Advanced Topics in Natural Language Processing

Individual Report

Jeno Toth (fwj301)

January 19, 2024

1 Introduction

Lake and Baroni (2018) has shown how to implement Sequence-to-Sequence Recurrent Networks on the SCAN (Simplified version of the CommAI Navigation) dataset, training models from scratch. This paper describes our efforts together with Matias Giovanni Salaris and Yannick Neubert to replicate the models and its results, as well as introduce a new framework based on fine-tuning T5 transformers in order to create new models capable of understanding the SCAN-tasks.

2 Group Reimplementation Project

2.1 Methodology

For most cases, similarly to the paper, we followed PyTorch's tutorial for seq2seq translation. The main differences between our methodologies therefore arise from the details not described in Lake and Baroni (2018). The first, and perhaps most influential of these is the use of padding. Padding in sequence-to-sequence translators is used mainly to ensure that all the input and output sequences are of the same length. Since whether it was used or not was neither described in the paper nor in the tutorial, we tried building models both ways. Despite its potential to improve translators, we found that when we removed padding, our models often performed better. One reason for this could be that this way, the models were able to focus on the true length of input strings, which contains relevant information for the output lengths. Another explanation could be that this way the model is able to focus on the actual input tokens more.

Another unclear parameter was teacher forcing. While it was said that teacher forcing of 0.5 was used, it was unsure whether this means that there is 0.5 probability for each iteration to be teacher-forced, or if it means that there is teacher-forcing for the first half of the iterations. After testing both methods, we found that the latter case provided better results.

Finally, due to the lack of computational resources, we were unable to train each model 5 times, similarly to the paper. This could cause a difference in our results, since there were a number of instances in the paper, where there was a large standard deviation in model accuracies.

The overall best model we trained was an encoder-decoder with a 2-layer LSTM, 200 hidden units per layer, no attention, and dropout at 0.5 level, as described in the paper. Despite our best efforts, we were unable to achieve the 99.5% accuracy that was in the paper, achieving a slightly lower result.

2.2 Experiments

For the first experiment, we trained an LSTM model with 2-layers, 200 hidden units per layer, without dropout. Our efforts to reproduce the paper's results were successful when training on 1%, 8%, 16%, 32% and 64% of the commands, only falling behind when it was 2% and 4% (this might have been caused by high variations in training accuracies, since these models were only trained once, unlike in the paper, where each were trained 5 times, taking the average of the accuracies). Our best model achieved 99.7% accuracy, only 0.15% lower than what the paper described.

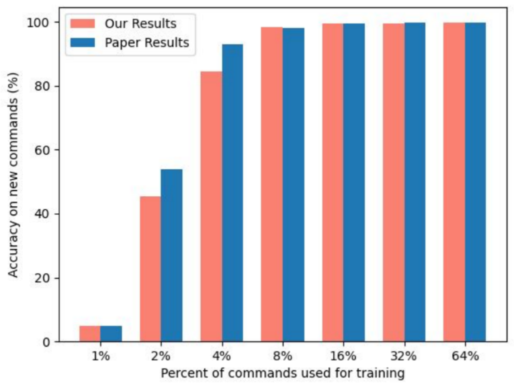

Figure 1: Our results for experiment 1.

The second experiment, where we trained on shorter sequences and evaluated on longer ones, was less successful. The model trained here was a GRU with 1-layer, 50 hidden units, with attention, and with 0.5 dropout. While the paper achieved 20.8% accuracy with this model, we found significantly lower results, with 9.3% accuracy. These models were trained multiple times, therefore simple between-training errors are unlikely to cause this. Despite this, the the accuracy distributions on different sequence and command lengths are similar to those shown in the paper, therefore it is possible that the models do work similarly, the difference in accuracies caused by small mistakes.

For the third experiment, for the LTURN case, once again, our model with 1-layer GRU, 100 hidden units, attention and 0.1 dropout achieved worse results (66.5% accuracy) compared to the 90.3% described in the paper. We also tried running a model here without attention, and it got similar results (64% accuracy), therefore our hypothesis is that there is an issue with the attention mechanish.

For the JUMP case in experiment 3, it was hard to evaluate how our model compares to the one in the paper. This is because the paper achieved very low results (1.2%), and it might be that our accuracy of 0.2% is close to the paper, and there is just a difference due to the randomness of the training.

2.3 Discussion

Despite our best efforts, for the majority of the experiments we were unable to reproduce the exact results that Lake and Baroni (2018) achieved. While our models were built on similar architectures, and the training data was the same, our outcomes underperformed those in the paper. Further resources, such as the exact architecture the paper used (regarding padding, teacher-forcing and the attention mechanism) might enable us to find our mistakes and fix our models.

3 Individual Implementation

3.1 Methodology

Since the SCAN dataset is rather simple, training models from scratch, such as in the reimplementation project, might prove fruitful. Another approach is to fine-tune a pre-trained large language model on the data. This method can capitalize on the vast knowledge and understanding of word structures of the pre-trained model. After trying out a number of pre-trained architedtured, the best potential was seen from the Text-To-Text Transfer Transformer (T5)-small model. It is a model based on the T5-architecture, limited to 60 million parameters. While larger models might be better for large tasks, due to the relatively small size of the SCAN dataset, the T5-small model was expected to do well.

After trying multiple variations, the final model was fine-tuned with padding, using AdamW with learning rate of 0.00005, and a custom tokenizer based on the SCAN dataset's words. Training and evaluation worked similarly to the previous section, the biggest difference being that instead of training for a number of iterations, with a random training sample in each iteration, the models were trained in epochs. The number of epochs was based on the number of training inputs, so that it would be close to 100,000, in order to have comparable results to the paper. During evaluation, exact matches were considered as the main indicator for model performance, however, partial matches were also taken into consideration.

After building the baseline model as described above, the exact match accuracy was a very low 0.2%. Despite this, partial accuracy was much higher at 6.2%, which meant that the model was understanding the data better than expected based on the exact accuracy. As such, three more baseline models were tested. One with approximately 500,000 iterations using the T5-small model, and two new models using the T5-base (220 million parameters) model, one with 100,000 and one with 500,000 iterations. The results proved that the model was working well with the data, it just did not have enough time to train fully on it. The T5-base model with around 100,000 iterations achieved 0.38% exact match and 11.4% partial match. The T5-small model with 500,000 iterations achieved 7.01% exact match and 30.1% partial batch while the T5-base model with the same amount of training achieved 14.12% exact match accuracy.

This caused an interesting trade-off to be observed. It was clear to see that compared to the paper, the fine-tuned models required more training. The T5-base models performed better than their small counterparts, which was expected, and they could have provided better results for all experiments. This approach, however, could have led to a slippery slope, since this could open ways to models like the T5-large (770 million parameters) and more training time. To limit this, the two models I choose were the T5-small with approximately 100,000 and 500,000 training iterations. Training these models already required significantly more capacity than the ones described in the paper, and the purpose of my findings was not to build the most optimal T5 model for the data, but to see the potential of fine-tuning such architectures on the SCAN dataset.

3.2 Experiments

For the first experiment, the training inputs were randomly chosen for each part, choosing 1%, 2%, 4%, 8%, 16%, 32% and 64% of the inputs respectively. The models were then trained in similar fashion to the baseline (with more epochs, since the number of training inputs were smaller), and evaluated the same way. Unsurprisingly, the models trained on 100,000 iterations achieved very low exact accuracies. For 1%, 2%, 4%, 8%, 16%, 32% and 64% of the inputs, they achieved 0%, 0%, 0.02%, 0.1%, 0.17%, 0.19% and 0.17% respectively. This was to be expected, since the baseline model also achieved such low results. For the models trained on around 500,000 examples, the results were 0.24%, 1.11%, 2.78%, 4.89%, 5.79%, 5.91% and 6.22%. This means that once again, more training examples provided with better model outcomes. However, the values were lower than baseline for the same model, unlike in the paper, where after 8% of the training commands used, the models performed close to the overall best.

The second experiment saw the same two models training, but on different data. They were on another dataset which had shorter sequences for training and longer ones for evaluating. In this experiment, the paper produced worse accuracies than before, therefore, it was not expected that the T5-models will perform as well as in experiment 1. Despite this, it came as a surprise, that the exact match accuracy was 0% for both models. After looking at a few of the outputs, it seemed that the model predicts the beginning of the output strings well, but it predicts the end of sequence too early. This is most likely caused by the fact that it was trained on shorter sequences.

3.3 Discussion

The exploration of fine-tuning a T5 transformer model on the SCAN dataset offered valuable insights into the adaptability and performance of pre-trained models on specific tasks. Our experiments revealed that while the T5-small model showed promising initial results, it required significantly more training iterations to achieve notable accuracies, compared to the models in the reimplementation project. This highlights the trade-off between model size and training efficiency, especially for datasets like SCAN, which are relatively simple yet nuanced. Our decision to limit the exploration to the T5-small model was driven by the need to balance computational resources with the objective of evaluating the potential of fine-tuning strategies. Interestingly, partial match accuracies were considerably higher than exact matches, suggesting that the T5 model was capturing some underlying structures of the task but not mastering it completely. These findings underscore the complexity of applying large pre-trained models to specialized tasks and pave the way for further research into optimizing training strategies for such models.

4 Conclusion

Our project, aimed at replicating and building on Lake and Baroni (2018) on sequence-to-sequence recurrent networks, underscores the complexities and challenges inherent in machine learning research. While we successfully implemented various models and conducted thorough experiments, our results varied from the original study, highlighting the critical role of precise methodology details in replicability. This experience not only deepens our understanding of sequence-to-sequence modeling but also contributes to the broader discourse on the replicability of machine learning studies. Future work should focus on improving transparency in methodological reporting and exploring the impacts of different model configurations on learning outcomes, particularly in the context of sequence processing tasks.

References

Lake, B. M. and Baroni, M. (2018). Generalization without systematicity: On the compositional skills of sequence-to-sequence recurrent networks. *arXiv*.