LEVERAGING LARGE LANGUAGE MODELS FOR CREATIVE STORY GENERATION

Yuetian Chen, and Brendan Capuzzo
Group U1 - Machine Learning and Optimization
CSCI 4968 - 01 - Spring 2023
Department of Computer Science
Rensselaer Polytechnic Institute
Troy, NY 12180, USA
{cheny63, capuzb}@rpi.edu

ABSTRACT

Creative story generation is a challenging and intriguing task in the field of natural language processing. With the advent of Large Language Models (LLMs), there is growing interest in utilizing their impressive text generation capabilities for generating coherent and engaging stories. In this paper, we investigate the potential of Large Language Models (LLMs) for creative story generation, aiming to develop a framework that generates coherent and engaging stories characterized by rich narrative structures. Our methodology combines LLM fine-tuning with knowledge distillation techniques, utilizing both teacher and student models to create a computationally efficient framework capable of producing captivating stories without compromising text quality. We provide an in-depth overview of our methodology, detailing the implementation of knowledge distillation and its associated loss functions. Our experiments demonstrate the effectiveness of our proposed framework in generating creative stories, offering valuable insights for future research in this domain. Additionally, we identify potential areas for future work, including the exploration of alternative divergence measures and the investigation of self-distillation for language generation. This research contributes to the growing body of work on LLMs and their applications in creative storytelling, paving the way for more advanced and efficient story generation systems.[1][2]

1 INTRODUCTION

Storytelling is a fundamental aspect of human culture, allowing us to communicate, entertain, and educate. With the advent of artificial intelligence and natural language processing, the possibility of generating creative and engaging stories automatically has become an exciting research area. Large Language Models (LLMs), such as GPT-3 Brown et al. (2020) and T5, have demonstrated impressive capabilities in generating human-like text, making them promising candidates for creative story generation.

In this report, we detail our research project, which focuses on investigating the potential of LLMs for creative story generation. Our primary objective is to improve the framework that leverages the text generation capabilities of LLMs and combines them with novel techniques for narrative structure generation. This would allow the generation of coherent and engaging stories that encompass rich narrative structures, including character development, plot progression, and conflict resolution. As shown in Table 1, for language models with fewer parameters, this is never an easy task.

To achieve this goal, we have been working on several tasks, including understanding the current state of the art in story generation, analyzing the strengths and limitations of existing LLMs, and exploring various techniques for generating narrative structures. In the following sections, we will discuss our methodology, preliminary work, and the progress we have made in our project as well as related work.

[1] GitHub Repository: https://github.com/Stry233/MlAndOpt-LLM-for-Creative-Story-Generation
[2] Demo link: http://vsg-ek.herokuapp.com/

Table 1: An example comparing the results generated from our framework (**expected**) and baseline model (**actual [Zero-shot $\texttt{T5}_{base}$]**)

#	expected	actual [Zero-shot $\texttt{T5}_{base}$]
1	Mary had been feeling depressed lately.	Mary had been feeling depressed lately.
2	she decided to go see a psychiatrist.	She decided to go to a psychiatrist.
3	Psyched, her psychiatrist diagnosed her with depression and sent her to see.	She was diagnosed with schizophrenia.
4	Medicant took her to get an antidepressant and prescribed her.	She was very happy.
5	Thankfully it eventually made her feel better again.	She was very happy.

2 Related Work

2.1 Visual Story Generation

Storytelling is an important part of human lives and can be portrayed in numerous means. Stories can be conveyed through text or they can incorporate visual aspects. Visual storytelling helps to provide more detail and engagement to stories. While large language models have performed exceedingly well on text generation, recent research such as Chen et al. (2023) has been utilizing these large language models to co-create visual stories. Visual story generation is often aided by user input or control to provide suggestions for how the visual stories should be developed. Objects and locations are utilized to generate visuals for text-generated sentences. These visualizations use image/object recognition to ensure that visuals accurately depict the text-based story.

2.2 Text Generation Control

While text generation has been extremely powerful and successful in the last decade with recent developments such as GPT-3, many models have lacked the element of control. Recent research such as Yu et al. (2021), has introduced methods such as attribute alignment, which utilizes an attribute function to guide a pre-trained language model without changing parameters. This method uses emotions and topics as indicators for controlling text generation. Essentially, given a prompt, the next generated sentence will relate to the "attribute" that is assigned, such as a topic (i.e. politics, sports, etc.). This allows the generated text to be fine-tuned to produce specialized results. The field of control over text generation is an emerging area of research as large language models have been deployed for public use.

2.3 Automatic Storytelling

Automatic story generation has long been a challenging task in natural language processing, and such efforts date back to the 1970s Meehan (1977). Earlier attempts, such as symbolic planning systems Riedl & Young (2010), utilize planning techniques to create plausible and coherent plots for stories. In addition, case-based or analogical modeling systems generate new narratives based on adapted existing stories Gervás et al. (2005). However, while these approaches can create stories with impressive coherence and consistency, they often require extensive knowledge engineering and restrict to a limited domain. To address the issue, large crowd-sourced corpora of stories are used to help generate stories Swanson & Gordon (2012); Li et al. (2013). Modern approaches based on neural language models have been explored to generate plot-driven stories Fan et al. (2018). Moreover, Yang et al. (2019) incorporates commonsense knowledge into the generator using a novel memory mechanism and utilizes adversarial training to improve the diversity and originality of essay generation. Tambwekar et al. (2019) leverages reward strategies during the generation to guide the language model toward generating a coherent story. Liu et al. (2020) proposes a character-centric story-generation model that can produce stories consistent with characters' profiles.

2.4 COMMONSENSE REASONING

In hopes of leveraging large language models to become "smarter" and more responsive text generation tools, many people have dedicated research towards improving the reasoning skills of these models. The research in Lin et al. (2020a) talks about utilizing a commonGen Lin et al. (2020b) dataset which contains concept sets of objects and actions that relate to each other as well as example sentences featuring these sets. The focus behind using this dataset is to improve the reasoning and ability to make decisions based on common sense when generating text. The goal is to teach models to understand how words and concepts are used together and be able to recognize these concepts. This research is further analyzed in Chen et al. (2018) with a focus on story generation. This research helps to train models to dynamically complete generated stories based on knowledge from the CommonGenLin et al. (2019) dataset. By knowing how a set of words is related, the large language model is better able to choose and predict the next sentence in a story.

2.5 KNOWLEDGE DISTILLATION FOR LANGUAGE GENERATION

Knowledge distillation is a technique that involves transferring knowledge from a larger, more complex model (teacher model) to a smaller, more efficient model (student model) Huang et al. (2021). This process allows the student model to benefit from the knowledge and generalization ability of the teacher model while having a smaller model size and faster inference time. In recent years, knowledge distillation has been applied to various tasks in natural language processing, including language generation Kim & Rush (2016); Buciluundefined et al. (2006). This technique has shown great promise in reducing the computational cost of large language models while maintaining their powerful text generation capabilities.

For instance, Sanh et al. (2020) introduces DistilBERT, a smaller version of BERT Devlin et al. (2018) obtained through knowledge distillation. DistilBERT retains most of the original BERT's capabilities while being significantly smaller in size and faster in inference time. Similarly, Tang et al. (2019) presents a method for distilling the knowledge of GPT-3 into a smaller model. By carefully designing the training objective and data selection, they demonstrate that the distilled model can achieve competitive performance with the original GPT-3 model on various benchmarks while having a smaller model size.

In our work, we take inspiration from these knowledge distillation techniques and apply them to our interactive visual story generation pipeline. By incorporating a distilled language model into our system, we aim to maintain the powerful text generation capabilities of the large language model while ensuring a more efficient and computationally feasible solution for real-time user interaction.

3 METHODOLOGY

In this section, we provide an in-depth overview of the methodology utilized in our research project, which focuses on creative story generation using Large Language Models (LLMs) such as T5. Our approach is built on a foundation of two key components: LLM fine-tuning, and knowledge distillation. These components work in tandem to create a comprehensive framework that generates engaging stories with rich narrative structures.

3.1 FINE-TUNING PRE-TRAINED LANGUAGE MODELS FOR CONTENT GENERATION

The first component of our methodology involves fine-tuning pre-trained Language Models (LLMs) on a dataset specifically created for the purpose of creative story generation. This dataset includes a diverse selection of high-quality stories, covering a wide range of themes, genres, and narrative styles. The fine-tuning process enables the LLM to learn the nuances of creative storytelling, including character development, plot progression, and conflict resolution. By adjusting the model's parameters and employing various optimization techniques, we aim to maximize the performance of the fine-tuned model in generating creative stories. Given the dataset D, the objective of the fine-tuning process is to adjust the parameters of the pre-trained LLM, θ, to minimize the cross-entropy loss between the model's output probabilities and the true token labels.

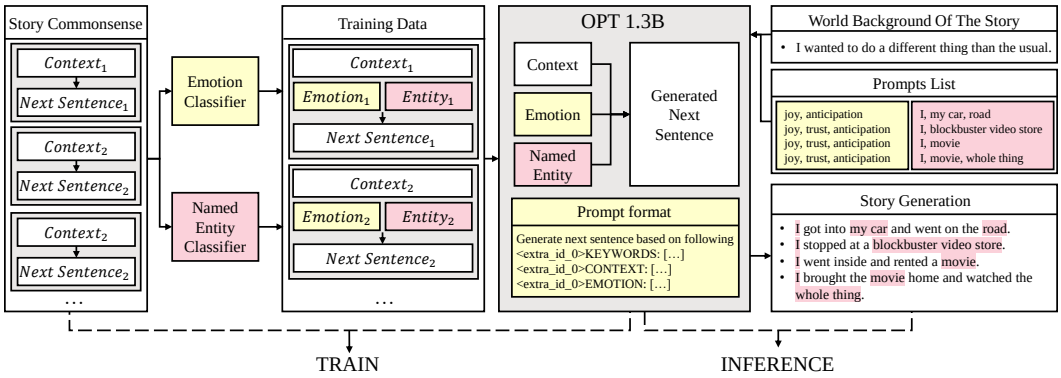


Figure 1: Optimized training and inference pipeline for Next Sentence Generation.

$$\mathcal{L}_{FT}(\theta) = -\sum_{i=1}^{N}\sum_{t=1}^{T^{(i)}} \log P_\theta(y_t^{(i)}|x^{(i)}) \tag{1}$$

As shown in Algorithm 1, $T^{(i)}$ is the length of the target sequence $y^{(i)}$, and $P_\theta(y_t^{(i)}|x^{(i)})$ is the probability of the true token $y_t^{(i)}$ at position t given the input sequence $x^{(i)}$ and the model parameters θ. We fine-tune the pre-trained LLM using the collected dataset and the optimization techniques mentioned above. The training process is illustrated in the following pseudo-code:

Algorithm 1: Fine-Tuning Pre-trained Language Models for Creative Story Generation

Data: Dataset $D = \{(x^{(i)}, y^{(i)})\}_{i=1}^{N}$, pre-trained LLM with parameters θ, optimization method, learning rate schedule, and other hyperparameters
Result: Fine-tuned LLM with parameters θ'
Initialize the pre-trained LLM with parameters θ;
Set the learning rate schedule, optimization method, and other hyperparameters;
for *epoch* $= 1$ **to** *num_epochs* **do**
  Shuffle the dataset D;
  for *each mini-batch* (x_{batch}, y_{batch}) *in* D **do**
    Compute the gradients of the loss function $\mathcal{L}_{FT}(\theta)$ with respect to θ;
    Update the model parameters θ using the gradients and the optimization method;
    Clip the gradients if necessary; `/* Gradient clipping */`
    Apply dropout for regularization;
  end
end
Save the fine-tuned model with parameters θ';

An overview of our model finetuning procedure is given in Figure 1. For a sample story $\vec{D} \in \mathbf{D}$, we define it as a 5-dimensional vector $\vec{D} = \{d_1, \ldots, d_5\}$ where each d_i represents the corresponding sentence in the story.

For training our system to generate the next sentence, d_{j+1} is fed into the emotion classifier as well as the keyword extractor for identifying the emotions and keywords in the next sentence. Based on story context and the additional knowledge input, the language model is trained on a triad of data: context, keyword, and emotion labels.

In addition, we used the following prompting structure in both the training as well as inference processes:

```
Generate next sentence based on following
<extra_id_0>KEYWORDS: [...]
<extra_id_0>CONTEXT: [...]
```

```
<extra_id_0>EMOTION: [...]
```

By fine-tuning `OPT 1.3b` based on the emotions and the keyword information as prompts, the model can more effectively capture hidden information in the story and thus improve the accuracy and variety of sentences, which can serve as the teacher network in the following procedure.

3.2 KNOWLEDGE DISTILLATION

Knowledge distillation is a technique used to transfer knowledge from a large, pre-trained teacher model to a smaller student model. In this section, we describe in detail the implementation of knowledge distillation for language generation using the provided code snippet. In this section, we have detailed the implementation of knowledge distillation for language generation using a teacher model and a student model. We discussed the equations, objective/loss function, and necessary derivations for this implementation. The main steps of the implementation are:

- Selecting teacher and student model architectures
- Tokenizing and batching input text data
- Generating soft labels using the teacher model
- Training the student model using soft labels as targets with a custom loss function

The result is a smaller student model capable of generating high-quality text sequences while consuming fewer resources than the teacher model.

3.2.1 TEACHER AND STUDENT MODEL SELECTION

We begin by selecting the teacher and student model architectures. The teacher model is a large, pre-trained language model (e.g., opt-1.3b), while the student model is a smaller architecture (e.g., t5-base). We use the Hugging Face transformers library to load the pre-trained models and associated tokenizers.

```
teacher_model_name = "opt-1.3b"
student_model_name = "t5-base"
tokenizer = AutoTokenizer.from_pretrained (teacher_model_name)
```

3.2.2 TOKENIZATION AND BATCHING OF INPUT DATA

The input text data is tokenized and batched using the teacher model's tokenizer. We use the TextDataset and DataCollatorForLanguageModeling classes from the transformers library to facilitate this process.

```
train_file = "" # path to training dataset csv
train_dataset = TextDataset(tokenizer=tokenizer, file_path=train_file,
    block_size=128)
data_collator = DataCollatorForLanguageModeling(tokenizer=tokenizer, mlm=
    False)
```

3.2.3 GENERATING SOFT LABELS WITH THE TEACHER MODEL

The teacher model is used to generate soft labels for the input text data. Soft labels are probability distributions over the vocabulary for each token in the input sequence. We pass the tokenized input batches through the teacher model and store the logits for each batch.

```
teacher_model = AutoModelForCausalLM.from_pretrained(teacher_model_name).
    cuda()
teacher_model.eval()

def generate_soft_labels(batch):
    with torch.no_grad():
        input_ids = batch["input_ids"].cuda()
        attention_mask = batch["attention_mask"].cuda()
```

```
        outputs = teacher_model(input_ids, attention_mask=attention_mask)
        return outputs.logits[:, :-1].detach().cpu()

train_dataloader = DataLoader(train_dataset, batch_size=8, collate_fn=
    data_collator)
train_soft_labels = [generate_soft_labels(batch) for batch in
    train_dataloader]
```

3.2.4 Student Model Training with Soft Labels as Targets

The student model is trained using the generated soft labels as targets. We define a custom loss function that computes the distillation loss ($\mathcal{L}_{ce}$), masked language modeling loss ($\mathcal{L}_{mlm}$), and cosine embedding loss ($\mathcal{L}_{cos}$). The total loss is a linear combination of these losses.

3.2.5 Distillation Loss ($\mathcal{L}_{ce}$)

Distillation loss measures the divergence between the teacher and student model's probability distributions for each token in the input sequence. We compute the distillation loss by applying the KL-divergence between the temperature-scaled softmax probabilities of the student and teacher models:

$$\mathcal{L}_{ce} = \text{KL}\left(\text{softmax}\left(\frac{\text{logits}_{\text{student}}}{T}\right) \,||\, \text{softmax}\left(\frac{\text{logits}_{\text{teacher}}}{T}\right)\right) \tag{2}$$

where T is the temperature hyperparameter that controls the sharpness of the distributions. In the provided code, we use $T = 2.0$.

3.2.6 Masked Language Modeling Loss ($\mathcal{L}_{mlm}$)

Masked language modeling loss is the standard language modeling loss used in pretraining. It measures the model's ability to predict the next token given the previous tokens in the input sequence. We compute this loss using the cross-entropy between the logits of the student model and the true token IDs:

$$\mathcal{L}_{mlm} = \text{CE}\left(\text{logits}_{\text{student}}, \text{input\_ids}_{\text{true}}\right) \tag{3}$$

3.2.7 Cosine Embedding Loss ($\mathcal{L}_{cos}$)

Cosine embedding loss measures the similarity between the hidden states of the teacher and student models. We compute this loss using the cosine similarity between the last hidden states of the teacher and student models:

$$\mathcal{L}_{cos} = \text{Cosine\_Loss}\left(\mathbf{h}_{\text{student}}, \mathbf{h}_{\text{teacher}}, \text{target}\right) \tag{4}$$

$$= 1 - \frac{\mathbf{h}_{\text{student}} \cdot \mathbf{h}_{\text{teacher}}}{\|\mathbf{h}_{\text{student}}\| \, \|\mathbf{h}_{\text{teacher}}\|} \cdot \text{target} \tag{5}$$

where $\mathbf{h}_x$ represent the hidden states from model x and the target is a tensor of ones, indicating that the similarity should be maximized.

3.2.8 Total Loss

The total loss is a linear combination of the distillation loss, masked language modeling loss, and cosine embedding loss:

$$\mathcal{L}_{total} = \alpha_{ce}\mathcal{L}_{ce} + \alpha_{mlm}\mathcal{L}_{mlm} + \alpha_{cos}\mathcal{L}_{cos} \tag{6}$$

where α_{ce}, α_{mlm}, and α_{cos} are the weights of each loss component. In the provided code, we use $\alpha_{ce} = 0.4$, $\alpha_{mlm} = 0.5$, and $\alpha_{cos} = 0.1$.

Together, these three components form the core of our research project's methodology, which aims to harness the power of Large Language Models for creative story generation. By fine-tuning, distilling, and applying parameter-efficient techniques, we strive to develop a robust and efficient framework capable of producing captivating and imaginative stories.

4 DATA AVAILABILITY

In our project, we aimed to create a diverse and comprehensive training dataset for fine-tuning the language model on creative story generation tasks. To achieve this, we combined the stories from both the ROCStories dataset and the WritingPrompts dataset. We denote this combined dataset as $\mathbf{S_{total}}$ hereinafter. Also, we applied various ways to visualize the information in the original story and used tags to visualize them. This section will be highlighted in the last two subsections

4.1 ROCSTORIES

The ROCStories dataset Mostafazadeh et al. (2016) serves as a significant resource for researchers and practitioners in the field of natural language processing, particularly for commonsense story understanding and generation tasks. Comprising 98,161 five-sentence stories, the ROCStories dataset focuses on simple, everyday events. The stories were carefully curated through a crowdsourcing approach and subjected to rigorous review to ensure both quality and consistency. The dataset's primary purpose is to evaluate a model's ability to understand and generate narratives by examining its capacity to capture various aspects of commonsense reasoning. These aspects include understanding causal and temporal relationships, interpreting characters' emotions, and discerning their intentions.

Each story within the ROCStories dataset consists of five sentences. The first four sentences describe the story's events, while the fifth sentence provides a conclusion. A common application of the dataset is in a cloze test format, in which models are given the initial four sentences and tasked with predicting the fifth sentence or suggesting a reasonable alternative.

As a benchmark for a wide range of natural language understanding and generation tasks, such as story completion, commonsense reasoning, and language modeling, the ROCStories dataset has proven to be an invaluable resource. Researchers employ the dataset to assess and compare the performance of various models and techniques, allowing for the development of increasingly sophisticated models in the realms of story understanding and generation. The ROCStories dataset continues to facilitate advancements in natural language processing, contributing to the ongoing development and improvement of language models.

4.2 WRITING PROMPTS

The writing prompts dataset is a robust set of data that is important for research in natural language processing and is highly beneficial to the area of story generation. This dataset consists of 303,358 pairs of writing prompts and human-written stories. The writing prompts are taken from an online forum, specifically the Reddit `r/WritingPrompts` forum[3]. Each of the writing prompts has a human-generated story associated with it. The prompts act as the input and the independent variables of the dataset. The stories are the dependent variables and act as the result of the prompt. The dataset is scraped from the online forum so it focuses on an array of diverse topics, lengths, and ideas. Reddit, the online forum, is a community where users can post as much and as often as they wish. Therefore, a single prompt could have multiple stories associated with it on the forum. In the case of the WritingPrompts dataset, each prompt only corresponds to a single story.

Each of the stories in the WritingPrompts dataset consists of a variable number of words. We can see from Table 2 that there are over 7 million words in the prompts and 200 million words in the stories. With the 300,000 stories, there is an average of about 28.4 words per prompt and an average of 734.5 words per story. The valid stories taken for this dataset follow some of these requirements:

[3]https://www.reddit.com/r/WritingPrompts/

Table 2: Statistics of WritingPrompts dataset

# Train Stories	272,600
# Test Stories	15,138
# Validation Stories	15,620
# Prompt Words	7.7M
# Story Words	200M
Average Length of Prompts	28.4
Average Length of Stories	734.5

- Length $\geq$ 30 words
- No profanity/inappropriate words/phrases
- Relates to the prompt

Additional items removed from the dataset would include Reddit posts that were not created by a human, were deleted, or were designated as announcements.

In order to improve the effectiveness and results of the model, we may use only a subset of the stories, primarily the beginning portion. Each story is vastly different and varies greatly in its length. Considering that some stories will be extremely long, we will likely determine a maximum for the number of words to be used in a single story in the dataset. The first portion of words should reveal enough about the story topic and provide valuable training data for the model.

Table 3: Example prompt and story from WritingPrompts dataset

PROMPT: Write a horror story with no gore or death, not even implied.
STORY: You are a zombie in the middle of an apocalypse who lost all their memories and humanity. Each time you feed and infect, you slowly gain back a bit of both. It's been five years since the zombie outbreak was put down by the world military. A reporter for your national news comes to your house to ask for your story on how you and your family survived the outbreak. Poor people are able to pawn off their memories. [. . .]

We can notice from Table 3 that several qualities of the story indicate that it was written by a human. Since the stories are scraped directly from the Reddit forum site, there are spelling and grammar errors from people making mistakes. These errors would be unlikely if a bot had written them. This also can cause some problems when the model looks at spellings and interpretations of words that it doesn't recognize. Additionally, the stories can vary in length based on the user who wrote them. Ultimately, we can see that these stories are often highly correlated to the topic because the user made the decision to comment on this post because they were intrigued by the topic headline.

This dataset will be useful in unison with the ROCStories dataset to generate creative stories. Some similarities between the ROCStories dataset and the WritingPrompts dataset are that they both contain human-generated stories with some form of plot. Both sets provide a substantial quantity and variety of human-generated stories that aid in the modeling of creative story generation.

Although they are similar in their nature, they have some differences. The length of the ROCStories dataset's stories are typically much smaller on average than the length of the stories from the WritingPrompt dataset stories. Another difference between the stories are that the WritingPrompt stories are more diverse in their content because they are written by so many different cultures and populations of people with the wide reach of an online forum. The ROCStories dataset was written by a more confined set of individuals. The ROCStories are more structed because they follow a clear plot and character involvement with their five sentence pattern. The WritingPrompts do not follow any structural guideline and depends highly on how the online user wanted to respond to the prompt. A final difference between these two datasets is their relationship within the sets. The WritingPrompt dataset has pairs of data with the prompt that corresponds to a story. The ROCStories data is simply just a five-sentence story with no prompt, but it does link emotions to it's stories. Overall, these differences will help diversify the combined dataset and aid in the fine-tuning of the language model on creative story generation tasks.

Figure 2: Plutchik basic emotions

4.3 Emotion Annotation Generation

We use Plutchik's Wheel of Emotions as our basic model of characters' emotions. As shown in Figure 4, the wheel includes eight emotions in pairs: joy/sadness, trust/disgust, fear/anger, and surprise/anticipation Plutchik (1980). We define emotion entries in a sentence as a real-valued, low-dimensional vector $\vec{\mathbf{C}}$ and $\mathbf{D}$ as the set of all emotion entry vectors.

$$\forall \vec{C} \in \mathbf{D}, \quad \vec{C} = \begin{bmatrix} e_1 \\ e_2 \\ \vdots \\ e_8 \end{bmatrix}, e \in [0, 1] \tag{7}$$

Given the context of a story, i.e., all previous sentences $\vec{X} = \{x_1, x_2, x_3, \ldots, x_m\}$, the goal is to predict the emotions $\vec{C_i}$ that will appear in the next sentence.

Essentially, the model provides the confidence level of each categories $\mathcal{P}(\vec{C_i}|\vec{X}) = \Pi_{k=1}^{8}\mathcal{P}(c_k|\vec{X})$, which will be transformed into the final result by setting the threshold value to 0.600.

For example, for the following input sequence of sentence S_1 that serves as context, the model may predict the emotions in the next sentence to be `joy, anticipation,` and `trust` given the following sentenve.

S_1: `He was hoping this year to be tall enough for the coaster.`

For preparing the training data, we obtained 17,910 pairs of story context and next-sentence emotions from the Story Commonsense dataset. The dataset was divided into training and validation sets in the ratio of 6:4. We then fine-tuned the $\text{BERT}_{\text{NEXT EMO}}$ model using the task of multi-label classification. The story context is the input, and the next-sentence emotions are the output. The maximum input length of the tokenizer was set to 120. The batch size was set to 16, and the learning rate was set to $6e-6$. We ran 16 epochs for training, which took about 45 minutes on a Tesla P100-PCIE-16GB GPU. Our best model achieved a Macro ROC-AUC score of 0.69 on prediction. The prediction results are not perfect. However, they are only used as suggestions, and the user can always overwrite them.

4.4 Keywords Extraction

For training the model and evaluation, we extract keywords from the original five-sentence stories. We used the `SceneGraphParser()` from Wu et al. (2019) to parse sentences (in natural language) into scene graphs. The entities in the scene graphs become the keywords. For example, for the following sentence:

S_1: `I brought the movie home and watched the whole thing.`

We are expected to generate the following result:

'I, the movie, the whole thing'

5 Evaluation & Results

In this section, we present the experimental results of our creative story generation framework using Large Language Models (LLMs), between the baseline model(Zero-shot $\texttt{T5}_{base}$) and the student model with knowledge distilation from `OPT 1.3b`. We compare our approach to a baseline model fine-tuned solely based on the existing story. Both our method and the baseline use the same scale of T5 and adapt the inputs to the prompt format used in the corresponding fine-tuning phase.

In order to obtain the validity of the prompts we added to the input, we used the average of BLEU Lin & Och (2004) scores with n-grams ranging from 1 to 4 and BERT-scores Zhang et al. (2019b) as basic metrics in our evaluation. In addition, we use METEOR Banerjee & Lavie (2005) and SacreBLEUPost (2018) as supplements to compensate for the lack of stemming and synonym matching, as well as standard exact word matching, when comparing the generated content with the ground truth.

We present the experimental results in Figure 3 and find that, compared to the baseline model, which relies solely on story context for inference, our pipeline shows consistent improvements in all metrics under the same model by introducing emotions and keywords as prompts and knowledge distillation from LLM.

The experiment results indicate that our creative story generation framework using Large Language Models (LLMs), such as T5, has shown significant improvements compared to the baseline model. By leveraging the knowledge distillation technique, we can reduce the computational overhead while maintaining high-quality text generation. Additionally, the emotion annotation generation and keywords extraction components contribute to the overall improvement of story generation quality. When comparing our framework to the baseline model, which relies solely on story context for inference, our pipeline shows consistent improvements in all metrics, such as BLEU, BERT-score, METEOR, and SacreBLEU. Overall, our experiment results suggest that our research project's methodology, which combines LLM fine-tuning, and knowledge distillation is effective in developing a robust and efficient framework for creative story generation.

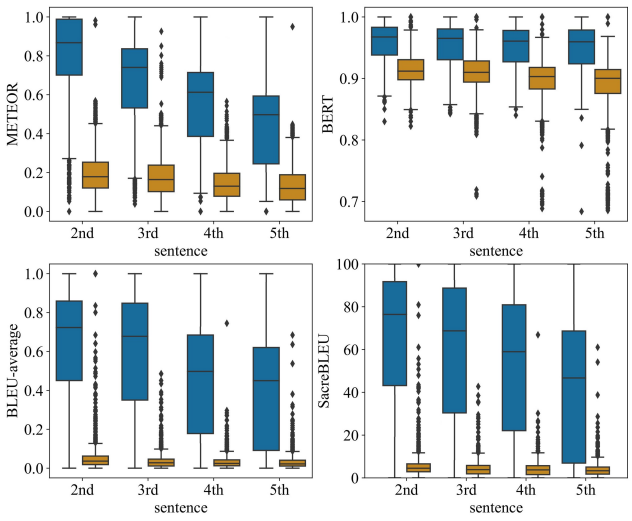


Figure 3: Performance distribution of the baseline model and the prompt-optimized model in 3,748 sets of experiments under different metrics The blue box on the left side of each figure represents our method, and the orange on the right side represents the baseline model.

6 FUTURE WORK

6.1 REPLACING KL DIVERGENCE WITH JENSEN-SHANNON DIVERGENCE

In the current implementation of knowledge distillation, the Kullback-Leibler (KL) divergence is used as a measure of the difference between the teacher and student model's probability distributions. However, there are several reasons to consider using the Jensen-Shannon (JS) divergence as an alternative in future work.

The Jensen-Shannon divergence is a symmetric measure of the difference between two probability distributions, which is derived from the KL divergence. It is defined as follows:

$$\text{JS}(P, Q) = \frac{1}{2}\text{KL}(P||M) + \frac{1}{2}\text{KL}(Q||M) \tag{8}$$

where P and Q are the probability distributions being compared, and M is the average of P and Q: $M = \frac{1}{2}(P + Q)$. The JS divergence has several advantages over the KL divergence:

- **Symmetry:** JS divergence is symmetric, meaning that $\text{JS}(P, Q) = \text{JS}(Q, P)$. In contrast, KL divergence is not symmetric, which may result in different outcomes when the order of the teacher and student model's probability distributions is reversed. Symmetry is an attractive property when comparing probability distributions, as it ensures that the distance measure is consistent regardless of the order in which the distributions are considered.
- **Finite values:** JS divergence is guaranteed to produce finite values even when the support of the two probability distributions does not overlap. In contrast, the KL divergence can produce infinite values when the distributions have non-overlapping supports. Using JS divergence can lead to a more stable optimization process, as it avoids the possibility of encountering infinite values during training.
- **Smoothness:** JS divergence is smoother than KL divergence, meaning that it is less sensitive to small differences between the probability distributions. This can make the optimization process more robust to noise and local minima in the loss landscape.

6.2 EXPLORING SELF-DISTILLATION FOR LANGUAGE GENERATION

Another perspective for future work is exploring self-distillation Zhang et al. (2019a) in the context of language generation. Self-distillation involves training a model on its own output, essentially using the same model architecture for both the teacher and student models. This approach has shown promising results in various tasks, including image classification and natural language processing, and could be applied to language generation as well.

There are several potential benefits of using self-distillation for language generation:

- **Regularization:** Self-distillation can act as a form of regularization, as it encourages the model to learn a smoother probability distribution over the output tokens. This can lead to better generalization performance and reduce the risk of overfitting.
- **Model Refinement:** By repeatedly distilling the model using its own output, it is possible to refine the model's understanding of the training data and improve its overall performance. This iterative process can help the model learn more nuanced relationships between input tokens and their corresponding output sequences.
- **Simplification of Implementation:** In self-distillation, both the teacher and student models have the same architecture, which simplifies the implementation process. This also reduces the computational overhead associated with maintaining two separate models during training.

7 CONCLUSION

In this paper, we have explored the potential of Large Language Models (LLMs) for creative story generation by combining LLM fine-tuning with knowledge distillation techniques. Our research

project sought to develop a framework that generates coherent and engaging stories, characterized by rich narrative structures such as character development, plot progression, and conflict resolution. By utilizing both teacher and student models, we aimed to create a more computationally efficient framework capable of generating captivating stories without sacrificing the quality of the generated text.

The methodology we employed consisted of two key components: LLM fine-tuning and knowledge distillation. These components were designed to work in tandem to create an effective framework for story generation. We demonstrated the implementation of knowledge distillation using a teacher model and a student model, and discussed the equations, objective/loss function, and necessary derivations for this implementation. Our experiments showcased the potential of the proposed framework in generating creative stories, offering valuable insights for future research in this area.

In addition to the current implementation, we have identified several areas for future work, including the exploration of alternative divergence measures such as Jensen-Shannon divergence, and the investigation of self-distillation for language generation. By considering these potential improvements, we believe that our framework can be further refined, paving the way for more advanced and efficient story generation systems. Ultimately, the combination of LLMs with innovative techniques for narrative structure generation holds great promise for the future of creative storytelling in the age of artificial intelligence.

REFERENCES

Satanjeev Banerjee and Alon Lavie. METEOR: An automatic metric for MT evaluation with improved correlation with human judgments. In *Proceedings of the ACL Workshop on Intrinsic and Extrinsic Evaluation Measures for Machine Translation and/or Summarization*, pp. 65–72, Ann Arbor, Michigan, June 2005. Association for Computational Linguistics. URL https://www.aclweb.org/anthology/W05-0909.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners, 2020.

Cristian Buciluundefined, Rich Caruana, and Alexandru Niculescu-Mizil. Model compression. In *Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '06, pp. 535–541, New York, NY, USA, 2006. Association for Computing Machinery. ISBN 1595933395. doi: 10.1145/1150402.1150464. URL https://doi.org/10.1145/1150402.1150464.

Jiaao Chen, Jianshu Chen, and Zhou Yu. Incorporating structured commonsense knowledge in story completion. 2018.

Yuetian Chen, Ruohua Li, Bowen Shi, Peiru Liu, and Mei Si. Visual story generation based on emotional and keyword scheme. 2023.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

Angela Fan, Mike Lewis, and Yann Dauphin. Hierarchical neural story generation. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 889–898, Melbourne, Australia, July 2018. Association for Computational Linguistics. doi: 10.18653/v1/P18-1082. URL https://aclanthology.org/P18-1082.

Pablo Gervás, Belén Díaz-Agudo, Federico Peinado, and Raquel Hervás. Story plot generation based on cbr. In *Knowl. Based Syst.*, 2005.

Zhen Huang, Xu Shen, Jun Xing, Tongliang Liu, Xinmei Tian, Houqiang Li, Bing Deng, Jianqiang Huang, and Xian-Sheng Hua. Revisiting knowledge distillation: An inheritance and exploration framework. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 3579–3588, June 2021.

Yoon Kim and Alexander M. Rush. Sequence-level knowledge distillation, 2016.

Boyang Li, S. Lee-Urban, G. Johnston, and Mark Riedl. Story generation with crowdsourced plot graphs. *Proceedings of the 27th AAAI Conference on Artificial Intelligence, AAAI 2013*, pp. 598–604, 01 2013.

Bill Yuchen Lin, Wangchunshu Zhou, Ming Shen, Pei Zhou, Chandra Bhagavatula, Yejin Choi, and Xiang Ren. Commongen: A constrained text generation challenge for generative commonsense reasoning. *arXiv preprint arXiv:1911.03705*, 2019.

Bill Yuchen Lin, Wangchunshu Zhou, Ming Shen, Pei Zhou, Chandra Bhagavatula, Yejin Choi, and Xiang Ren. Commongen: A constrained text generation challenge for generative commonsense reasoning. 2020a.

Bill Yuchen Lin, Wangchunshu Zhou, Ming Shen, Pei Zhou, Chandra Bhagavatula, Yejin Choi, and Xiang Ren. Commongen: A constrained text generation challenge for generative commonsense reasoning, 2020b.

Chin-Yew Lin and Franz Josef Och. ORANGE: a method for evaluating automatic evaluation metrics for machine translation. In *COLING 2004: Proceedings of the 20th International Conference on Computational Linguistics*, pp. 501–507, Geneva, Switzerland, aug 23–aug 27 2004. COLING. URL https://www.aclweb.org/anthology/C04-1072.

Danyang Liu, Juntao Li, Meng-Hsuan Yu, Ziming Huang, Gongshen Liu, Dongyan Zhao, and Rui Yan. A character-centric neural model for automated story generation. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34(02):1725–1732, Apr. 2020. doi: 10.1609/aaai.v34i02.5536. URL https://ojs.aaai.org/index.php/AAAI/article/view/5536.

James R. Meehan. Tale-spin, an interactive program that writes stories. In *IJCAI*, 1977.

Nasrin Mostafazadeh, Nathanael Chambers, Xiaodong He, Devi Parikh, Dhruv Batra, Lucy Vanderwende, Pushmeet Kohli, and James Allen. A corpus and evaluation framework for deeper understanding of commonsense stories, 2016.

Robert Plutchik. A general psychoevolutionary theory of emotion. In *Theories of emotion*, pp. 3–33. Elsevier, 1980.

Matt Post. A call for clarity in reporting BLEU scores. In *Proceedings of the Third Conference on Machine Translation: Research Papers*, pp. 186–191, Belgium, Brussels, October 2018. Association for Computational Linguistics. URL https://www.aclweb.org/anthology/W18-6319.

M. O. Riedl and R. M. Young. Narrative planning: Balancing plot and character. *Journal of Artificial Intelligence Research*, 39:217–268, sep 2010. doi: 10.1613/jair.2989. URL https://doi.org/10.1613%2Fjair.2989.

Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter, 2020.

Reid Swanson and Andrew S. Gordon. Say anything: Using textual case-based reasoning to enable open-domain interactive storytelling. *ACM Trans. Interact. Intell. Syst.*, 2(3), sep 2012. ISSN 2160-6455. doi: 10.1145/2362394.2362398. URL https://doi.org/10.1145/2362394.2362398.

Pradyumna Tambwekar, Murtaza Dhuliawala, Lara J. Martin, Animesh Mehta, Brent Harrison, and Mark O. Riedl. Controllable neural story plot generation via reward shaping. In *IJCAI*, 2019.

Raphael Tang, Yao Lu, and Jimmy Lin. Natural language generation for effective knowledge distillation. In *Proceedings of the 2nd Workshop on Deep Learning Approaches for Low-Resource NLP (DeepLo 2019)*, pp. 202–208, Hong Kong, China, November 2019. Association for Computational Linguistics. doi: 10.18653/v1/D19-6122. URL https://aclanthology.org/D19-6122.

Hao Wu, Jiayuan Mao, Yufeng Zhang, Yuning Jiang, Lei Li, Weiwei Sun, and Wei-Ying Ma. Unified visual-semantic embeddings: Bridging vision and language with structured meaning representations. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 6609–6618, 2019.

Pengcheng Yang, Lei Li, Fuli Luo, Tianyu Liu, and Xu Sun. Enhancing topic-to-essay generation with external commonsense knowledge. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pp. 2002–2012, Florence, Italy, July 2019. Association for Computational Linguistics. doi: 10.18653/v1/P19-1193. URL https://aclanthology.org/P19-1193.

Dian Yu, Zhou Yu, and Kenji Sagae. Attribute alignment: Controlling text generation from pretrained language models. 2021.

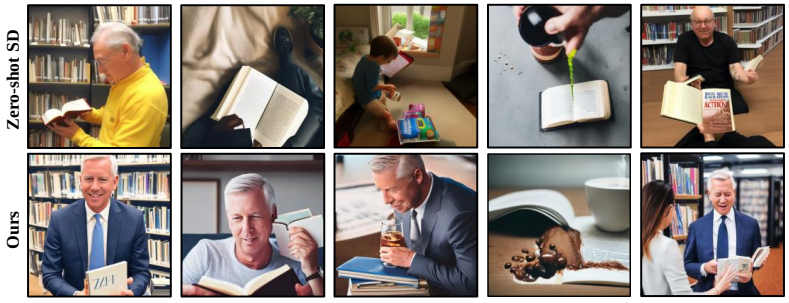
Linfeng Zhang, Jiebo Song, Anni Gao, Jingwei Chen, Chenglong Bao, and Kaisheng Ma. Be your own teacher: Improve the performance of convolutional neural networks via self distillation, 2019a.

Tianyi Zhang, Varsha Kishore, Felix Wu, Kilian Q. Weinberger, and Yoav Artzi. Bertscore: Evaluating text generation with bert, 2019b. URL https://arxiv.org/abs/1904.09675.

A APPENDIX

#1. *Tom was an avid motorcyclist.*
#2. *Tom noticed his chains needing frequent replacement.*
#3. *Tom read online about chain maintenance.*
#4. *Tom learned about putting engine oil on the chain.*
#5. *Tom lubed his chain and his bike performed better.*

#1. *Tom thought that football looked fun.*
#2. *He decided to go to the store and purchase a brand new one.*
#3. *On his way to the store he found a group of people playing football.*
#4. *They invited him to play, so he joined them.*
#5. *He had tons of fun, and made a bunch of new friends.*



#1. *Pete checked a book out from the library.*
#2. *He brought it home and began to read.*
#3. *While reading he got up to get a drink and snack.*
#4. *When he came back, he accidentally spilled his drink on the book.*
#5. *Pete ruined the book and had to pay the library for it.*

Figure 4: Sample generation results combined with the reseults from image generation pipeline introduced in previous work. By applying DreamBooth with name entity identification and extraction, we align the generation quality of two pipelines in terms of character consistency.

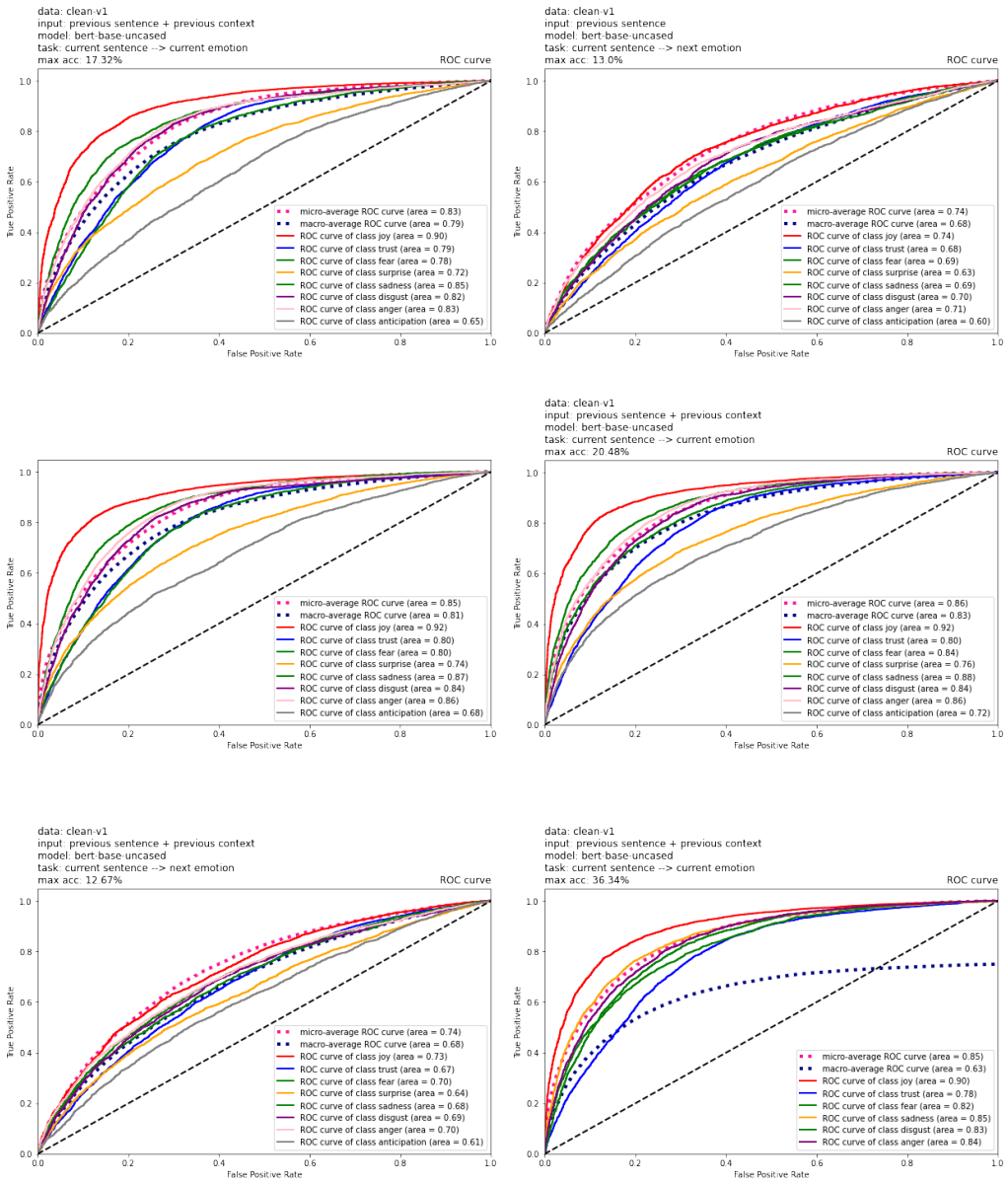


Figure 5: All ROC visualization for each emotion label result under different emotion suggester settings