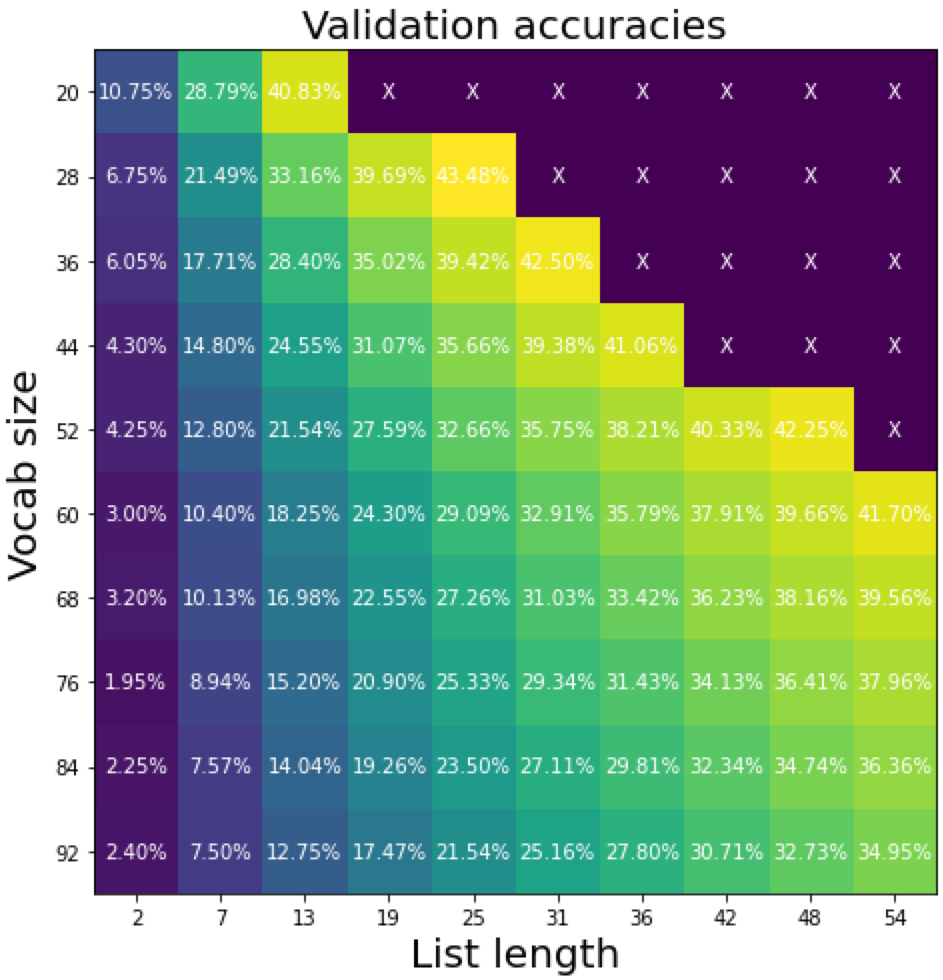


Figure 6. Final validation accuracies (per token) achieved by zero-layer transformer on particular list length and vocabulary sizes on lists of tokens with enabled repetition. The scores are visibly worse than for the case without repetition, dropping from the best case value of 95% to 42% (list length 48, vocab size 52).



	2	7	13	19	25	31	36	42	48	54
20	10.75%	28.79%	40.83%	X	X	X	X	X	X	X
28	6.75%	21.49%	33.16%	39.69%	43.48%	X	X	X	X	X
36	6.05%	17.71%	28.40%	35.02%	39.42%	42.50%	X	X	X	X
44	4.30%	14.80%	24.55%	31.07%	35.66%	39.38%	41.06%	X	X	X
52	4.25%	12.80%	21.54%	27.59%	32.66%	35.75%	38.21%	40.33%	42.25%	X
60	3.00%	10.40%	18.25%	24.30%	29.09%	32.91%	35.79%	37.91%	39.66%	41.70%
68	3.20%	10.13%	16.98%	22.55%	27.26%	31.03%	33.42%	36.23%	38.16%	39.56%
76	1.95%	8.94%	15.20%	20.90%	25.33%	29.34%	31.43%	34.13%	36.41%	37.96%
84	2.25%	7.57%	14.04%	19.26%	23.50%	27.11%	29.81%	32.34%	34.74%	36.36%
92	2.40%	7.50%	12.75%	17.47%	21.54%	25.16%	27.80%	30.71%	32.73%	34.95%