Unlocking the Potential of Small Language Models with Reinforcement Learning

David Florez Fernandez
University of California, Berkeley
davidflorez@berkeley.edu

December 9, 2023

Abstract

This paper explores the implementation of the Reinforced Self-Training (ReST) algorithm, a novel approach from DeepMind, specifically adapted to enhance the ALBERT model for sentence similarity tasks within the Microsoft Research Paraphrase Corpus (MRPC) dataset.

My implementation demonstrates the integration of ReST with ALBERT, an encoder-based model known for its efficiency in language understanding. I meticulously examine the model's performance in identifying sentence-level semantic similarities as a classification problem, leveraging the ReST algorithm's capability to refine the model's outputs based on aligned rewards. The research concludes by reflecting on the implications of these findings and proposing future directions for advancing LLM alignment methodologies.

1 Introduction

Recent advancements in large language models (LLMs) have significantly pushed the boundaries of text generation and language task solutions. These models, trained to maximize the likelihood of generating the next token in a sequence, have demonstrated impressive abilities in producing high-quality text outputs. However, a notable challenge that persists in LLMs is aligning their outputs with human preferences. Without proper alignment, the potential of these models is not fully realized, and they may even produce undesirable content.

To address this, DeepMind proposes a novel method called Reinforced Self-Training (ReST, see Refence [3]). ReST allows for the efficient production of datasets and iterative improvement of the model. The flexibility of ReST allows it to be applied to various generative learning settings, although DeepMind's research primarily focuses on machine translation.

The objective of the present research is to analyze how ReST can be integrated with ALBERT model to demonstrate the potential improvements in

performance for sentence similarity tasks. In the following sections I will talk about the methodologies, experiments, and findings of the study.

2 Description of the ReST Algorithm

In this research, I implement and explore an inspired Reinforced Self-Training (ReST) approach, specifically examining its application to an ALBERT model for sentence similarity tasks. It is important to note that it could be differences with the ReST the authors of Deepmind have used. The focus of the study is on the MRPC dataset, a widely recognized benchmark in natural language processing. The MRPC dataset, comprising sentence pairs with annotations indicating whether each pair captures a similar meaning, provides an ideal testbed for assessing the effectiveness of ReST in enhancing the model's capability to discern nuanced linguistic similarities.

This is the high-level description of the ReST algorithm:

1. Grow (G) Step:
    - The LLM policy, initially a supervised policy, generates multiple output predictions. This augments the training dataset by a number **N** of samples from the original dataset.
    - This step corresponds to the acting or data-generation phase in reinforcement learning (RL). It involves creating an augmented dataset of trajectories by sampling many output sequences from the current policy.
2. Improve (I) Step:
    - The augmented dataset is then ranked and filtered using a scoring function. In the experiments, a learned reward model based on NLL is used as the scoring function.
    - A threshold is defined as the average of the NLL in the augmented dataset, it started with that value as filter threshold and in next iteration the new filter threshold is subject to be lower strictly to the previous one.
    - The LLM is fine-tuned on this filtered dataset. This step can be repeated with an increasing filtering threshold, and the final policy is used in the next Grow step.

2.1 Detailed Description of the Implemented Algorithm

This research implements the Reinforced Self-Training (ReST) algorithm using the ALBERT model for sentence similarity tasks on the MRPC dataset. The implementation is structured into colab notebooks.

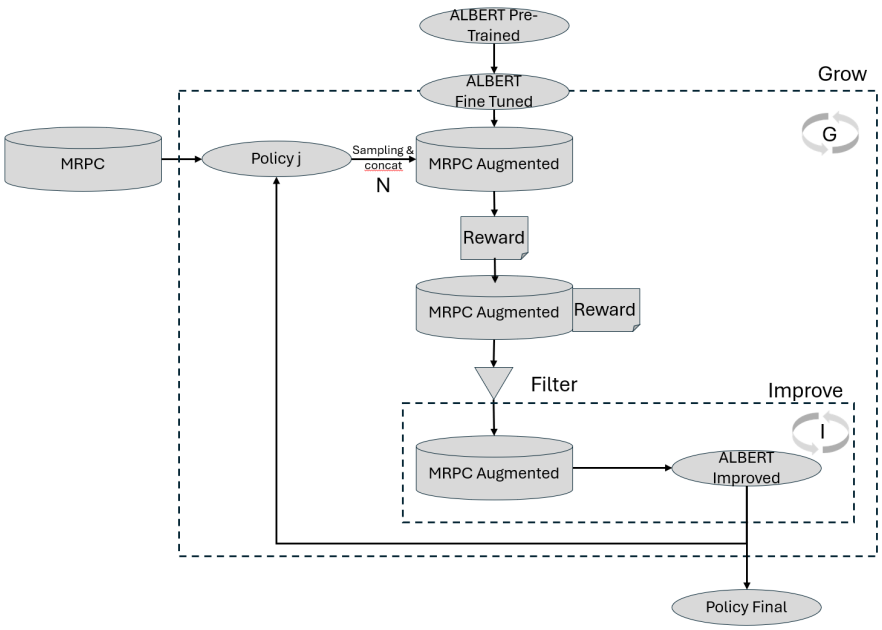

Figure 1: Representation of the ReST algorithm for ALBERT and MRPC

The Figure 1 outlines a process for iteratively refining a policy using a pre-trained ALBERT model, with the ultimate goal of improving its performance on the MRPC dataset. The MRPC dataset serves as the starting point for this process.

First, a policy pre-trained is fine tuned (denoted as "G=0, I=0") and then is applied to the MRPC dataset to select samples and concatenate additional data, creating an "MRPC Augmented" dataset. This augmented dataset is then evaluated to assign rewards to each data sample based on how well the policy performed—essentially scoring the policy's decisions.

Next, this rewarded dataset undergoes a filtering process. Here, samples are chosen based on their reward scores—only those that meet a certain threshold are kept (lower than a given τ). This filtered dataset, which is now a subset of the "MRPC Augmented" containing lower-reward (remember lower is better, because NLL are negative) samples, is used to further train (or "improve") the ALBERT model. This stage is where the actual policy optimization occurs, fine-tuning the model based on the "better" examples as determined by the reward scores.

The improved ALBERT model then defines a new policy ("Policy Final"), which ideally performs better than the initial one due to the iterative refinement process. Overall, the process depicts a feedback loop where the model is continuously trained and refined with the aim of incrementally enhancing its decision-making capabilities with each iteration. Important to note is that this

process can easily incorporate human feedback instead of having a self generating prediction. This opens possibilities to LLM alignment as detailed in the final section.

3 About the Dataset and the Reward Function

In the MRPC (Microsoft Research Paraphrase Corpus, see ([1])) dataset, each example consists of two sentences, Sentence1 and Sentence2, and the task is to determine if they are semantically equivalent (paraphrases of each other). Training consists in 3668 pairs and Validation in 408. Going forward, I am going to use the NLL (Negative Log Likelihood) as the reward function.

Let's consider an example:

- Sentence1: "Legislation making it harder for consumers to erase their debts in bankruptcy court won overwhelming House approval in March."
- Sentence2: "Legislation making it harder for consumers to erase their debts in bankruptcy court won speedy, House approval in March and was endorsed by the White House."

The model's task is to predict if these two sentences are paraphrases. Let's say the model predicts they are paraphrases with a certain probability. The NLL for this example would be a measure of the "cost" or "penalty" for the model's prediction given the actual label. If the model is correct, the NLL would be lower; if incorrect, it would be higher.

4 Results

In my implementation, I use GPT4 as the baseline to compare all my models on their performance over the validation data. I use Open AI API to run queries to GPT4 models using zero-shot prompting with the following prompt:

```
Are the following two sentences paraphrases of each other?
Sentence 1: {sentence1} Sentence 2: {sentence2}.
Answer with 'Yes' or 'No'.
```

I used the pre-trained ALBERT model to check how this model acts for a new dataset. And ALBERT fine-tune it is trained with the MRPC train dataset in 3 Epochs. For the further training in ReST I used 1 epoch to avoid over-fitting.

In Table 1, we present the accuracy of three different language models. The GPT4 Zero shot model has an accuracy of 0.7819, which is quite high compared to the ALBERT Pre-Trained model with an accuracy of 0.3161. However, the Albert Fine-Tuned model surpasses both with an accuracy of 0.8588 showing how for some tasks is better the use of SML.

Table 2 details the performance of the ALBERT ReST model under various settings. The results are segmented into two configurations based on the number

of observation sampling and predicted with the latest model/policy, N=100 and N=1000, alongside the percentage improvement compared to a baseline. Notably, the configuration with G=2, I=3 at N=100 results in the highest recorded performance of 0.8774 with a 4.68% improvement over the initial policy of the ReST execution (G=0, I=0). For N=1000 I dindt do more growing steps because the results were already worse than N=100. Also the comparison need to be done at column level, it means compared with the G=0, I=0 aka fine-tune to keep the same waterfall of model improvements.

Model	**Accuracy**	**# Parameters**
GPT4 Zero shot (Baseline)	**0.7819**	175 B
ALBERT Pre-Trained	0.3161	12 M
ALBERT Fine-Tuned	0.8588	12 M

Table 1: Model Accuracies

ALBERT ReST	**N=100**	**%Improve**	**N=1000**	**%Improve**
G=0, I=0	0.8382	-	0.8651	-
G=1, I=1	0.8333	-0.58%	0.8724	0.84%
G=1, I=2	**0.8627**	**2.92%**	0.8701	-0.58%
G=1, I=3	0.8455	0.87%	0.8701	0.58%
G=2, I=1	0.8284	-1.17%	-	-
G=2, I=2	0.8480	1.17%	-	-
G=2, I=3	**0.8774**	**4.68%**	-	-

Table 2: ALBERT ReST Performance

The analysis of the rewards and the comparison with Accuracy is interesting. In the original paper Deepmind dint mention even a single time accuracy all is focused in rewards. Figure 2 of NLL scores obtained after various iterations of the ReST algorithm presents an intriguing insight into the performance optimization of the reward model. As indicated by the data, a reduction in NLL does not unequivocally translate to enhanced accuracy. This non-linear relationship can be attributed to the complexity of the model's learning landscape and the intricacy of the data it aims to represent.

The plot distinctly showcases three different stages of the ALBERT model with ReST: fine-tuned, G=1, l=3, and G=2, l=3. While the fine-tuned variant achieves a lower accuaracy of 0.8382, it is not immediately apparent whether this improvement in accuracy corresponds to a proportionate increase in likelihood. The interplay between NLL and accuracy is multifaceted, often obscured by the nuances of the model's architecture and the data's inherent characteristics.

The optimization procedure implemented for reward (NLL) reduction demonstrates a manual approach, which, while yielding significant insights, necessitates automation. The process should ideally converge autonomously upon the most efficient model and hyperparameter set at each stage. So far I am only do-

ing a decreasing Learning Rate over each Improvent step to avoid over-fitting. This would not only streamline the model's development cycle but also potentially prevent overfitting to the training data by halting the training process when the model begins to exhibit diminishing returns in terms of accuracy gains. Some more advanced reward models (see Reference [2]) were studied for further research.

In conclusion, the observed data underscores the necessity for a more sophisticated, automated approach to model optimization that can discern the optimal balance between NLL reduction and actual performance gains. Such an approach would be invaluable in ensuring that the computational resources are dedicated to achieve improvements in model accuracy.

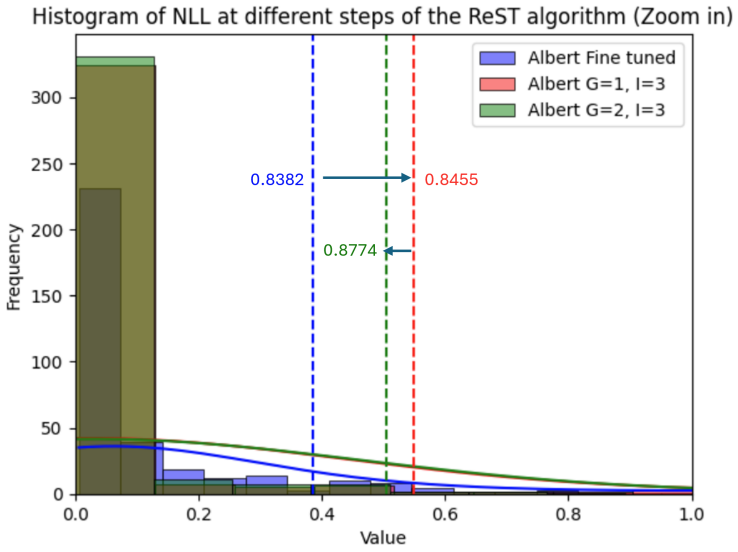

Figure 2: Zoom of the histogram with average value in dotted lines and accuracy in numbers in color

5 Conclusions

The efficacy of the ReST algorithm is evident in its ability to enhance smaller models, which are inherently more cost-effective and manageable for fine-tuning compared to larger models. The iterative process facilitated by ReST leads to performance improvements, achieving incremental gains that are significant in the realm of machine learning. Getting the reward settings right is crucial for the success of the ReST algorithm, and I'm eager to see how DeepMind has approached this in their work once they will share it publicly. This could make optimizing smaller models even more effective.

6 Appendix

6.1 Future Research

The concept of alignment in large language models (LLMs) has become increasingly important as these models find broader applications in various domains. Misaligned models could inadvertently generate biased, offensive, or harmful content, leading to mistrust and potential misuse of the technology. Therefore, alignment ensures that LLMs are not only technically competent but also socially responsible and beneficial.

One of the current research trends in this field is the development of algorithms that can effectively align LLMs with desired human-centric outcomes and can learn how to learn. Among these is the Open AI Q* algorithm, which represents a significant stride in reinforcement learning. Unlike traditional RL methods that heavily rely on trial and error, Q* focuses on learning from an existing corpus of human feedback. This approach presents a more data-efficient way of training LLMs, reducing the computational overhead and the potential risks associated with live interactive training. The Q* algorithm, with its emphasis on learning from curated human feedback, exemplifies the shift in research towards creating LLMs that are not only powerful in processing language and reasoning but also aligned with the nuanced expectations and ethical considerations of human users.

6.2 Development Environment

I used Google Colab Pro account to run all these experiments. I used T4 environment. The key packages were:

- SentencePiece
- Datasets
- Transformers
- Tensorflow
- Numpy
- Seaborn
- Matplotlib

For GPT4 I used the Open AI API.

References

[1] Download microsoft research paraphrase corpus from official microsoft download center. https://www.microsoft.com/en-us/download/details.aspx?id=52398, 2023. Accessed: 2023-10-06.

[2] Reward modeling. https://huggingface.co/docs/trl/reward_trainer, 2023. Accessed: 2023-09-30.

[3] Caglar Gulcehre, Tom Le Paine, Srivatsan Srinivasan, Ksenia Konyushkova, Lotte Weerts, Abhishek Sharma, Aditya Siddhant, et al. Reinforced self-training (rest) for language modeling. https://doi.org/10.48550/arXiv.2308.08998, Aug 2023.