Advanced Topics in Natural Language Processing: Individual Report

Jonas Esterer
University of Copenhagen
fcq595@alumni.ku.dk

Abstract

This study advances on previous work done by Lake and Baroni (2018) and the group project by studying compositional generalization ability of transformer models in a relatively simple sequence-to-sequence environment. In different experiments the zero-shot generalization performance of fine-tuned "T5" models Raffel et al. (2020) is tested. Furthermore, this study specifically investigates the effect of pretraining, increased model size, and number of samples presented. The results show that pretraining and the increasing number of samples improves performance compared to a benchmark while the size effects is estimated to be negative.

1 Introduction

Lake and Baroni (2018) study the compositional skills of artificial neural networks (ANNs). They motivate this by illustrating that humans once they learn a new verb such as "dax", they are seamlessly able to understand composition of this verb such as "dax twice" given the understand "twice". To test this ability of ANN, the authors use a variety of recurrent neural networks (RNNs) on different sequence-to-sequence tasks. For these experiments, they derive a data set from the CommAI environment Mikolov et al. (2018) and call it SCAN (**S**implified versions of the **C**omm**AI** **N**avigation tasks). It is a set of simple language-based navigation tasks used for studying compositional learning and zero-shot generalization which includes more than 20'000 samples. The data consists of commands and corresponding action sequences.

2 Results group project

The goal of the group project was to reproduce the first three experiment of Lake and Baroni (2018) with a self-built transformer modeled after the original transformer architecture of Vaswani et al. (2017). The transformer models included between 570k and 770k parameters. Generally, the transformers performed better than the models used by Lake and Baroni (2018) on all three experiments. This can be interpreted as another piece of evidence for the superiority of the transformer architecture over traditional RNN models. However, Lake and Baroni (2018) also used attention mechanism in their models. Therefore, this mechanism which is often seen as the main strength of the transformer architecture might to be not the decisive factor. It is likely that also size and potentially more extensive hyperparameter optimization (HPO)[1] helped the transformer surpass the results from Lake and Baroni (2018).

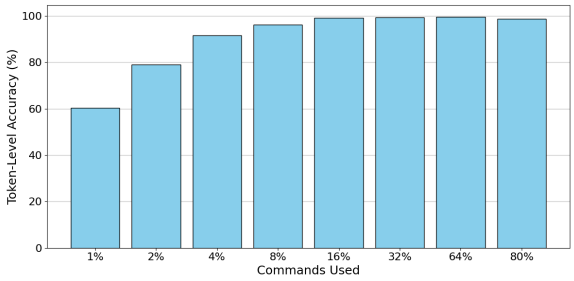


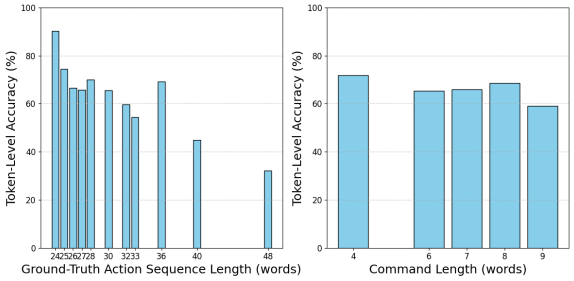
Figure 1: Group Results Experiment 1. Token-level accuracies across share of total data used for training.

2.1 Experiment 1

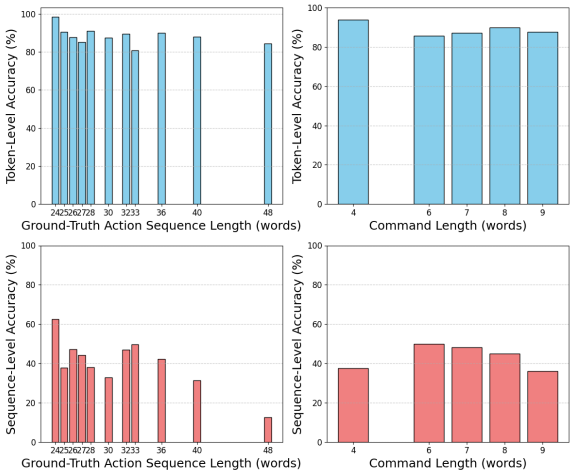
Experiment 1 tests the model's generalization ability as it needs to compositionally recombine pieces of learned commands based on the patterns learned. For this the total data set has been split according to various training shares from 1% to 80%.

The results presented in Figure 1 show the evolution of token-level accuracies across the share of total data used as training data. At 4% of data used for training, which is less than 1,000 distinct samples, the model has a token-level accuracy of above 90%. From 16% onward the performance seems to

[1]The hyperparameters of the models was given to us for each experiment probably based on some HPO.



(a) Group Results Experiment 2 Standard. Performance across action sequences longer than observed during training.



(b) Group Results Experiment 2 Oracle Lengths. Performance across action sequences longer than observed during training with oracle lengths.

Figure 2: Combined Group Results for Experiment 2.

plateau close to 100% token-level accuracy. This is a drastic improvement in performance compared to the original results from Lake and Baroni (2018) in the 1% and 2% data sets. For bigger training shares the performance seems comparable.

This experiment shows the generalization performance of the model. Compared to the originally used RNN models, our self-built transformer model seems to require less distinctive samples to learn the "grammar" of the data set and generalize well to unseen commands.

2.2 Experiment 2

Experiment 2 studies a more systematic form of generalization by splitting the data according to the length of the action sequence. The model must generalize to action sequences longer than observed during training. The models used by Lake and Baroni (2018) perform very poorly on this task and can only extrapolate to samples similar to training (low action sequences and high command length). To test for the causes of this suboptimal performance, the authors test for performance improvements when giving oracle length[2] to the model during evaluation. The authors report improvements in performance but still the model struggle to produce the longer sequences hinting towards deeper problems in generalization.

Results in Figure 2, show that the transformer is able to produce token-level accuracies of more than 50% for all command and action sequence lengths except for the longest (48 words) action sequences. The results show a downward trend of accuracy across action sequence length. When providing oracle length, the accuracies improve substantially especially for longer action sequences which might explain the initial downward trend of the standard results.

The transformer model showed better performance across all sub-experiments compared to the RNN models used by Lake and Baroni (2018). Furthermore, the transformer model seemed to have benefited more from the provision of oracle length. This could be explained by the transformer in general being able to decently generalize to longer sequences but struggling to predict the end-of-sequence token correctly in the extrapolation of very long sequences. Still, the sequence-level accuracies of the transformer remain low even with oracle length.

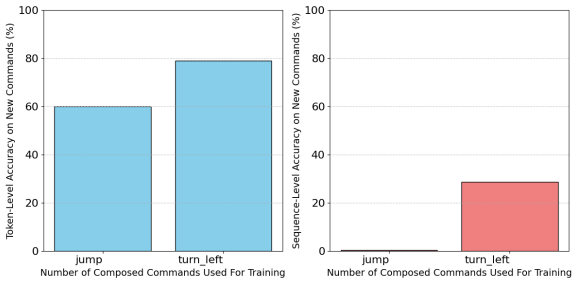
2.3 Experiment 3

The third experiment tests whether the model can generalize rules learned about how primitive commands behave in composition across different primitive commands. The experiment tests this by including few or no samples of "jump" and "turn left" together with other primitives.

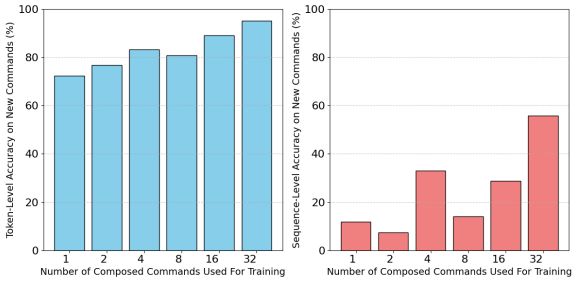
When testing for all the composed commands of the completely omitted primitive commands, the authors find good performance for "turn left" (90.0%) but very poor performance for "jump" (0.08%). The difference likely stems from the fact that "LTURN" is part of action sequence samples without "turn left" occuring in the command by which the model can learn its usage in composed commands. However, when giving the model increasing amounts of distinctive composed commands, the accuracy improves.

Our transformer has much better performance for "jump" and slightly worse performance for

[2]The model is forced to produce the correct length of the action sequence by controlling the end-of-sequence token.



(a) Group Results Experiment 3. Total token-level accuracy between "jump" and "turn left" experiment configurations.



(b) Group Results Experiment 3. The effect of varying the number of composed commands for "jump".

Figure 3: Combined Group Results for Experiment 3.

"turn left" as can be seen in Figure 3. Therefore, also when providing more examples of "jump" the baseline is higher but we see a similar positive trend with the exception of providing four samples which achieves the lowest accuracy. Sequence-level accuracy is very low for both but slightly higher for "turn left" and again the same pattern appears as for token-level accuracy for multiple distinctive samples.

This experiment shows that the studied models do not have "all-or-nothing" compositional generalization but learn differently than expected from e.g. a human.

3 Individual experiment

This study expands on the paper of Lake and Baroni (2018) and the group project by studying the performance of fine-tuned LLMs on the first and second experiment. Furthermore, this study also investigates the effects of pre-training and fine-tuning, size of the used model, and number of samples presented on performance in SCAN tasks. Evidence for the effect of the number of samples presented has been observed during testing of the group project. The size effect has been hypothesized to drive the performance increase of the group project compared to (Lake and Baroni, 2018). The effect of pre-training and general language understanding is interesting for this task as it might reveal underlying logical patterns which can be learned during pre-training making their reasoning closer to human. In summary, study tries to illuminate the driving factors behind performance of transformers on logical tasks.

To study this, the model trained is set-up in different ways. I compare a LLM with its pre-trained initialization and a randomly initialized version. As this model is much larger than the transformer of the group project, also the effect of size is studied as the used LLM has a very similar architecture to the group project transformer[3]. Furthermore, the number of samples which are presented to the model is varied. This directly influences the amount of steps to update the model can do without changing the informational content of the model by keeping the number of distinct samples fixed.[4]

In my individual experiment, I choose the "T5" ("**T**ext-**t**o-**T**ext **T**ransfer **T**ransformer") model (Raffel et al., 2020) as a pre-trained LLM for fine-tuning on this task. This model has a very similar structure to the original transformer built in the group project proposed by Vaswani et al. (2017). "T5" was pre-trained on a filtered version of the Common Crawl web archive [5] which the authors call *C4* (**C**olossal **C**lean **C**rawled **C**orpus). Furthermore, the model was also trained on multiple supervised and unsupervised tasks and datasets for which all tasks have been converted to text-to-text format. Tasks include for example question answering, document summarization, or sentiment classification.

The model was loaded from Hugging Face. I used the "small" version and single runs due to computational and timely limitations. "T5-small" comprises close to 77M total parameters of which 35M are in the encoder and 42M are in the de-

[3] Of course this is not a direct apples to apples comparison due to various reasons. For example, there are slight adjustments in the architecture of the LLM. Furthermore, the group project transformer underwent HPO while the LLM was just run with a recommend learning rate.

[4] Be aware: Lake and Baroni (2018) are a bit unclear on their implementation of batching and feeding data to the model. Our group project implementation therefore, randomly selected batches with replacement and rounded the amount of epochs based on the length of the data set. This leads to a small variance in the actual number of samples presented. While this effect was negligible for 100,000 samples, when studying smaller samples this effect becomes substantial. Therefore, the implementation was changed to rounding on the batch size. Still, the current implementation restricts the random draw with replacement to have equal draws of the data set as long as possible.

[5] https://commoncrawl.org/

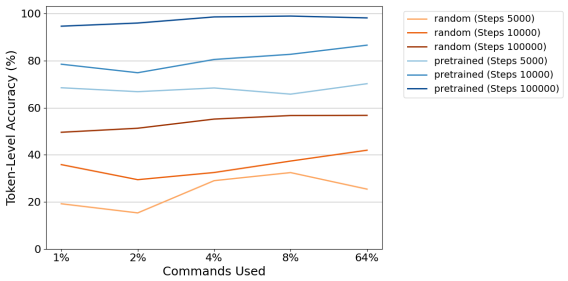


Figure 4: Individual Results Experiment 1. Run for the untrained and pre-trained "T5" model while varying the amount of training samples presented to the model (5k, 10k, and 100k).

coder. Therefore, this pre-trained model is much larger than the self-built transformer from the group project which has at most close to 770k. This is a size increase of about 100 times.

I decreased the batch size to 32 to avoid memory limits in the GPU. The learning rate was set to 0.0002 as the middle value of the range recommended by the Hugging Face documentation (Hugging Face). As for the group project, the AdamW algorithm was used as an optimizer. Training and evaluation was done in Google Colab with the L4 GPU.[6]

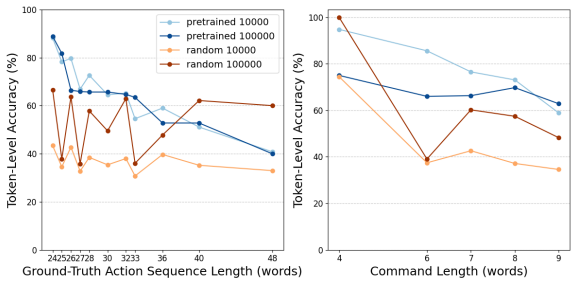
4 Individual results

4.1 Individual results experiment 1

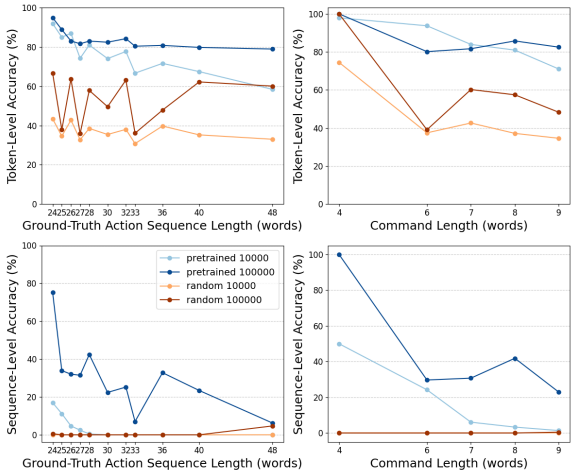
Figure 4 shows my individual results for experiment 1. The random models perform substantially worse than the group transformer even for similar number of samples during training. This hints towards a neutral or even negative effect of increasing the size. Likely the model is to big for the amount of data and optimization steps. However, keep ind mind that the group project transformer benefitted from HPO.

The effect of pre-training is very clear in this task as even the weakest pre-trained model outperforms the randomized model. The pre-trained model with 100k samples shows very strong performance of accuracies close to 95% for 1% of the data. The model generalizes extremely well from very limited sample size to unseen data. Even for only 10000 samples presented, and therefore around 310 updating steps, the model already has accuracies higher than 80%. The low baseline of the random

[6]The L4 GPU showed very similar results in speed compared to the more expensive A100 GPU. This is likely due to the efficiency bottleneck when decoding, see section 5.



(a) Individual Results Experiment 2 Standard. Performance across action sequences longer than observed during training.



(b) Individual Results Experiment 2 Oracle Lengths. Performance across action sequences longer than observed during training with oracle lengths.

Figure 5: Combined individual results for Experiment 2.

model makes the effect of pre-training even more evident.

As for pre-training, the effect of increasing number of samples can be seen very clearly. Each increase in samples presented leads to an increase in performance.

4.2 Individual results experiment 2

The results of my individual implementation of experiment 2 can be found in Figure 5. Number of samples has been chosen at 10k and 100k as these implementations showed the ability to both have good performance in experiment 1 but with enough variance between them.

The random initialization performs worse in most cases for both versions trained on 10k and 100k total samples. Unfortunately, the implementation of the oracle lengths for the random models seems to have a bugg as the token-level accuracies are mirrored for both the standard and oracle length variant. Also the sequence-level accuracies seem

to be incorrect at least for some split[7] Still, similarly to experiment 1, the results of experiment 2 can be interpreted as signs that the size of the group transformer already is sufficient to learn the pattern of the task decently well. When keeping in mind that the group transformer underwent HPO, this makes sense. It is likely that the randomly initialized model is lacking fine-tuning of hyperparameters.

The results of the pre-trained models show good performance and clearly outperform the randomly initialized models in the standard set-up. Due to the incorrect implementation of the random models with oracle length, the comparison for this set-up cannot be studied. However, it is still notable how the pre-trained models improve significantly due to the oracle lengths. Especially, for 100k samples there is a drastic boost in performance. The level of performance is very comparable to the group implementation. Arguably, pre-training on general language tasks can help to improve generalization ability across test regions which have been unexplored during training.

For the random models a consistent performance improvement can be seen for more samples. For the pre-trained model, the effect is less clear for the standard set-up. With oracle lengths, the effect becomes more distinguished. Increasing the number of samples presented increases the models ability to update its parameters. This might lead to more stable performances. However, it is also possible that the model starts to overfit at some point. This would fit the theory that the model had more time to learn the general patterns of the data but overfit to strongly regarding to e.g. end-of-sequence tokens. In experiment 1, we might not have observed this bias as the training was representative across all dimensions of the test data. Therefore, the model benefits a lot from a relief of this bias. In this set-up unfortunately, there is no mechanism to give an indication of overfitting.

In conclusion, the results likely show that randomly initialized "T5" models perform worse than the group transformer. Pre-training improves the performance of the model compared to random initialization. Furthermore, it seems that more samples presented to the model improves performance although this trend is not unambiguously. The trends are similar to the ones observed in experiment 1, although slightly more nuanced.

5 Discussion

This implementation has shown that size, pre-training, and number of samples presented do matter in the implementation of experiments 1 and 2. However, the results especially ine experiment 2 show that the effects are more nuanced. One big limiting factor in the analysis is the difference in HPO comapred to the group project implementation. This disenables an direct comparison between group and individual results.

Studying the computation time revealed interesting patterns between the group project implementation and the "T5" implementation. For the group project, training was the limiting factor using around 45 minutes for training and below one minute for evaluation. The "T5" implementation had up to 2 hours for training[8] and up to an hour in experiment 2[9]. The bottleneck for the individual implementation was however evaluation which took two hours in many instances[10] due to the "T5" tokenizer only having a decode() function and not a decode_batch() function. As the "T5" implementation is 100 times larger than the group transformer an efficiency improvement in training is fair to attribute while a clear disadvantage in evaluation has to be highlighted. Overall, the group implementation had a more efficient performance; using less time but often producing similar performance levels.

In conclusion, the results of this paper showed that pre-training and the number of steps can have a strong effect on performance. The effect of step size however was not always very clear as e.g. in experiment 2 a pre-trained model with limited number of samples performed already quite strongly. The size effect was negative for this specific implementation likely due to a bias of lacking HPO. Future research could improve the comparability of the configurations by studying the effect of HPO and controlling for computation time.

[7] The random model for 100k samples has 100% token-level accuracy for a command length of 4 but has 0% sequence-level accuracy for command length 4. This hints towards a bugg in the implementation likely when calculating the accuracies. Also possible is that for some reasons the random models have been using outdated code. Unfortunately, due to timeliness reasons I could not rerun the experiments.

[8] For 100k samples presented in experiment 1.

[9] However, for a maximum of 100k samples which is less than the 200k of the group implementation.

[10] Unfortunately, the evaluation time is not effected by number of samples presented which made each run very costly.

References

Hugging Face. T5 Documentation. https://huggingface.co/docs/transformers/en/model_doc/t5. Accessed: 2025-01-17.

Brenden Lake and Marco Baroni. 2018. Generalization without systematicity: On the compositional skills of sequence-to-sequence recurrent networks. In *International conference on machine learning*, pages 2873–2882. PMLR.

Tomas Mikolov, Armand Joulin, and Marco Baroni. 2018. A roadmap towards machine intelligence. In *Computational Linguistics and Intelligent Text Processing: 17th International Conference, CICLing 2016, Konya, Turkey, April 3–9, 2016, Revised Selected Papers, Part I 17*, pages 29–61. Springer.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of machine learning research*, 21(140):1–67.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *CoRR*, abs/1706.03762.

AI Statement

AI has been used to improve writing and coding of this study. In particular, ChatGPT has been used to improve the writing style, check grammar, and help debug code. Copilot has been used to code more efficiently and for debugging. Gemini has been used in Google Colab to debug code.