“End Of Story”: Can small language models generate good story endings?

Licheng Zhong
lz0417@berkeley.edu
School of Information
University of California, Berkeley

Siddharth Manu
siddharthmanu@berkeley.edu
School of Information
University of California, Berkeley

Abstract

Completing a story is a challenging open-ended generative task requiring structured grammar, language fluency, and sensible creativity. Its versatility empowers its applications in various real-life use cases. While large language models like GPT-4 have proved their competency over this task, they may be proprietary or costly to use. In this study, we tackle the story completion task by fine-tuning two lightweight language models, BART and TinyLlama with ROCStories, a collection of short stories. We found significant improvements over the pre-trained language models to the fine-tuned one in both subjective eyeball reviews and quantitative metrics, such as ROUGE and LLM-Eval.

1 Introduction

Story completion is an interesting natural language generation problem, as it is a creative task and has no single right answer. Completing a given story involves understanding the context, including the plot and characters and following temporal and causal sequences and generating a conclusive sentence that completes the story's arch. The problem presents unique challenges with short stories, as they have small story arches and must provide a consistent, reasonable and satisfying end, based on the story context provided.

While LLMs are powerful enough to generate story endings, they come with challenges related to cost as well as the difficulty in fine-tuning them for specialized tasks. Small language models, on the other hand are lightweight, easy to fine-tune and can be deployed on edge devices as well.

In our study, we explore the ability of smaller language models (SLMs), both encoder-decoder models and decoder-only models to generate meaningful endings to short stories. We evaluate the baseline and fine-tuned model performance for both the model types. We are hopeful that the model and the techniques learned can be applied to other datasets and longer length stories and have real world uses. Our project is hosted at https://github.com/sidmanu/EndOfStory

2 Background

Story completion has intrigued many NLP researchers. Completing a given story involves understanding the context, including the plot and characters and following temporal and causal sequences and generating a conclusive sentence that completes the story's arch.

Since the availability of the ROCStories dataset and the StoryCloze test, this problem has been approached in myriad ways. CaTeRS (Mostafazadeh et al., 2016) introduces a semantic annotation framework to learn causal and temporal relations between events. (Chen et al., 2018) try a similar approach and also introduce implicit common-sense knowledge outside the story context to make gains on the story completion performance.

There have also been more complicated approaches using graphs and annotations like the use of Multi-Level Graph Convolutional Networks over Dependency Parse trees. (Huang et al., 2021). Encoder-decoder models have also been built using LSTM (Chen et al., 2018) for story ending generation.

With the availability of transformer-based language models, with their great ability to understand context, the task requires a fresh-look and using the extraordinary power of LLMs, the benchmarks now sees great improvement. However, the use of small language models (SLMs), which are much more accessible and trainable, has not been widely researched. Unlike their large counterparts, SLMs have limitations with handling complex prompts and following instructions. They also have challenges due to lack of depth and complexity in language understanding, vocabulary size, linguistic style and rich story narratives, due to their limited training and parameter constraints. Hence, given their limitations, further experimentation is needed to see if they could be trained for story generation tasks.

In our work, we try to explore fine-tuning two lightweight models, one based on encoder-decoder architecture and another that is a decoder-only model to generate meaningful endings to short stories. Among the two, due to the ability of encoder-decoder models to build richer story contexts in the encoder, we expect the encoder-decoder model to perform better as compared to the decoder-only model. However, decoder-only models are becoming more popular and we hence also want to explore their performance at the task and their ability to be fine tuned.

3 Methods

We selected BART (et al., 2019) and TinyLlama (Zhang et al., 2024) as two lightweight language models for this study. We construct pairs of story bodies and reference story endings from the ROCStories and Story Cloze Test. Our evaluation metrics are perplexity, ROUGE (ROUGE-1, ROUGE-2, and ROUGE-L), LLM-Eval score by Prometheus. (Kim et al., 2024). We establish our baseline scores by evaluating the generated outputs from the pre-trained BART and TinyLlama models against the reference story endings. Then, we fine-tune both models, re-evaluate them and cross-compare their generated story endings along with the reference story endings.

3.1 Dataset

To train and evaluate our models, we decided to use a well-known dataset, ROCStories and StoryCloze Test. We chose this dataset because it's a successfully established dataset with good credibility and clean data. Moreover, while this dataset has been widely used, it's mainly used to train a classifier for StoryCloze Test, a classification task determining which story ending is correct, given a story body. But in our study, we use this dataset for generating story endings; given this dataset is of five-sentence short stories, its manageable size makes it a good candidate for us to fine tune our lightweight language models and aligns with our study mission: trainable with easy access.

This dataset has two parts, one is a ROCStories dataset of full five-sentence stories, the other is a Story Cloze test dataset of four-sentence story body along with a correct story ending and an incorrect story ending. There are two iterations of ROCStories. One is ROCStories and Story Cloze Test from 2016 (et al., 2016), which has ROCStories 2016, ROCStories 2017, and Story Cloze 2016. The other is Story Cloze Test from 2018 (Sharma et al., 2018). The Story Cloze Test 2018 is a recommended over the Story Cloze 2016 because it has been mitigated for bias.

We have found no duplicates among ROCStories 2016, ROCStories 2017, Story Cloze 2016, Story Cloze 2018. We combined ROCStories 2016 and ROCStories 2017 into a one consolidated ROCStories dataset, with a size of 96K. Given that we use ROCStories datasets for fine-tuning our models, we decide to have the whole Story Cloze 2018 as a part of test data. Since it's possible that Story Cloze dataset has a different creation process from ROCStories, having them in the test data increases the test data variance, and such variety enhances the generalizability represented by the test data. Given Story Cloze 2018 has a size around 1.5K, we split 2 percent of the consolidated ROCStories, roughly 1.9K to form a balanced test data with the Story Cloze 2018. For the rest of consolidated ROCStories, we split 80 percent as the training dataset with a size around 76K and the rest as the validation dataset with a size around 19K. Story Cloze 2016 is reserved for a special task of only fine-tuning BART's encoder.

The ROCStories dataset have five sentences. We combine first four sentences as the story body input and the fifth sentence as the story ending reference. For Story Cloze 2018, we also combine first four sentences as the story body, but we chose the correct ending as the story ending reference; we discarded the incorrect ending, given our it is intended for the Story Cloze Test classification task, which is not applicable to our story ending generation. For Story Cloze 2016, given it's later used for fine-tuning BART's encoder as a classification task, we create two stories by combining first four sentences respectively with the correct and incorrect ending.

To ensure evaluation can be done in a more accessible platform, we use DataLoader with a batch size of 32 to load tokenized story body input and plain story text strings. Without doing so, evaluation can crash a GPU with 32 GB RAM. With the DataLoader, we are able to evaluate with a free-tier Colab T4 GPU with 16 GB Ram.

Original story body	**Expected ending**
Evan had been saving for years. He went to the dealership and bought a really fancy BMW. Evan was so proud of his new car. He showed it off around town.	Evan knew he looked cool in the new car.

Table 1: Story completion example

BART baseline$*$	**BART fine-tuned**	**TinyLlama baseline**$+$	**TinyLlama fine-tuned**$+$
Evan had been saving for years. He went to the dealership and bought a really fancy BMW. Evan was so proud of his new car. He showed it off around town. It was a big hit with his friends and family. He was so excited to have a new car to drive.	Evan's friends and family loved his new car, and he was happy.	Evan had been saving for years. He went to the dealership and bought a really fancy BMW.	Evan was driving home one day. He saw a car with a broken taillight. He pulled over and helped

Table 2: Generated story-endings
$+$ Generated outputs included the original story and prompt, which was removed in post-processing
$*$ BART baseline tries to generate the complete original story along with an ending, but includes errors. After fine-tuning, only endings are generated.

3.2 Evaluation metrics

Since story-ending generation is a text-generation task with a conclusive ending. However, story writing is an art and often there could be multiple endings to the same given story. Thus, ideally, we would have loved to have human evaluation the true test of our story-ending generation model. However, since that is not a possibility, after much discussion and exploring various metrics including BLEU, we decided to stick to two metrics that complement each other - Perplexity and ROUGE.

Since story ending generation needs to fluently follow the given story context, we use perplexity as a metric to capture the flow of the story as well as the continuity of semantics. On the other hand, in order to make sure the generated ending follows the short-arch of the story and aligns with the reference during evaluation, we chose ROUGE, specifically ROUGE-1, ROUGE-2 and ROUGE-L which evaluate the generated ending against the reference text and make sure it as per our expectations.

Finally, we decided to choose LLM-Eval score as a metric, the next best alternative to human evaluation. We use the 7B version of Prometheus (Kim et al., 2024), an open-source model built for evaluation that rivals GPT-4's evaluation capabilities, to score the endings generated by our models. We configured the GPU Memory Utilization to 0.5 and capped Max Model Len under 2048 to balance memory use and speed so that it can be used in a accessible training platform. We provide a tailor-made rubric. to the LLM to generate scores 1-5.

3.3 Fine-tuning BART

We decided to use the Bart-large-cnn, a classic encoder-decoder model with roughly 406M parameters. We believe that a small language model may benefit from the encoder-decoder architecture given the encoder may empower the model with better abilities of understanding and generating based on context. Moreover, given the Bart-large-cnn has been pre-trained over a corpus of CNN report, we believe the model has some abilities of understanding paragraphs and story flows. The BART model is first fine tuned with an learning rate of 5e-5 and weight decay of 0.01 for 3 epochs and then further fine-tuned for another 6 epochs. It takes a free-tier Colab T4 GPU 4.5 hours to fine tune the BART model for 3 epochs. In our study, we also use a Colab A100 GPU to speed up the research process, which takes 1.2 hour to fine tune BART for 3 epochs.

Model	ROUGE-1	ROUGE-2	ROUGE-L	Perplexity-BART	Perplexity-TinyLlama	LLM-Eval
Original story	NA	NA	NA	1384072.03	15.3	**3.55**$*$
BART baseline	0.136	0.02	0.11	502094.90	-	1.85
BART fine-tuned	**0.221**	**0.053**	**0.197**	1100737.79	-	**2.039**
TinyLlama baseline	0.17	0.024	0.148	-	12.81	1.158
TinyLlama fine-tuned	0.20	0.039	0.172	-	13.97	1.849

Table 3: Comparison of results across models.
Please note that perplexity cannot be compared across models.
$*$ Here the reference ending is itself treated as the generated ending. Benchmarks the original story's score.

In the previous research, ROCStories datasets are often used to train a Story Cloze Test classifier. We wonder if it can go the other way around: can we use Story Cloze Test dataset to fine tune the encoder of our model to generate better outputs? One intuition is that by fine-tuning the encoder with story endings classification task, the encoder may have an enhanced understanding about the relationship such as fluency and coherence between the story bodies and the story endings. With PyTorch framework, we froze all decoder parameters of our fine-tuned BART model and wrapped it inside a BARTClassificationModel. What the BARTClassificationModel is to feed two story paragraph inputs (same story body but one with correct ending and the other with incorrect ending) into the encoder of the fine-tuned BART model, then pass the pooled outputs calculated from the encoder down to a sigmoid function, By calculating the cross entropy loss against the provided correct ending labels, we can fine tune only the encoder of the fine tuned BART model with the Story Cloze Test classification.

3.4 Fine-tuning TinyLlama

For the decoder-only models, we decided to use a TinyLlama 1.1B , a small language model that has been pre-trained on the SlimPajama corpus, which is a multi-source dataset trained with 627B tokens. It is much more diverse than the CNN dataset used for training BART.

Given the ROCStories dataset being relatively small and with compute constraints, we decided to fine-tune the model which is still relatively large using LoRA along with prompt-engineering Appendix A to provide the story input.

(Hu et al., 2021). We decided to target the attention blocks using LoRA, specifically q-proj, k-proj, v-proj and o-proj. Our intention behind this was to improve the model's context sensitive reasoning. We performed multiple iterations of PEFT training using an Nvidia A100 GPU with high RAM, with varied LoRA configs and SFT trainer parameters, given the constraints of GPU memory and training dataset. Good results were seen with LoRA rank of 8 and a learning rate of 2e-4 with a cosine LR scheduler, significantly better than the baseline. With a batch size of 64, the average training time for TinyLlama was around 2 hours on Nvidia A100 GPU with GPU RAM use around 30GB.

4 Results and discussion

4.1 BART

Through the validation loss, we have found that the fine tuned BART model performs the best when it's trained by 2 epochs. The baseline BART model always tried to include the original prompt in the generated output. However, it doesn't copy down the exact prompt; we have observed that sometimes there are slight deviations such as missing punctuations or wrong letters. Moreover, the baseline model may include some not so reasonable CNN reference in its output, such as "Dan's parents were overweight. Dan was overweight as well. The doctors told his parents it was unhealthy. His parents understood and decided to make a change. They are no longer overweight and are now healthy and happy. Click here to follow Dan's journey on CNN iReport.". The fine tuned BART model resolves both of these issues. As you may see from the sample result in Table 2, fine tuned BART model generates a clean story ending without attempting to reiterate the original prompt. We tweaked generate method parameters, such as temperature, top k, and top p, no noticeable quality difference observed in generated story endings. The perplexity score has worsened from the baseline to the fine tuned BART model. We have observed that in both BART and TinyLlama models, the fine tuned ones have a worse perplexity score. Besides, the baseline perplexity scores of both model are actually better than the perplexity scores from the original stories. Therefore, we hypothesize that perhaps lower perplexity is not always better; a story ending may be a good ending in terms of perplexity score but may not be the case from natural human speaker perspective. This also motivates us to use LLm as a evaluation judge to further evaluate model's story ending generation performance. Table 3 shows that compared to the BART baseline, the fine tuned BART has improved on all ROUGE scores by at least 70 percent and has a better LLM-Eval Prometheus Score as well. The BART model with encoder further fine tuned by classification task has unsuccessful results. Random letters rather than meaningful story endings were generated. We hypothesize that fine-tuning for Story Cloze Test Classification interrupts encoder parameters that were good over the contextual embeddings and processing for generation tasks.

4.2 TinyLlama

We experimented with various prompts and found the (Instruction, Input, Response) prompt-template to be most effective with the TinyLlama model, as the model has been built for complex instruction following. While the baseline performance in story ending generation was not great, we saw significant improvements with LoRA fine-tuning, with ROUGE-1 and ROUGE-L scores improving by 3 percentage points. The perplexity increased by approximately 1 point, coming closer to the perplexity score of the original story, exhibiting its fluency in the story generation task. It did however, generate repetitive texts in some cases and was not always very consistent at good story ending generation.

We also saw strong improvements with the LLM-as-a-judge evaluation by Prometheus with the fine-tuned model, with the average score jumping from 1.15 for the baseline to 1.84. The fine-tuned model thus exhibited good generalizability . Although the model could not beat the BART model, the improvements do highlight its ability to improve and it might be further fine-tuned for the story completion task, given more resources at hand.

4.3 Discussion

We propose that the BART model slightly outperforms the TinyLlama model despite its smaller size due to its encoder-decoder architecture, where the encoder is able to build a rich context from the story beginning provided, which helps in the decoding task of generation of story endings. One more factor which might help BART is having a smaller set of parameters to train and given our relatively small training data, it is easier to fine-tune the BART model.

5 Conclusion

We presented two fine-tuned small language models to approach the story completion problem, using well known short story datasets. By treating the task as a text generation problem using the given context, we were able to perform fine-tuning using our chosen datasets. We performed supervised fine-tuning on the BART model and LoRA fine-tuning on the TinyLlama model using a custom prompt, and we were able to see significant improvements in ROUGE and LLM eval scores by Prometheus model as a judge. We observed that the BART 406M model, based on the encoder-decoder architecture slightly outperforms the TinyLlama model 1.1B model, possibly due to the encoder layer, being able to absorb rich story contexts. Our results exhibit the power of small language models to be used for the task of story completion. With the ease of access, trainability and deployment of small language models, we are hopeful that small language models will be useful in real world applications, especially for whom needs to create their in house language models.

5.1 Ethical Concerns

We realize the model may be biased depending on the data that the foundation model has been trained on, as well as the ROCStories data that has been used for fine-tuning it for the story completion task.

We also realize that the model could sometimes generate story endings that are disturbing or tragic, causing stress or shock to the reader. Therefore, sufficient guardrails need to be built around the story endings being generated.

We also do not know the entire extent of the corpus used to train the foundation models used and any intellectual property misused because of that. Appropriate measures need to be taken to safeguard original content from writers.

5.2 Limitations

Since the models have been trained for stories of five-sentence lengths only, they may not generalize to stories of varying lengths. Also, the models may not truly understand the narrative of the story itself and may not be suitable for all audiences due to lack of guardrails.

5.3 Future work

In order to further improve our proposed lightweight story generation models, we would like to incorporate relevant common-sense knowledge graphs in the prompt using tools like (Hogg, 2024)LLM Graph based on named-entities in the story.

We would also like to provide more training data, possibly of stories of varying lengths in order to make the model more generalizable for story completion tasks. Using synthetic data generated by LLMs could also be explored in order to have sufficient training data.

Evaluation metrics have been a pain point during our study. Generating story endings is different from a translation or summarization task. It's more open ended such that two story endings with entirely opposite semantics can be both correct and good candidates. Therefore, evaluation metrics requiring a ground truth label/reference, such as ROUGE and BERTScore, may not capture the full picture. While the perplexity is a reference-free metric, it's lack of granularity evaluating the fluency, grammar, semantics, paragraph coherence of a generated endings. Additionally, in our study, we are unsure if the perplexity is the lower, the better, or the closer to a score calculated from stories by humans, the better. Besides, our LLM as a judge evaluation, Prometheus Score, also requires references. Therefore, more future work around the evaluation metrics need to be done. Potential directions are using Amazon Mechanical Turk or other human based evaluations, and constructing a open source, free to use and accessible, LLM based evaluation framework that doesn't need references.

Acknowledgements

We would like to extend our sincere appreciation to Professor Jennifer Zhu for her guidance, encouragement and for her resourcefulness in exposing us to many tools including Performance Efficient Fine-Tuning like LoRA as well as Quantization, which were immensely helpful in pursuing our task given the limitations of compute resources.

We would also like to thank the Google Colab team for providing free compute resources which were quite helpful in our work.

6 References

References

Jiaao Chen, Jianshu Chen, and Zhou Yu. 2018. Incorporating structured commonsense knowledge in story completion. *CoRR*, abs/1811.00625.

Lewis et al. 2019. Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. *arXiv preprint arXiv:1910.13461*.

Mostafazadeh et al. 2016. A corpus and cloze evaluation for deeper understanding of commonsense stories.

Dylan Hogg. 2024. Llmgraph: A lightweight graph framework for language models. Accessed: December 8, 2024.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2021. Lora: Low-rank adaptation of large language models. *Preprint*, arXiv:2106.09685.

Qingbao Huang, Linzhang Mo, Pijian Li, Yi Cai, Qingguang Liu, Jielong Wei, Qing Li, and Ho-fung Leung. 2021. Story ending generation with multi-level graph convolutional networks over dependency trees. *Proceedings of the AAAI Conference on Artificial Intelligence*, 35:13073–13081.

Seungone Kim, Jamin Shin, Yejin Cho, Joel Jang, Shayne Longpre, Hwaran Lee, Sangdoo Yun, Seongjin Shin, Sungdong Kim, James Thorne, and Minjoon Seo. 2024. Prometheus: Inducing fine-grained evaluation capability in language models. *Preprint*, arXiv:2310.08491.

Nasrin Mostafazadeh, Alyson Grealish, Nathanael Chambers, James Allen, and Lucy Vanderwende. 2016. CaTeRS: Causal and temporal relation scheme for semantic annotation of event structures. In *Proceedings of the Fourth Workshop on Events*, pages 51–61, San Diego, California. Association for Computational Linguistics.

Rishi Sharma, James Allen, Omid Bakhshandeh, and Nasrin Mostafazadeh. 2018. Tackling the story ending biases in the story cloze test. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 752–757, Melbourne, Australia. Association for Computational Linguistics.

Peiyuan Zhang, Guangtao Zeng, Tianduo Wang, and Wei Lu. 2024. Tinyllama: An open-source small language model. *Preprint*, arXiv:2401.02385.

7 Appendix

A TinyLlama Prompt Template

Below is an instruction that describes a task, paired with an input that provides further context. Write a response that appropriately completes the request.

Instruction: Complete the story in a single sentence, based on the story beginning provided.

Input: Serena was planning a surprise for her husband's birthday. She wanted to throw him a party, but his schedule was tough. He would always arrive home at widely different times. To get around it, she worked with his co-workers.

Response:

B LLM-as-a-Judge Scoring Rubric

"criteria":"Does the model successfully generate a response that is a suitable ending to the provided story body, in terms of language fluency, semantics coherence, and story flow",
"score1_description":"The ending is riddled with language errors, is incoherent or disconnected from the story body, and disrupts the narrative flow. It leaves the reader confused or unsatisfied.",
"score2_description":"The ending has noticeable language issues, inconsistencies, or rushed transitions. While it ties some loose ends, it feels incomplete or awkward.",
"score3_description":"The ending is adequately written with minor language errors and mostly logical progression, but it lacks emotional impact or creativity in its resolution.",
"score4_description":"The ending is well-crafted, fluent, and coherent, with a fitting and satisfying resolution. It enhances the story's themes and characters, though it might lack exceptional originality or depth.",
"score5_description":"The ending is flawless in language fluency, beautifully integrates with the story, and delivers a compelling, imaginative, and emotionally resonant conclusion that elevates the entire narrative."