Hi everyone. This is Yuetian. Today we’d like to share with you some ideas presented in our work: “Leveraging Large Language Models for Creative Story Generation”. This project is also conducted by Yiyang at the early stage who are not with us today. And is also under the guidance and feedback from Prof. Gittens.

In this work, we designed an end-to-end framework that encourage interactive narrative experience based on keyword control by using a language model in a cheaper way and it also comes with automatically generated images for every stage of stories to help users to have a better understanding of story progress. As you can see from the example presented on our right, we used an example from ROCStories dataset, which we will describe in detail later. We can see that the whole story is about Ethan making potatoes. By adjusting the sentiment, our framework enables real-time plot tree expansion and effectively changes the story direction.

, [work through the picture...]

This is also part of the keywords we offer for modification.

And for our framework. The first thing that needs to be made clear is that unlike previous approaches like Rytr or Jasper, that to implement an interactive story generation by modifying the storyline itself, we intend to introduce more control to this process that we can make use of both priori knowledge of language models and user’s creativity that brings immersion and manual control to the plots. By giving the language model the prompt that includes key information, which we will discuss in detail later, in the next stage of the story. This allows the story to be logical while still accepting the evaluation from the users. As we progress in our framework, the next essential element is the implementation of knowledge distillation to improve the model's performance. Knowledge distillation allows us to extract the knowledge from a large, pre-trained teacher model and transfer it to a smaller student model. This results in a more efficient model with reduced computational resources, making it ideal for our interactive story generation system.

As for the controlled image generation implementation, we produced a pipeline based on prompt learning to generate images in real time based on the content provided in the above process to further enhance the flexibility of story generation as well as to further couple the text-to-image process into our framework. In this presentation, we will not be focusing on the specific implementation of image pipeline but using my previous work and make use of it to create the story in a more immersive way. In the next slides, we will elaborate further on the challenging part of the implementation of the previous two points.

And now, let's go back to the first point, where we will further start discussing the details of our implementation of the text pipeline and how to use it. Here, as we mentioned before, as a replacement for the storyline modification, we use the existing background sentences as a priori knowledge by which we can predict two pieces of information as our key elements. One is the key named entity, which we will shorten as the keyword in following, in the next sentence while the other is the sentiment label. In addition to letting the user do the input manually, we have implemented both automatic suggester that predict both information based on the BERT-like as well as 3rd party API based on language model like sng\_parser from spacy.

To finetuning these language model, we mainly used story content based on ROCStories as our dataset, where each group of stories consists of five sentences and during the crowdsourcing process, narrative creator was asked to make sure that stories are making progress while keep its coherent and consistency. Specifically, some sample stories are placed on your right-hand side, where we can see that…

The story alone is not enough, we also need to visualize some additional information in the story, including emotions and characters serves as the keyword. So, we also used Story Commonsense as an additional dataset. The stories section is a subset of ROCStories and contains a sample of 15,000 stories, which is quite a bit less than the nearly 100,000 samples in ROCStories. but on top of it, it contains additional information for each story, including sentiment, motivation, character quest, etc. Here we use categorical sentiment annotation, which uses Plutchik’s wheel of emotion model that using the following 8 emotions for each stage of stories to express the emotions of the main character in the story, including …

As for the implementation of the suggester, on the sentiment predictor, we predicted the 8-class sentiment as a one-hot vector, with each entry representing one sentiment. That is, the goal of the model is reduced to a multi-label classification, that predicting the confidence level of each entry in an array of length 8. And at the end, we achieved 78.3% of the macro ROC-AUC in the prediction. Meaning that it can make valid sentiment predictions and provide quite reasonable suggestions.

As for named entities, we use sng\_parser to generate named entities that have appeared before for users to choose from. In addition to this, we used the full ROCStories dataset and sng\_parser again for the extraction of named entities for the next sentence. By combining both results, this gives us the key information, but it's not enough for story generation, so let's go back to the previous page and further explain how to use this information next

Before we delve deeper into the next stage, let us reiterate our aim to design a framework that allows the user to control the story generation process. As such, the model provides suggestions based on statistically consistent cognitive patterns, but its output is entirely modifiable by the user before proceeding to the next step. This user-modified information is then incorporated into the appropriate prompt template and carried forward.

In the final step, we run the text and image generation as two parallel pipelines. The image pipeline provides visual references for users, while the text pipeline iteratively returns to the first step to serve as context for further content generation. To ensure that the model understands the content and value each set of information brings, it is crucial to introduce our training strategy, including the implementation of knowledge distillation, and the dataset used.

Knowledge distillation, a technique that allows us to transfer knowledge from a large, pre-trained teacher model to a smaller student model, plays a vital role in our framework. This process results in an efficient model that retains high-quality text generation capabilities while consuming fewer computational resources. In this case, we choose a large model named opt-1.3b as the teacher, and a smaller model like t5-base as the student. The Hugging Face transformers library facilitates the process by providing pre-trained models and tokenizers.

By incorporating knowledge distillation into our training strategy, we create a more powerful and resource-efficient language model, thus enhancing the overall performance of our interactive story generation system. With this approach, our framework effectively combines the prior knowledge of language models and the user's creativity, fostering a more immersive and controlled storytelling experience.

The knowledge distillation technique comprises several steps, such as selecting the teacher and student model architectures, computing distillation loss, masked language modeling loss, and determining the total loss as a linear combination of the various loss components. And after we get the generation results, the new sentence will iteratively go back to the first step and serve as the context for further content generation. in order to make the model understand the content and value that each set of information brings, I think we need to start introducing our training strategy and the dataset used.

The teacher network is a crucial component in the knowledge distillation process, as it serves as the source of knowledge for the student model. The teacher network is typically a large, pre-trained language model with an extensive capacity for understanding and generating text. This model has been trained on massive datasets, enabling it to capture intricate patterns and relationships within the text data, so it can serve as the source of knowledge for our student network, which is a smaller, more resource-efficient model designed to generate interactive stories that we will discuss it later.

The teacher model, such as opt-1.3b, boasts an expansive architecture with numerous layers, hidden units, and attention heads. This enables the model to learn a wide range of linguistic features and relationships, allowing it to generate high-quality text. The model's extensive pre-training on vast amounts of text data endows it with a comprehensive understanding of syntax, semantics, and context. Consequently, the teacher network is capable of providing valuable knowledge and guidance to the student network during the distillation process.

Furthermore, the teacher network's configuration incorporates various techniques to optimize its performance. For instance, OPT employs layer normalization, which stabilizes the training process by normalizing the outputs of each layer. It also uses a different positional encoding for processing huge-volume input text to embed information about the position of tokens in a sequence. These and other architectural enhancements contribute to the teacher network's ability to generate coherent, contextually appropriate text.

As part of the knowledge distillation process, the teacher network generates probability distributions over tokens for a given input sequence. The student network learns from these distributions, aiming to mimic the teacher's performance while maintaining a smaller, more efficient architecture. The KL-divergence between the temperature-scaled softmax probabilities of the teacher and student models serves as the distillation loss, which we minimize to facilitate knowledge transfer.

As the following, the student model we used is a T5 with finetuned CommenGen published by Manu Romero in huggingface community. By finetuning this dataset, we can gain sentences based on keyword entered. Specifically, when the input value is

The reason of selection of this model is that we want the model to be able to use the experience gained in this part of the task when performing controlled prediction tasks, that we discussed in following, so that the named entities in left-side can be effectively inserted into the next sentence generated on the right

And for our formal task that brings keywords and prompt template to task, we are now able to set context, named entity and sentiment into the template and serves as the input to get the sentence we want.

The knowledge distillation process aims to compress the knowledge acquired by the teacher network into a smaller, more efficient student model. To achieve this, we define a loss function that measures the divergence between the teacher and student model's probability distributions for each token in the input sequence. This loss function, known as the distillation loss, quantifies the differences between their probability distributions for each token in the input sequence. By minimizing this loss, we ensure that the student model closely resembles the teacher model in terms of generated text quality.

To compute the distillation loss, we use the KL-divergence between the temperature-scaled softmax probabilities of both the teacher and student models. The temperature hyperparameter, $T$, plays a vital role in controlling the sharpness of the distributions. A higher value of $T$ results in a softer distribution, making it easier for the student model to learn from the teacher model. In our experiments, we found that a temperature value of 2.0 yielded optimal results.

As the student model learns from the teacher model, we also evaluate its performance using masked language modeling loss. This loss measures the model's ability to predict the next token in the input sequence, given the previous tokens. The cross-entropy between the logits of the student model and the true token IDs is used to calculate this loss.

The total loss, which combines the distillation loss, masked language modeling loss, and cosine embedding loss, serves as the optimization objective. By carefully tuning the weights of each component, we strike a balance that leads to the desired transfer of knowledge.

To assess the validity of the prompts added to the input, we employ several evaluation metrics such as BLEU, BERT-score, METEOR, and SacreBLEU. These metrics help account for various aspects of the generated content, including exact word matching, stemming, and synonym matching when comparing the generated content with the ground truth.

Our experimental results indicate that our creative story generation framework demonstrates consistent improvements in all evaluation metrics compared to the baseline model, which relies solely on story context for inference. By incorporating emotions and keywords as prompts and utilizing knowledge distillation from the Large Language Model, our pipeline enhances the overall quality of story generation.

In our discussion of future work, we will explore two primary avenues for improving the current implementation of knowledge distillation. First, we will consider replacing the Kullback-Leibler (KL) divergence with the Jensen-Shannon (JS) divergence. Second, we will explore the concept of self-distillation in the context of language generation.

Currently, KL divergence is used as a measure of difference between the teacher and student model's probability distributions. However, there are several reasons to consider using the Jensen-Shannon divergence as an alternative. JS divergence is a symmetric measure derived from KL divergence, which offers three key advantages:

Symmetry: JS divergence is symmetric, ensuring that the distance measure is consistent regardless of the order in which the probability distributions are considered. This property is valuable when comparing probability distributions, as it prevents potential discrepancies that could arise due to the non-symmetric nature of KL divergence.

Finite values: Unlike KL divergence, JS divergence is guaranteed to produce finite values even when the support of the two probability distributions does not overlap. This 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, making the optimization process more robust to noise and local minima in the loss landscape.

Another promising direction for future work is exploring self-distillation in the context of language generation. Self-distillation involves training a model on its own output, 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, encouraging the model to learn a smoother probability distribution over the output tokens. This can lead to better generalization performance and reduced 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.

Therefore, by exploring the use of Jensen-Shannon divergence and self-distillation in the context of language generation offers promising avenues for future work. These approaches have the potential to improve the stability, performance, and simplicity of knowledge distillation methods in language generation tasks.

 And here is our demo time!

|  |  |  |  |
| --- | --- | --- | --- |
| **0** | **[Emotion]** | **[Keyword]** | **Oswald decided to write a novel.**  **[generate image using this]** |
| 1 | joy, trust, anticipation | Plan A: He, book, month  Plan B: He, book | A: He worked on his book for several months.  B: he worked on the book all summer |
| 2 | joy, anticipation | Oswald, the book, publisher | Oswald took the book to a publisher. |
| 3 | Plan A:  fear, sadness, disgust  Plan B:  joy, trust, surprise | publisher, book | A: The publisher rejected his book almost immediately.  B: a publisher agreed publish the book. |
| 4 | [Follow B] | book, Oswald | the book was discovered and Oswald was happy. |