Advanced-NLP Project Report

Pushing the Limits of What a Transformer Can Do - TinyStories

Team: *skill_issue*

Ashwin Kumar (2022101091) — Tanay Gad (2022101043)

November 20, 2024

1 Introduction

Large Language Models (LLMs) are the talk of the hour, revolutionizing the field of Natural Language Processing (NLP) by demonstrating remarkable capabilities in understanding and generating human-like texts. However, their immense size and computational demands pose significant challenges in terms of efficiency, accessibility, and deployment, particularly in resource-constrained environments. This project draws inspiration from the paper `TinyStories: How Small Can Language Models Be and Still Speak Coherent English?`, which seeks to address these limitations by exploring the feasibility of using smaller models for creative language generation.

This report explores the potential of small transformer models using the TinyStories dataset. The project aims to push the limits of what small language models can achieve by examining the trade-offs between model size, efficiency, and performance. We trained a decoder-only transformer model from scratch on the `TinyStories` dataset, adjusting hyperparameters to assess performance based on grammar, creativity, and consistency.

2 Overview

The project aims to re-implement the TinyStories paper from scratch while trying some extra features. Lets have a brief overview of the approach and methodology we followed in the project. To start off, we used the `TinyStories/33M` dataset from `huggingFace`, which is a highly specific dataset curated by the authors for the original paper, more about the dataset details in the coming sections. We then trained a decoder-only Transformer model on the dataset using an Autoregressive Language Modeling technique. Each story in the dataset

is converted into tokens and the model predicts the next token in a sequence given all previous tokens. Once trained, the generated stories are evaluated using GPT-3 through API calls. It rates the stories based on their grammar, creativity, consistency and some other aspects. We have also implemented the TinyStoies-Instruct variant which is an extension of the original TinyStories framework, designed to evaluate the ability of small language models to follow instructions. In this variant, the model is trained not just to generate stories based on a given prompt, but also to follow specific instructions that guide the story's content, style, or structure. We used GPT-2 variants as the baseline models for evaluation of the new stories generated.

In addition to all the work done in the original paper, we have tried some extra approaches for experimentation. The main aspect of the TinyStories paper was the dataset used, so we tried to modify the dataset by replacing certain words with their synonyms. This allows the model to improve its vocabulary and learn more words that can be used interchangeably. The feature aimed to enhance the diversity of words used in the story while still preserving the loss values.

We also tried a fine-tuning approach that tries to work like TinyStories-Instruct, but for a specific topic. To put it more clearly, lets take the word 'dog', we use loss values, similarity scores and a reward function to fine-tune the model in such a way that it only generates dog related stories even without prompting it to do that like in case of TinyStories-Instruct.

3 Dataset Characteristics

The TinyStories dataset consists of short, simple stories aimed at training small language models. It combines all the qualitative elements found in natural language such as grammar, vocabulary, facts and reasoning, but is also smaller, less diverse, and more restricted in terms of its content making it ideal for evaluating compact Transformer models. The stories cover a range of topics, including everyday situations and imaginative tales, while maintaining simplicity and consistency. The stories have been generated by prompting GPT-3.5/4 such that it only uses vocabulary that a typical 3-year-old child would understand.

- **Dataset Size and Composition**: There are approximately 2.1 million stories with an average word count of 70 in the train set.
- **Tokenization and Preprocessing**: We have used the GPT-2 tokenizer from Hugging Face and added end-of-text tokens and padding tokens if needed to each story.

The `TinyStoriesDataModule` class manages the loading and preprocessing of the dataset, including tokenization, chunking, and batching. Initially, the dataset is loaded using the `load_dataset` function from Hugging Face's datasets library,

and each story is tokenized with the `GPT-2 tokenizer`. The tokenized stories are split into fixed-length chunks based on the specified `max_length` to ensure consistency in input size for the model. These chunks are then cached to disk for faster future access. The data is loaded in batches, with padding applied to ensure uniform sequence lengths across each batch. Additionally, attention masks are created to differentiate between padding and actual tokens.

4 Model Architecture

The model relies on a standard GPT-2 architecture implemented using PyTorch and the specific details are as follows :

- **Embedding Layer:** Converts token IDs into dense vectors. The embedding vectors represent the position of each token in a sequence.
- **Positional Encoding:** GPT-2 uses learned positional encodings added to token embeddings to incorporate the order of tokens.
- **Self-Attention Layers:** Each layer has multiple heads of self-attention. Multi-head attention allows each head to attend to different parts of the sequence, capturing context across the sequence.
- **Feed-Forward Networks:** Each transformer layer includes a feed-forward network, which applies transformations to each token's embedding.
- **Residual Connections and Layer Normalization:** Each layer uses residual connections around the self-attention and feed-forward blocks, along with layer normalization to stabilize training.
- **Output Layer:** `GPT2LMHeadModel` includes a language modeling head, projecting the final hidden states to the vocabulary size for next-token prediction or sequence generation.

The key hyperparameters involved in training majorly are :

Hyperparameter	**Value**
Batch Size	32
Max Sequence Length	512
Learning Rate	1e-4
Epochs	1
Warmup Steps	1000
Weight Decay	0.01

Table 1: Major Hyperparameters

The above hyperparameters were kept constant across training for all models, however `the hidden_size`, `n_layers` and `n_heads` were changed for every model (as done in the original paper).

1. **hidden_size** : The hidden_size refers to the dimensionality of the hidden states in each transformer layer. It determines the capacity of the model to capture complex patterns in the data.

2. **n_layers** : This refers to the number of transformer blocks stacked on top of each other. Each block consists of a self-attention layer followed by a feed-forward network.

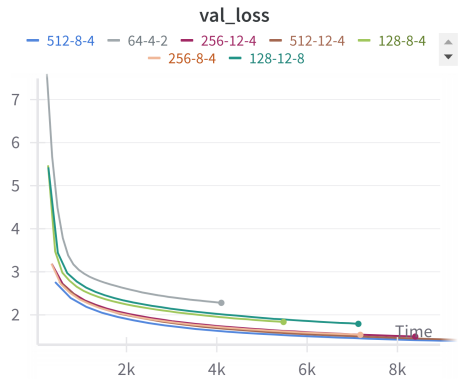
3. **n_heads** : The number of attention heads in each self-attention layer. They are distributed across the hidden_size, and each head attends to a different subspace of the sequence.

We used the institute GPU cluster (`ADA`) for training. The models were trained for 1 epoch each due to compute constraints. However, it didn't have much impact as the loss values already started to flatten by the end of first epoch. The loss values were close enough to the values in the original paper.

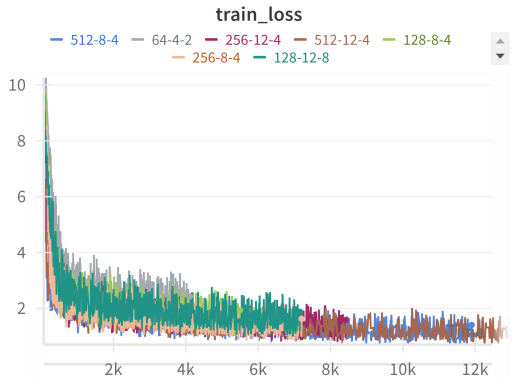
5 Training

The training process in our implementation follows a typical training setup. The model is initialized with parameters such as number of layers, attention heads and random weights. During training, the model processes batches of input sequences, and learns to predict the next token in a sequence to generate coherent and contextually appropriate text. We try to minimize the cross-entropy loss between the model's predicted token probabilities and the true next token in the sequence. Backpropagation is used to compute the gradients of the loss with respect to the model's parameters. The optimizer (`Adam Optimizer` in our case) updates the model's weights to minimize the loss during training.

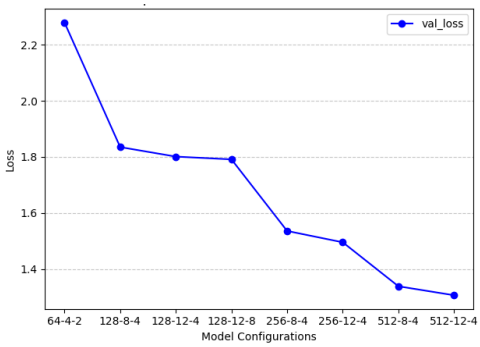
After training, the model can generate text by autoregressively predicting the next token, using its own previous predictions as context until a stopping criterion (e.g., maximum length or end-of-sequence token) is reached.



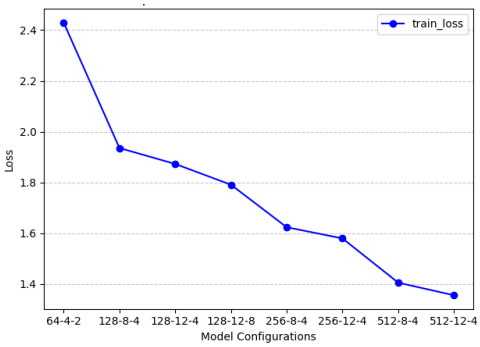
(a) Evaluation loss across epochs



(b) Train Loss across epochs



(c) Evaluation Loss across models



(d) Train Loss across models

6 TinyStories-Instruct

TinyStories-Instruct is a specialized variant of the TinyStories dataset designed to evaluate and train models on instruction-following tasks. Unlike the standard TinyStories dataset, which focuses on generating coherent, creative, and grammatically correct stories, TinyStories-Instruct incorporates prompts and instructions within the story narratives, requiring models to generate outputs that adhere to specific directives or contexts provided in the prompts.

The training process involves certain changes as compared to the previous way. We provide the model with a list of words and a sentence that should appear somewhere in the story. There is also a list of features (possible features: dialogue, bad ending, moral value, plot twist, foreshadowing, conflict)along with a short summary (1-2 lines) of the story. The model then generates a story based on the given instructions. All of the above features are already included in the `TinyStories-Instruct` dataset available on `huggingFace`.

7 Gemini Eval

The evaluation of language models usually rely on structured evaluation datasets in the form of a task where the output of the model has to match a given answer. However this approach isn't very effective to judge tasks like story generation. Therefore we use the already heavily trained LLM's to do the job for us. To be specific, we use `Gemini` for the evaluation. It was chosen due to the free API credits it provides.

Once the model generates the output, 15 stories are randomly selected from the generated dataset. We then insert a special symbol *** somewhere in the beginning of the story. The evaluation process involves giving the model a specialized prompt that is intended to guide the evaluation. This prompt typically includes instructions that specify what the model is expected to return in response to the input. The special prompt is designed to test specific aspects of the generated stories, such as creativity, consistency, or narrative coherence. Based on the prompt instructions and the part prefix to the special symbol, `Gemini` scores the later part of the story on its grammar, consistency, plot-sense from 1 to 10. For every model, the average across all the 15 stories is computed and is used as the score for that particular model.

The evaluation method for TinyStories-Instruct remains the same just with some addition to the prompt sent to Gemini specifying the details about instruction following.

8 Performance Analysis

8.1 First Glimpses

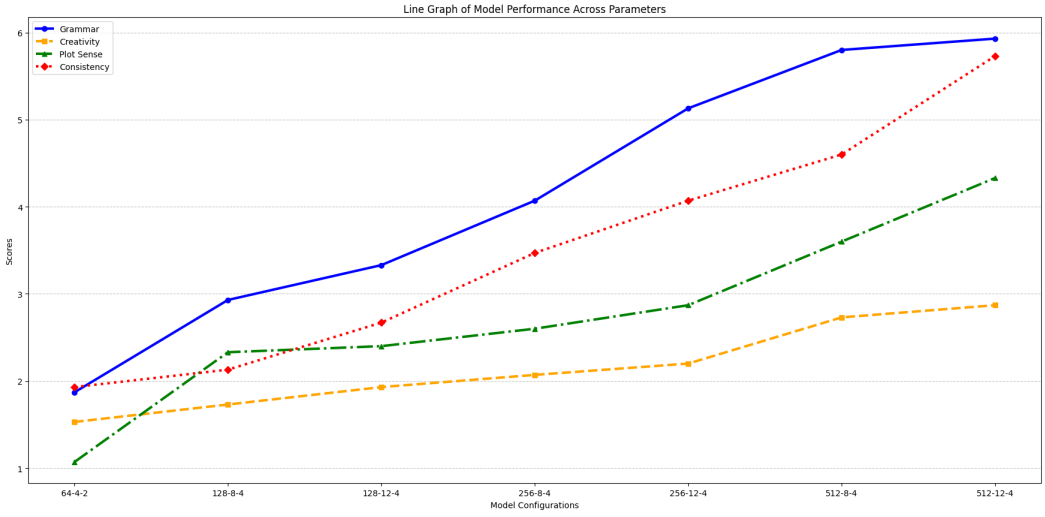


Figure 2: Enter Caption

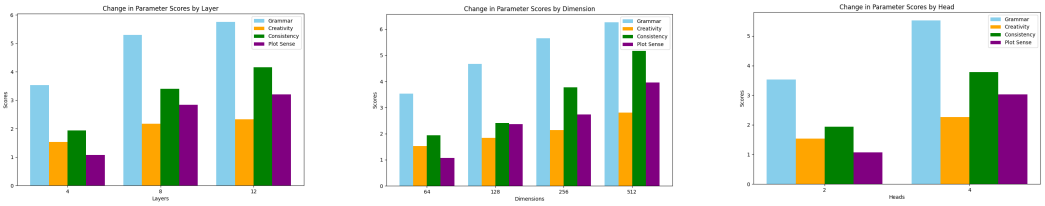
The evaluation method using Gemini gives us a good idea about the quality of the stories generated quantitatively. Now we will try to capture the relation between the scores and model parameters by observing certain trends. The exact values can be seen in Table-3.

- The first obvious conclusion that can be drawn is the consistent improvement with decreasing evaluation loss as can be seen from the graphs above.
- The grammar scores are significantly better as compared to the other parameters for all model sizes with its slope being the flattest among all. However, we can still see the increase in score as the model size and parameter count increases. So its easy to conclude that grammar is relatively easier to learn and after a point the grammar scores tend to plateau.
- Consistency and creativity seem slightly tricky to handle. The smaller models do not get good scores, and the slope also increases significantly with the model size, which gives us an idea that the model size, particularly the depth of the model (number of layers) plays a key role in maintaining the context.
- Creativity values remain relatively low even for the best models with 81M parameters. This can be due to the concise and limited nature of the dataset used for training.
- Plot sense while maintaining consistency and coherence comes with increasing both the hidden-dimension and number of layers to capture more information for long.
- Instruction following by theory should depend on global-attention to capture all the instructions properly. This can be verified with the fact that it depends quite heavily on the number of layers.
- Our trained models perform a lot better than the pretrained GPT-2 model in terms of scores. This also strengthens out claim of SLM's being very effective on targeted concise datasets like TinyStories.

Configuration	Grammar	Creativity	Consistency	Plot Sense
64-4-2	1.87	1.53	1.93	1.07
128-8-4	2.93	1.73	2.13	2.33
128-12-4	3.33	1.93	2.67	2.40
256-8-4	4.07	2.07	3.47	2.60
256-12-4	5.13	2.20	4.07	2.87
512-8-4	5.80	2.73	4.60	3.60
512-12-4	5.93	2.87	5.73	4.33
512-12-4	5.93	2.87	5.73	4.33

Table 2: Comparison of Model Configurations Across Metrics (Grammar, Creativity, Consistency, and Plot Sense).

Note : The evaluation scores by Gemini are significantly lower as given by GPT-4. We observed this using manual prompting for some stories on GPT-4. However due to its paid API credits, we switched to Gemini.



(a) Enter Caption 1 (b) Enter Caption 2 (c) Enter Caption 3

Figure 3: Combined Caption for All Subfigures

In summary, hidden dimensions strongly impact consistency and creativity, layers are critical for consistency and instruction-following, while attention heads play a moderate role in boosting creativity and maintaining global coherence.

8.2 Reference Examples

To get an idea of how our models performed, lets analyze the story generation with an example. Consider the following story from the training set :

"Hi Jane, have you seen Alice? I can't find her anywhere," said Jack. Jane looked up from her book. "I saw her near the big tree. She said she was on a secret mission." Jack's eyes lit up. "A mission? What kind?" "I don't know," Jane said. "But she looked excited. Maybe she's solving another mystery." Jack ran to the tree. There, he found a note from Alice: "Jack, I'm looking for treasure! Follow the red ribbons if you want to help. Hurry! Alice." Jack smiled. He spotted a red ribbon tied to a bush

~~and followed it. The next was on a tree branch. He ran deeper into the woods, eager to catch up with Alice and join her adventure.~~

The blue part is given as the input prompt to the model. Our model generates stories based on the highlighted part. Lets compare the outputs of certain models :

```
"Hi Jane, have you seen Alice? I can't find her anywhere", said Jack.
*** Jill was very sad. She wanted to find her friends, so she decided
to go to the park. When they arrived, they saw a big slide. It was
so big that she got excited. She wanted to go down the slide, but
she was too scared to climb. Jack started to climb the slide. He
was very high and he could reach the top. He was so proud of himself.
Jill was so proud of herself. She had found her friends and they were
all very happy.
```

This was the output of the model with around 8.3M parameters. It seems fine, however we can do much better. So lets try out with the model of size 41M parameters.

```
"Hi Jane, have you seen Alice? I can't find her anywhere", said
Jack. *** Jane was tensed. She wanted to know what to do. "Let's
go and find Alice." said Jack. So they went to the park and looked
around. They saw a big tree with Alice swinging on a swing. Jane
was so excited. She said, "Let's go and swing with Alice." They had
so much fun swinging together. They laughed as they swung higher and
higher. When they were done, they said goodbye to the swing and went
home. Jane was so happy to have found Alice and a new place to play.
```

We can see a clear improvement in the story's language, it makes more sense. But one common thing among both the stories generated by out models is that they clearly outperform or at least perform as good as GPT-2 which has significantly more parameters than them, around a factor of 500.

Another crucial thing to note is the fact that the model did not just learn the initial story and both the plots are significantly different from the original story in the train dataset further showing the competence of our model.

8.3 TinyStories-Instruct Analysis

Now lets try to analyze the results of TinyStories-Instruct. We have used Gemini for its evaluation as well. The prompts sent to Gemini have been slightly modified to take car of the instruction following details. Below are the instruct values of our models :

NOTE : Due to the limited compute available, we have only trained our models for 1 epoch which slightly affects the performance. So for complex tasks like instruction following the model needs to be trained more. Threfore, the TinyStories-Instruct results might not be as good as in the paper, however it still performs the given tasks till some extent showcasing its capabilities that it can perform much better if trained exhaustively.

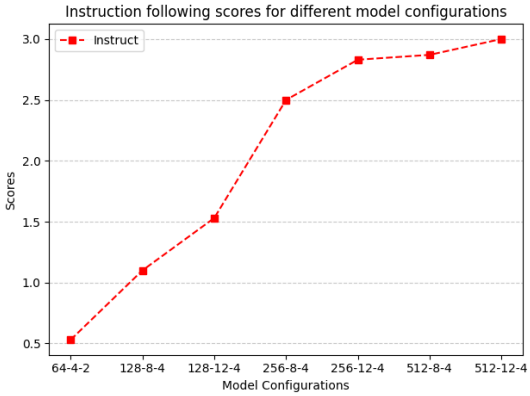


Figure 4: Instruct scores vs Model-size

Below is a sample example showing the results of the instruction model based on certain prompts. We used our biggest model `512-8-4` for this task. We can see that it does the task decently well and can be improved with further training.

Words: come, road, sad

Summary: A bus becomes jealous of a shiny new car and undergoes an operation to go faster, becoming good friends with car making everyone in the town happy.

Story: There was a bus in town. It was slow. One day, a new car **comes**. The car was fast and shiny. He was **sad**. So the bus went to the doctor. The doctor felt **sad** for the bus and thought to help. The doctor made him faster. The bus became faster than the car. He and the car became friends now. The town was also happy because they became friends.

8.4 Knowledge, reasoning and context-tracking

Next, we assess the capabilities of the different models on three additional types of prompts:

- **Factual prompts,** which test the models' knowledge of common sense facts.
- **Reasoning prompts,** which test basic reasoning abilities, such as cause and effect and elimination.
- **Consistency (context-tracking) prompts,** which test the models' ability to maintain coherence and continuity with the given context, such as the names and actions of the characters, the setting, and the plot.

We rate each model and prompt in three tables and color-code them according to their success (green), failure (red), or partial success (yellow). The results show that as the embedding dimension and the number of layers increase, the performance in regards to all three categories improves. The models with higher embedding dimensions

and more layers tend to generate more accurate, relevant, and natural continuations, while the models with lower embedding dimensions and fewer layers tend to generate more nonsensical, contradictory, or irrelevant continuations. This suggests that the embedding dimension is more crucial for capturing the meaning and the relations of words, while the number of layer for capturing long-range dependencies.

8.4.1 Context-tracking

Message	64-4-2	128-8-4	256-8-4	256-12-4	512-8-4	512-12-4
Hi Jane, have you seen Alice? I can't find her anywhere", said Jack.	The girl and her mom and said, "I'm sorry"	Jack was so excited. He wanted to see what was wrong.	Jane smiled and said, "I'm sorry, Jack. I'm sorry. I didn't know."	Jill was very sad. She wanted to find her friends, so she decided to go to the park.	Jane was so excited. She wanted to know what to do.	Jane was sad. She had lost her best friend.
Max had two dogs. One was white and the other was black. Max walked up the street and saw a kid with a dog. He told the kid, "I see you have a Brown dog. I also have	a big hug.	a dog. Max was very happy.	a ball.	a dog. Do you want to play with me?	a dog. Do you want to play with me?"	a big dog." The kid was scared. He did not like dogs. He did not like dogs.
Anne had a piece of candy in her left pocket and a piece of chocolate in her right pocket. Anne's mom asked her, "Anne, what is that you have in your left pocket?"	"I want to go to the park. I want to the park?"	Her mom said, "I'm sorry, Emma. I'm sorry, I'll have a new toy."	Her mom said, "It's a surprise."	"I'm sorry,'. I'm sorry, I'm sorry,".	Anne smiled and said, "I'm sorry, Mom. I was just looking for candy."	Anne smiled and said, "I'm looking for a piece of candy."
Alice had both an apple and a carrot in her bag. She took the apple out of the bag and gave it to Jack. She reached into the bag again and took	it."Look, I'm so pretty,"	it to the park.	it home.	the carrot.	a bite. It was delicious!	the carrot. Jack was very happy.
Alice and Jack walked up the street and met a girl in a red dress. The girl said to them, "Hi, I'm Jane. What are your names?"	The girl said, "I'm sorry. I want to help you."	The girl said, "I'm going to be a princess.	Lily and Jack smiled and said, "I am Lily. I am a princess.	The girl said, "I am Lucy. I am the best friend ever.	Jill smiled and said, "My name is Jane. I am a princess.	Jill smiled and said, "My name is Jane. Jane smiled and said, "I am a princess.
Diva was hungry, and wanted to bake a cake, but she didn't have any sugar at home, so she decided to go ask around. She started walking and met a squirrel. She asked the squirrel, "Would you happen	The bird was so happy and she wanted to be careful	and I't be careful," The squirrel said, "Yes, I will help you."	if you want to help me find the sugar ?	? The squirrel said, "Yes, I will help you.'	to help me ?	to help me ?

Table 3: Performance of different model on Context-tracking Prompts

8.4.2 Reasoning

Message	64-4-2	128-8-4	256-8-4	256-12-4	512-8-4	512-12-4
Lily likes cats and dogs. She asked her mom for a dog and her mom said no, so instead she asked	her mom. "Yes, mom. I have to play with the little dog."	her mom for help. Her mom said yes, but she said she would be careful and not have fun.	her mom. Her mom says, "No, I don't want to play with you.	her to stop. Her mom said, "No, Lily, you can't have the dog.	her to leave. "No, I don't want to leave you," her mom said. "You have to stay here and be nice to the cats.	for one. Lily was sad. She wanted to have a new dog, but she knew she had to be careful.
Jack told Mary, "If you give me your banana, I'll give you my apple". Mary gave Jack her Banana so	he to the car and said, "I'm sorry, I will be careful."	he could help. They went to the store and bought some yummy food.	he could give her the banana. Mary was so happy. She thanked him and said, "Thank you, Jack!"	he could give her a big hug. Jack was so happy that he gave the banana to her.	she could eat it. Jack was so excited. He thanked her and said, "You're welcome. I'm glad you had enough to give you your apple."	she could eat it. Jack was so happy. He thanked Mary and said, "Thank you, Mary. I'm so glad you had a good day." Mary smiled and said, "You're welcome, Jack. I'm glad you like it."
On weekends Jack went to visit his grandmother whereas on weekdays he would go to school. Last weekend, when Jack was on his way to	the park. He was so excited to see the park.	school, he saw a big box of the box. He was so excited that he ran to the box and asked his mother if he could buy it.	school. He was very excited to see the world around.	school, he saw a big, scary monster. He was scared and ran away.	his grandmother's house.	school, he saw a big box of crayons. He wanted to draw a picture of himself.
Lily and Ben were having an argument. Ben said that cake is much better than ice cream and Lily said that	"Yes, I can play with it?" Lily said.	they can make a mess.	they should be careful.	she should not have a cake.	it was too much for him.	she should have a slice of ice cream.
Jack's mother was not home, and his father was at home. When Jack came home, he said hello to	the park.	his mother.	his father.	his mother.	his father.	his mother. "I'm sorry, mother. I'm sorry."
Both Ben and Lily wanted cake. Father said that there was only one piece of cake left. They	had a big smile and a big box.	both wanted to make the cake.	had to go to the store and buy some cake. They went to the store and bought some cake.	said they could not have any money.	had to wait until they were get another cake.	had to wait until it was ready.

Table 4: Performance of different model on Reasoning Prompts

8.4.3 Factual

Message	64-4-2	128-8-4	256-8-4	256-12-4	512-8-4	512-12-4
Alice was so tired when she got back home so she went	to the park.	to the park.	to bed. She was very sad.	to bed.She was very tired	to bed. She had been so tired and had been playing all day.	to her room and lay down on the floor. She was so tired that she fell asleep.
Jack and Lily saw a rainbow after a rainy day. They were amazed by the colors. Jack said, "Look, Lily. A rainbow has	a big tree.	a rainbow.	a rainbow!	a rainbow in it. It is so pretty."	a rainbow in the sky."	a rainbow
Jack and Lily liked to watch the moon at night. They noticed that the moon changed its shape every night. Sometimes the moon was big and round, and sometimes it was	a big, and a little girl.	the most beautiful.	like a star.	very special.	too big.	small.
Jack wanted to read a book, so he went to	her mom.	the store. He asked his mom and dad, "Can I have a book?"	the store and bought a book.	the park.	his room and looked for a book.	his room and opened the book. Inside was a big bookcase full of books.
Can cows fly?", Alice asked her mother.	She said, "I'm sorry, but I want to play with you.	"Yes," she said.	"Yes, I can.'	Yes, I'm sorry, I was not sure.	Yes, I can.	"Yes, I can.
It was winter and cold outside so his mother told him, "You should	be careful.	always listen to her mother and not touch it.	be careful.	be careful.	be careful.	not go outside and play.

Table 5: Performance of different model on Factual Prompts

8.5 Rouge Score Analysis

The ROUGE score is used to evaluate the quality of summaries or text generation tasks. It compares the overlap between n-grams in the generated text and the reference text(s). The ROUGE score mainly focuses on recall, measuring how much of the reference content appears in the generated output. It consists of several variants, such as ROUGE-N (based on n-grams), ROUGE-L (longest common subsequences), and ROUGE-S (skip-bigrams). Therefore we use the ROUGE score to check the similarity between the stories generated by our models and the original data. It is done on the Training-dataset.

We measure the diversity of the stories quantitatively using word and n-gram overlap. We inspect the overlap of words and n-grams between different stories generated by the models, and compare them with the overlap in the dataset. We find that the models' generations have a very low overlap with the dataset, indicating that they are not repeating the same words or phrases.

We perform the following experiment: We randomly pick 100 stories from the training dataset, we cut each story in the middle, keeping roughly the first 40%, and use it as a prompt. We ask the model to generate a completion from each prompt. The ROUGE score is then calculated between both the original and generated stories. The lower the value, lower is the sentence overlap meaning less repetition.

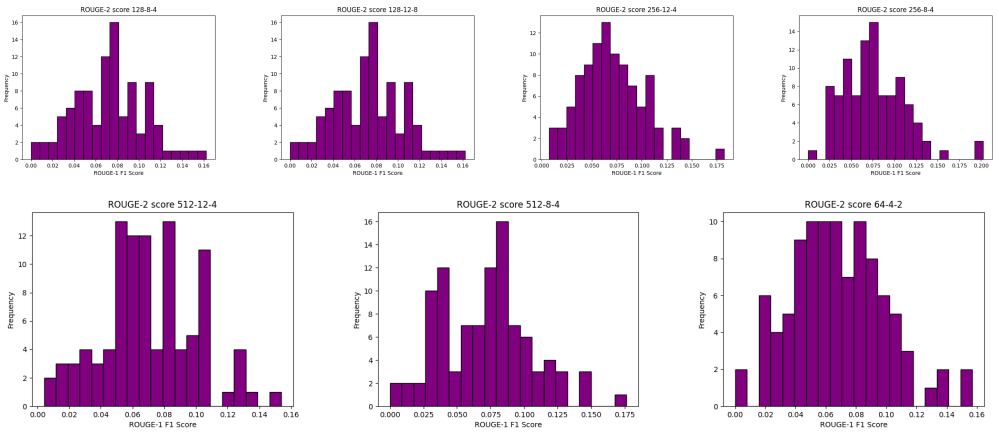


Figure 5: Rogue2 (precision) score between the model's completion and the original story from the same beginnings (we select 100 from the training dataset).

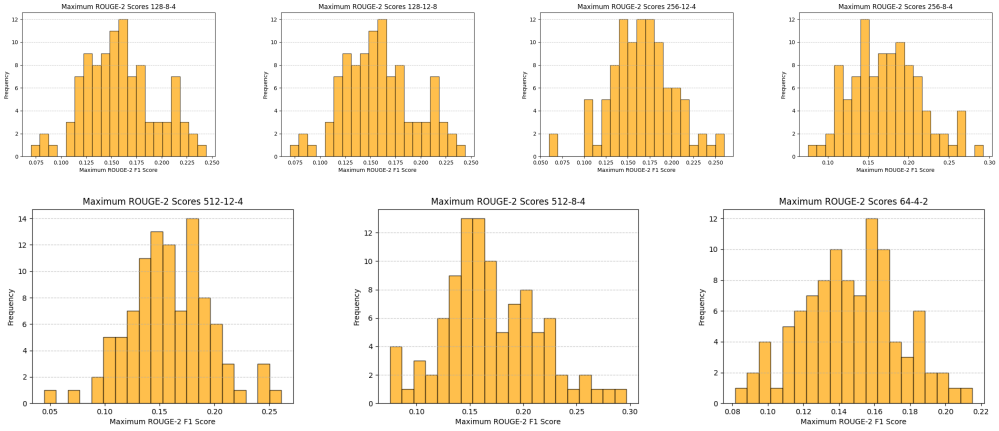


Figure 6: Maximum Rouge2 score (fmeasure) similarity between the 100 generated stories for each model. Original model means the ones generated by GPT-3.5

9 Interpretability

9.1 Analysis of Attention Heatmaps

The provided heatmaps represent attention patterns from different attention heads while processing the input text. Each heatmap is a 2D matrix where:

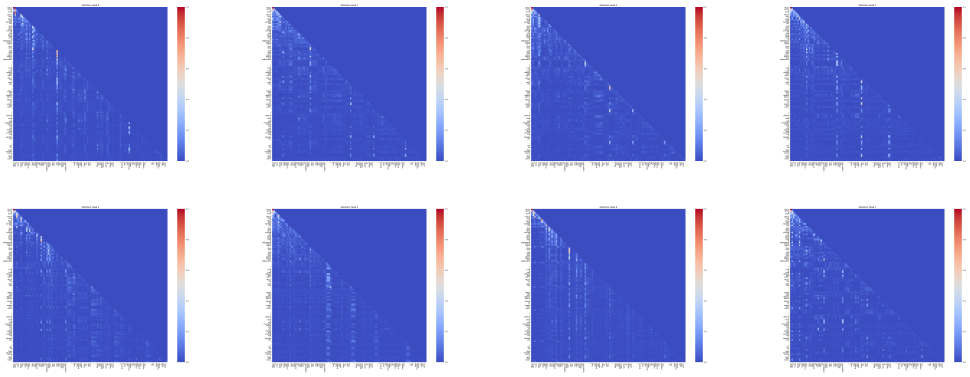


Figure 7: Maximum Rouge2 score (fmeasure) similarity between the 100 generated stories for each model. Original model means the ones generated by GPT-3.5

- The **x-axis** represents the input tokens.
- The **y-axis** also represents the input tokens.
- The values at each position in the matrix (i, j) represent the amount of attention the token at position i pays to the token at position j.

9.1.1 Head 0

- **Observation:** The attention seems concentrated along the diagonal, especially at the start of the sequence (e.g., *"Once upon a time"*). This indicates that the head focuses on tokens in close proximity.
- **Interpretation:** This behavior is likely indicative of **positional attention**—the model pays attention to nearby tokens, which is typical for capturing short-term dependencies.

9.1.2 Head 1

- **Observation:** The attention is more spread out compared to Head 0, with no clear focus on the diagonal or isolated regions. There are weak connections distributed across the entire sequence.
- **Interpretation:** This head appears to exhibit **semantic attention**, potentially capturing relationships between semantically related words, even if they are farther apart in the sequence.

9.1.3 Head 2

- **Observation:** The attention is again partially diagonal but shows isolated, high-intensity points of focus across the sequence. These strong off-diagonal connections suggest attention to specific long-range dependencies.
- **Interpretation:** This could indicate **long-range semantic attention**.

9.2 General Insights

The concentrated diagonal patterns likely indicate attention heads tuned to local context, while spread-out patterns and off-diagonal points indicate the model's ability to connect related words across sentences. In models of smaller size both semantic and positional attention heads seem to be prevalent.

10 Vocab Diversification Using TinyStories-Syn

10.1 Introduction

The models we trained performed as expected, but if the sample space of stories on the same topic was increased, the model would use the same word for a specific thing. For instance, lets take the word 'happy', everytime the model wanted to show an instance of happiness in the story, it would use the word 'happy'. But we know as a fact that there are a high number of synonyms in English which can be interchangeably used, like the word 'happy' can be replaced with words like 'joy', 'glad' etc.

Therefore, to inculcate the use of synonyms in story generation, in addition to training with the original dataset, we created a version of TinyStories with a diversified vocabulary. We achieved this by augmenting the dataset with synonyms, replacing certain words with alternatives to introduce variety without altering story meaning.

10.2 New Dataset Creation

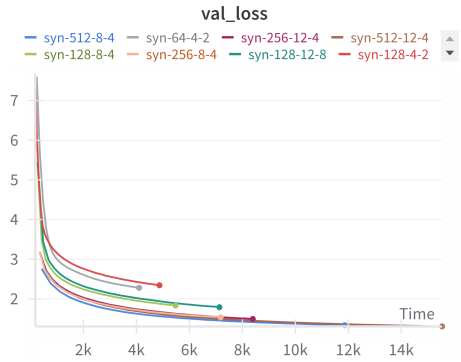
We started off with downloading the original TinyStories dataset from HuggingFace in `.txt` format and converted it into `JSON` format. Then we implemented a synonym substitution mechanism using `NLTK WordNet`. The method identifies content words (nouns, verbs, adjectives, and adverbs) within each story and replaces a percentage of them (30%) with their synonyms, selected randomly from WordNet. This ensures the augmented stories retain their original meaning while introducing lexical variation. This synonym-replacement is only done for 50% of the dataset. The new `JSON` is then shuffled and a new dataset is created. The choice for the percentage of dataset and words to be replaced was done by trying out some other higher and lower percentages of change. After trying out some values we decided to go with the current ones.

We also implemented our own `custom_data_loader` function that can be used in place of the original `data_loader` Pytorch function used to load models directly from HuggingFace. However, for the sake of simplicity we uploaded the datasets on HuggingFace by the name `ashh18/synonymTiny`.

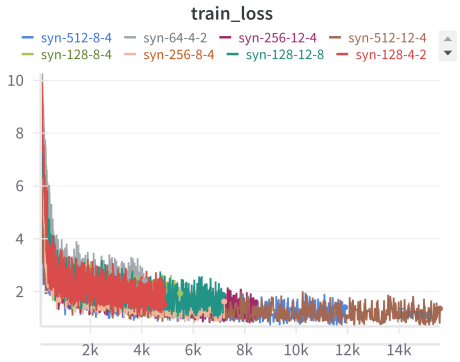
Note : The word replacement using `WordNet` is effective but has some flaws. It does not take into account the overall context of the sentence and hence can occasionally disrupt grammatical structure or coherence, particularly in complex narratives. But as TinyStories is relatively very less complex and we apply changes only for certain aspects of the sentence (like noun, verb, adjective), therefore the effects are not expected to be drastic.

10.3 Model Architecture and Training

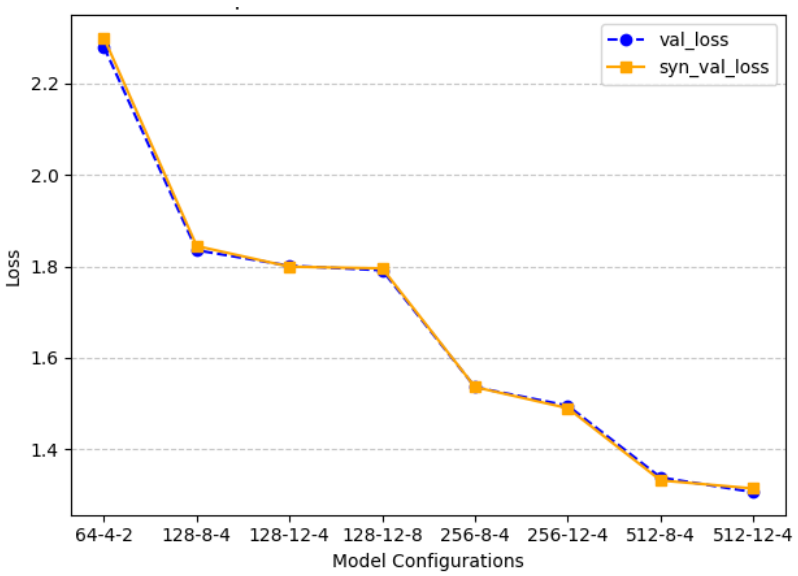
The model architecture, hyperparameters remain exactly the same. The training step for `TinyStories-Syn` however involves using our newly generated custom dataset. The loss values remain consistent with their counterpart models that were trained without synonyms. Therefore, one would not expect much deviation in factors like grammar, consistency which can be seen in the evaluation values. But on expected lines, the vocabulary diversity sees an increase due to the added synonyms. The creativity score also shows marginal increment.



(a) Evaluation loss across epochs



(b) Train Loss across epochs



(c) TinyStories vs TinyStories-Syn Evaluation Loss

10.4 Evaluation and Analysis

As mentioned above as well, the overall model does not change much, only the vocabulary gets enhanced. This change is however difficult to notice and quantify due to the less amount of stories used as queries to prompt Gemini for evaluation (due to restricted compute and limited API credits).

To cater this issue, we handpicked stories that were to be sent for evaluation. We marked down the adverbs, nouns and adjectives in the stories. Then using `NLTK-WordNet` we generated the possible synonyms of around 30% of those words. The selected words were then wrapped in a special symbol `< >`. The prompt was modified to specially mention those words and include a list of synonyms for each of the selected word. Gemini was then instructed to rate stories more that involve correct grammatical and syntactical use of the synonym words over the original words. This way, we were able to evaluate our TinyStories-Syn. The results were quite satisfactory and the vocabulary-diversity showed improvement.

Configuration	**Grammar**	**Creativity**	**Consistency**	**Plot Sense**	**Vocab-div**
`64-4-2`	1.87 / 1.85	1.53 / 1.60	1.93 / 1.85	1.07 / 1.10	0.79 / 1.15
`128-8-4`	2.93 / 3.00	1.73 / 1.80	2.13 / 2.10	2.33 / 2.31	1.67 / 2.82
`128-12-4`	3.33 / 3.30	1.93 / 2.00	2.67 / 2.80	2.40 / 2.44	1.89 / 3.02
`256-8-4`	4.07 / 4.10	2.07 / 2.05	3.47 / 3.40	2.60 / 2.56	2.07 / 2.83
`256-12-4`	5.13 / 5.05	2.20 / 2.24	4.07 / 4.10	2.87 / 2.85	2.36 / 3.40
`512-8-4`	5.80 / 5.57	2.73 / 2.80	4.60 / 4.55	3.60 / 3.55	2.47 / 3.54
`512-12-4`	5.93 / 6.10	2.87 / 2.75	5.73 / 5.70	4.33 / 4.41	3.14 / 4.10
`512-12-4`	5.93 / 5.95	2.87 / 2.98	5.73 / 5.91	4.33 / 4.52	3.07 / 4.25
Comparison	**Similar**	**Better**	**Worse**	**Similar**	**Better**

Table 6: Comparison of Messages Across Configurations.

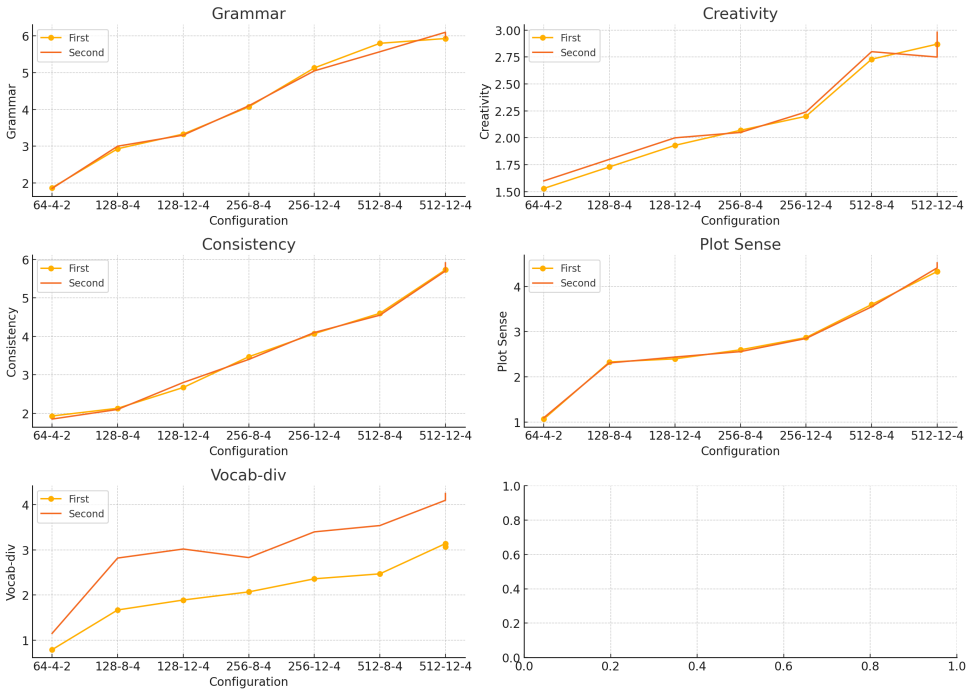


Figure 9: TinyStories-Syn vs TinyStories

The prompt for Gemini was changed slightly to give `more score to non-trivial words.` and due to the enhanced vocabulary of the synonyms model, it clearly overshadows the normal model in terms of `vocab_diversity`. It also rated the new model's creativity marginally higher. The other parameters however remain similar in terms of scores on expected lines.

11 TinyStories-Reward vs TinyStories-Instruct: Can we do better for small instructions ?

Lets say the scenario where we only want the model to generate stories on cats. One potential way is to prompt the model using TinyStories-Instruct and instruct it to only generate cat stories. But as mentioned above, we wanted to try something new for this. So we come up with TinyStories-Reward.

11.1 Introduction

TinyStories-Instruct showed how SML's can follow instructions given to them for story generation. It involved giving the model certain prompts before generation based on which the model generated the suitable stories. However, creating this particular variant of the model also involves high computation and training which can sometimes be

TinyStories-Reward an approach that generates targeted stories on a certain topic by using a reward function to determine how desirable the outputs are, guiding the training without explicit supervision. Its like algining the model on a topic which makes the model generate only the stories related/based on that topic without the need of any prompt, unlike TinyStories-Instruct.

11.2 Basic Implementation

Dot products can be used to calculate the similarity between 2 things. We will use this to minimize our loss function. Lets begin with creating the embedding for the word 'cat'. Then once a story is generated, its sentences are converted into embeddings. Similarity is calculated between the two. Based on the similarity, we will calculate the loss value and backpropagate accordingly.

We begin by initializing the output_ids tensor with the prompt, padded with end-of-sequence tokens. The model then generates tokens sequentially, updating the output_ids tensor one token at a time, repeating this process multiple times. To calculate the joint probability of the entire generated story, we accumulate the log probabilities of the sampled tokens. After completing the generation, we evaluate each sequence using a reward function based on embedding similarity.

The final loss is computed as the sum of the negative log probabilities of the generated story, weighted by the reward. Since higher log probabilities indicate more likely sequences, their negatives serve as a measure of cost. By weighting this cost with the reward and applying back-propagation, we guide the model to improve its outputs.

11.3 First Analysis

The above steps were quite easy and seem too easy to be true. And that's exactly the case when we do the analysis of the stories generated. We can easily observe that :

- The model outputs the word 'cat' itself way too much.
- The stories have degraded in terms of grammar, consistency etc.

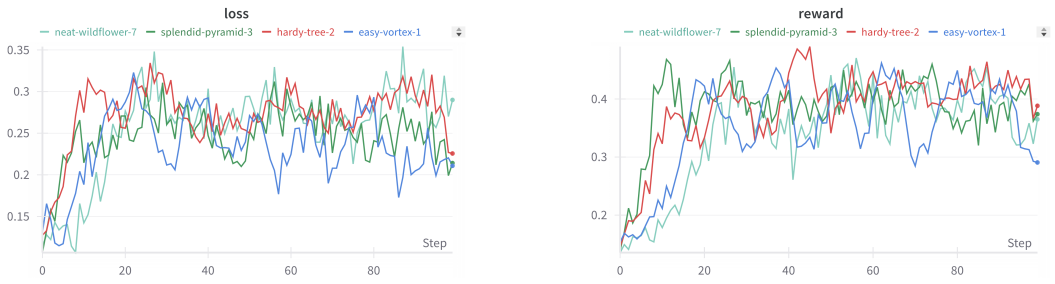
All this makes sense, because due to the nature of the reward function, for every story the model will simply try to make the outputs more and more similar to the word 'cat' which would ultimately lead to the model only outputting the word 'cat. But this is not right. We need to take maintain these features.

11.4 Controlling model divergence

To prevent the model from overfitting to the reward function like generating "cat" repeatedly, we tie it to a fixed reference model. This helps change its behavior to focus on cat stories without breaking its general capabilities. We measure the difference in output distributions using KL-divergence to control the alignment.

KL_FACTOR determines the weight of KL-divergence. A high value limits optimization away from the reference model, while a low value weakens alignment control, risking the model breaking (e.g., overfitting to "cat").

11.5 Final Analysis



(a) Loss graph

(b) Reward graph

We define a new reference model which is our pre-trained `512-8-4` config model. The loss curve doesn't decrease like in the previous case without kl-divergence. But that's expected as we are now penalizing deviations from the reference model and after a point it will get difficult to only generate stories about cats while staying close to the reference model. And for the reward values, we can see that the upper cap that is now hit is much lower than in the previous case because now the model penalizes for just generating the word 'cat' as it makes the story unaligned with the reference model.

11.6 Instruct vs Reward

Now comes the main part, can our method even compare with TinyStories-Instruct ? One very obvious conclusion we can infer is that TinyStories-Reward does not take into account the context of the input prompt to which the similarity is calculated. So, it is only suited for small one-word reference texts. So, for big and complex instructions TinyStories-Instruct is way better than this. But lets consider for the single word case :

- We prompt the Instruct model to contain the word 'cat' and begin with the phrase 'Once upon a time there was'. This instruction is clearly vague for the model to generate good stories. Lets verify this with results.
- For the Reward model we set the reference word as 'cat'. Then generated stories revolving around cats.
- We can see from the table below, the stories generated by each model and rate them as Good, Okay, Bad.

NOTE : Both models are working on out best configuration `512-12-4`. We will show a couple of outputs from both the models which would give us an idea of the capability of the Rewards model.

TinyStories-Instruct	TinyStories-Reward
"Once upon a time there was a very special cat. A cat was a cat, all different colors! The cat had a secret. It had a very big, original hat. The cat wanted to learn how to use the hat. That made it very happy! The cat started to learn lots of new things. But the cat was very smart."	"Once upon a time there was a small cat. The cat was very eager. It wanted to go to town and explore more. Every morning the cat put on a collar and set off. One day the cat went to town. It stopped and paused to watch the people. They were wearing colorful collars so the cat could see what they were doing. The people in town stopped and asked the cat, ‘‘Why are you looking so eager every day? You should come see what other people are doing.’’ The cat smiled, ‘‘Yes, I am. I enjoy discovering new things and exploring the town.’’"
"Once upon a time there was a cat, a cat named Sam, living in the cat, because she loved to explore. One day Sam was looking around very weird. She noticed that the cat was shaking and she couldn’t shake it. Sam felt so curious, she wondered why the cat was shaking and couldn’t understand what was happening. She looked to her owner and said, ‘‘Why does your cat do this funny?’’ Her owner laughed and said, ‘‘Well, I’m not sure why. The cat is looking for something fun to do that I was worried. Maybe if the cat found her way to another fun place a few feet away from the floor, we can come back and give her a nice shake.’’"	"Once upon a time there was a very hairy cat. The cat always wanted to find a special place and one day it came across a beautiful, dark spot. The cat was so intrigued that it went and looked closer. There in the dark corner it saw something that made it smile. The sun was shining and the cat liked that it would smile at the sun. The cat kept smiling and his furry purring and he knew something special was in the dark. He went to find more light, so he could show the sun how much he liked it. He could bring the sun sunshine to the dark spot and keep it company. The cat was so happy and the sun was happy too!

Table 7: Performance of different models on Factual Prompts

12 References

1. Ronen Eldan and Yuanzhi Li, "TinyStories: How Small Can Language Models Be and Still Speak Coherent English?" arXiv preprint arXiv:2305.07759, 2023.

Available at: https://arxiv.org/abs/2305.07759.

2. Aligning TinyStories https://philliphaeusler.com/posts/aligning_tinystories/.