Deep Learning-Based Language Modelling for Automatic Story Generation

Geetika Bandlamudi(gb2642) [*], Ramanarayanan Sankaranarayanan (rs8117) [*], Vinayak Muttathu Santhoshkumar (vm2368)[*]

[1]New York University
GitHub: Link

Abstract

Automatic story generation (ASG) involves the ability to generate coherent and engaging stories and has many potential applications, from creative writing to chatbots and virtual assistants. In recent years, It has gained a lot of popularity and is an active research area. The best perplexity achieved by our architecture is 42.254% and the best BLEU score is 0.59. We have discussed about the different models we tried along with the details about which approach works and why.

Introduction

Story generation is an intriguing field within natural language processing (NLP) that involves the automatic generation of coherent and engaging narratives. It has gained significant attention in recent years due to its potential applications in various domains, such as interactive storytelling, virtual assistants, and content generation. By leveraging machine learning techniques, story generation models can produce original stories based on given prompts, enabling creative and dynamic storytelling.

In this report, we explore the use case of story generation using the WritingPrompts dataset, released by Facebook Research. The dataset consists of over 300,000 writing prompts, each designed to inspire storytelling. It includes corresponding human-authored stories that serve as examples or responses to the prompts. Our objective is to train and evaluate story generation models using a part of this dataset, aiming to generate coherent and imaginative narratives that closely align with human-authored stories.

To accomplish this, we employ a TD-VAE model which is a powerful framework that combines the principles of variational autoencoders (VAEs) and temporal modeling. It enables us to capture the temporal dynamics and dependencies within a story, allowing for the generation of compelling and coherent narratives. By leveraging the TD-VAE model, we aim to generate stories that not only adhere to the given story prompts but also exhibit a realistic and natural flow of events. The evaluation of the generated stories is a crucial aspect of our study. By delving into story generation using the WritingPrompts dataset, VAE models, fine tuned pre-trained models we aim to contribute to the advancement of automated storytelling techniques and explore the potential of generating compelling narratives in a data-driven and creative manner. Through our investigation, we hope to gain insights into the capabilities and limitations of the VAE approach for story generation and pave the way for future research in this exciting field.

Literature survey

Automatic Story Generation has been a topic of interest for a very long time. Deep learning methods are gradually being adopted to solve problems in traditional story generation, making story generation a new research hotspot in the field of text generation. However, in the field of story generation, the widely used seq2seq model is unable to provide adequate long-distance text modeling. Researchers were motivated to use Seq2Seq models to generate complete stories due to its success in different NLP tasks. Jain et al. (Jain et al. 2017) combined two off-the-shelf systems to construct a story generator that generates stories when given a sequence of independent short descriptions. As an advancement, a lot of research has been carried out with Recurrent Neural Networks in the domain of story generation. According to previous research, while Recurrent Neural Networks (RNNs) have been successful in various Sequence-to-Sequence (Seq2Seq) tasks, they have not met expectations in the domain of story generation. In many systems, RNNs struggle to generate coherent stories beyond a few sentences. This limitation arises from the fact that a story is a sequence of consistent events that often spans a longer context than an RNN can effectively maintain. Khandelwal et al. (Khandelwal et al. 2018) demonstrated that the predictions of RNNs rely heavily on a relatively small portion of the preceding tokens, which becomes problematic as the story generation progresses. Consequently, RNNs tend to lose the connection between the currently generated event and the distantly preceding events, leading to inconsistencies and a lack of overall coherence in the generated story. In a thesis by Araz (Araz 2020), a transformer neural network was proposed for generating stories based on given prompts. The generated stories were found to be innovative and feasible. However, the model exhibited certain drawbacks, including repetitions and grammatical errors in the generated text. Additionally, it was observed that the model did not adequately prioritize the provided prompts during the story-generation process. Large-scale pre-trained

[*]These authors contributed equally.

language models have shown high abilities in processing natural languages. Samples of text generated by the GPT-2 model (Radford et al. 2019) show that these models can generate text comparable to humans' writings. This encouraged researchers to use pre-trained models in story generation. (Kaggle 2020) explored some aspects of fine-tuning for a specific dataset and fine-tuning the GPT-2 model. We got motivation from this approach and dived deep into multiple hyperparameter tunings for the best results.

Another paper (Yu et al. 2017) on SeqGAN explores the use of Variational Autoencoder (VAE) models for automatic story generation. They propose SeqGAN, a novel framework that combines sequence generative adversarial networks with policy gradient methods. While SeqGAN primarily focuses on reinforcement learning, it leverages the VAE model as the generator component. This work demonstrates the potential of VAE-based approaches in generating diverse and coherent stories, providing insights into the application of VAEs in the context of automatic story generation. This aspect interested us a lot and hence, we explored more variations of Autoencoders and their applications. One such paper (Gregor et al. 2018) by Deepmind proposed a framework that combines the benefits of Variational Autoencoders (VAEs) with Temporal Difference (TD) learning. The architecture is shown in Figure 1. This was particularly interesting because TD-VAE (Temporal Dependency VAE) incorporates a temporal difference loss term to encourage the latent space to capture temporal dependencies in sequential data. This TD-VAE aims to learn a latent space that is consistent across adjacent time steps, leading to improved modeling of dynamics and smoother transitions in generated sequences. Upon our study, we noticed that coherency and continuation are one of the most important aspects of a good story generation. Since, stories inherently possess a sequential structure, where events and actions unfold over time, TD-VAE's incorporation of temporal difference learning allows it to capture the temporal dependencies and dynamics present in sequential data. This enables the model to generate coherent and contextually appropriate sequences of events, making it suitable for generating story-like narratives. This is one of the active research areas and we found that one of the recent studies at the University of Edinburgh in 2021 (Wilmot and Keller 2021) has proven good results with the TD-VAE model for story generation. The paper has shown that the TD-VAE model can outperform alternative LSTM and transformer models on automatic coherence evaluation.

Dataset

For this task, we have used the WritingPrompts dataset (Facebook Research 2020) which was released by Facebook Research. This contains over 300,000 short stories. Each writing prompt provides a brief scenario, situation, or idea intended to inspire storytelling. The prompts cover a wide range of themes, genres, and narrative settings, allowing for diverse story generation possibilities. The dataset also includes corresponding human-authored short stories that serve as examples or responses to the prompts. Owing to the limited computing resources that we had for this task, we used a part of the dataset only. Researchers can compare the generated stories against the human-written ones to assess the quality, coherence, and creativity of the generated narratives. The reason why this dataset was picked out of others is because of the availability of human-authored stories associated with the prompts which enables a robust evaluation of story generation models. Our approach is to use this dataset and compare our model's working in comparison with other state-of-the-art models. The WritingPrompts dataset has gained recognition and popularity within the research community, making it a common benchmark for story generation models. We took the first 10,000 records for training and the first 2,500 records from the validation data and 2,500 records from the test set. This subset provides a representative portion of the dataset while accommodating the resource constraints. While the selected subset may not capture the entirety of the dataset, it provides a sufficient basis for evaluating our model's effectiveness in generating coherent and engaging stories.

Methodology

1. Preprocessing

The preprocessing stage plays a crucial role in preparing the data for subsequent modeling tasks and ensuring its suitability for the specific requirements of our approach. The preprocessing steps included:

- Tokenization: We tokenize the text data, breaking it down into individual tokens such as words or subwords. This allows us to represent the text as a sequence of discrete units, facilitating further processing and analysis. For our implementation, we used a BERT Tokenizer. BERT (Bidirectional Encoder Representations from Transformers) generates token representations that consider the context of the surrounding words. This contextual information helps capture nuances in the meaning of the tokens, which is particularly useful in story generation where the sequence of words plays a crucial role in conveying the narrative. BERT also introduces special tokens, such as the [CLS] token for classification and the [SEP] token for separating segments of text. These tokens provide additional information and aid in specific tasks like story generation, where the model needs to understand the structure and boundaries of the generated text. While working with GPT, we used a GPTTokenizer. This tokenizer converts the text into a sequence of tokens, including special tokens like <sep> for separating prompts and stories, and <eos> for end of story.
- Padding: Story sequences in the dataset may have different lengths due to variations in the number of tokens present in each sequence. However, to process the data in batches, it is necessary to have consistent sequence lengths within each batch. Padding resolves this issue by adding a special padding token to the shorter sequences, extending them to match the length of the longest sequence in the batch. It ensures that the model can pro-

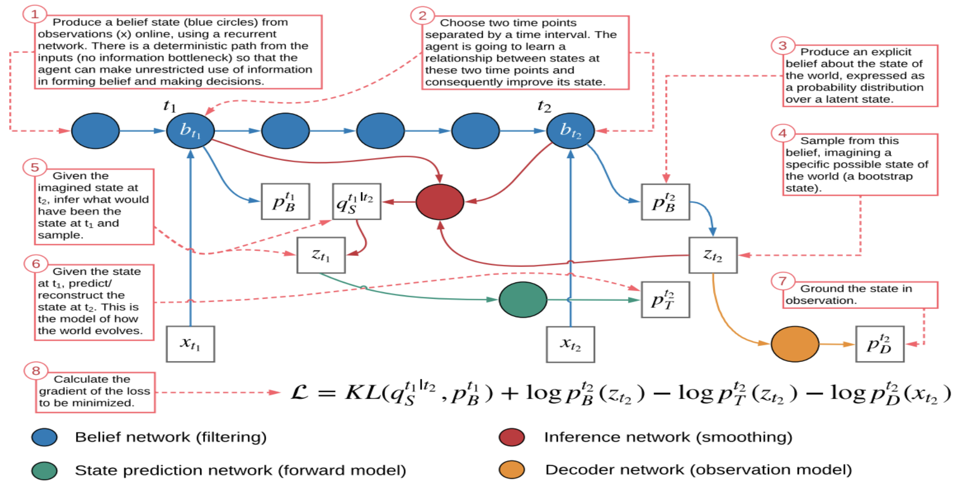

1 Produce a belief state (blue circles) from observations (x) online, using a recurrent network. There is a deterministic path from the inputs (no information bottleneck) so that the agent can make unrestricted use of information in forming belief and making decisions.
2 Choose two time points separated by a time interval. The agent is going to learn a relationship between states at these two time points and consequently improve its state.
3 Produce an explicit belief about the state of the world, expressed as a probability distribution over a latent state.
4 Sample from this belief, imagining a specific possible state of the world (a bootstrap state).
5 Given the imagined state at t_2, infer what would have been the state at t_1 and sample.
6 Given the state at t_1, predict/ reconstruct the state at t_2. This is the model of how the world evolves.
7 Ground the state in observation.
8 Calculate the gradient of the loss to be minimized.
t_1
t_2
b_{t_1}
b_{t_2}
$p_B^{t_1}$
$q_S^{t_1|t_2}$
$p_B^{t_2}$
z_{t_1}
z_{t_2}
$p_T^{t_2}$
$p_D^{t_2}$
x_{t_1}
x_{t_2}
$\mathcal{L} = KL(q_S^{t_1|t_2}, p_B^{t_1}) + \log p_B^{t_2}(z_{t_2}) - \log p_T^{t_2}(z_{t_2}) - \log p_D^{t_2}(x_{t_2})$
Belief network (filtering)
Inference network (smoothing)
State prediction network (forward model)
Decoder network (observation model)

Figure 1: Originally published TD-VAE Architecture from (Gregor et al. 2018)

cess multiple sequences simultaneously and perform operations efficiently across the batch dimension.

- Test-preprocessing: We removed the punctuation and white space characters. We made the choice not to do lemmatization but to do stop word removal for this problem in hand. We wanted to preserve the original form of the text and retain as much contextual information as possible. Storytelling often relies on the specific choice of words and the flow of sentences, so lemmatizing words could potentially alter the narrative structure and tone. Therefore, we decided to prioritize the preservation of the original text while still applying basic preprocessing steps to ensure consistency and readability. The labels for training are created by copying the input IDs and applying padding to match the attention mask.
- Train-Validation-Test Split: We split the preprocessed dataset into training, validation, and test sets. The training set is used to train the story generation models, the validation set is employed for hyperparameter tuning and model selection, and the test set is used for final evaluation and performance assessment. The split that we chose was 10,000 prompts for training, 2500 prompts for validation, and 2500 prompts for testing.

2. Model Architecture

2.1 Seq2Seq model with VAE-like architecture

Our initial architecture consisted of a standard sequence-to-sequence (seq2seq) model with a VAE-like architecture. We included an encoder, decoder and re-parameterize module. In the encoder, the source input sequences are embedded using an embedding layer. The embedded sequences are then passed through an LSTM layer to capture sequential information and extract the hidden representations. The final hidden state is fed into linear layers to predict the mean (mu) and log variance (logvar) of a latent variable called 'z'. The reparameterization trick is employed to sample 'z' from a Gaussian distribution defined by the predicted mean and log variance.

The decoder takes embedded target sequences and the sampled latent variable 'z' as input. Similar to the encoder, the target sequences are embedded using an embedding layer. The latent variable 'z' is expanded and concatenated with the embedded target sequences along the time axis. This concatenated input is then passed through an LSTM decoder to generate a sequence of hidden states. Finally, a linear layer maps the decoder's hidden states to the target vocabulary size, producing logits for each token in the output sequence. We then trained this network with an Adam optimizer and a CrossEntropy loss function. Upon training this network for 30 epochs, we found that the loss was decreasing substantially upon each passing epoch. Also, the bleu score which captures the similarity between the generated text and the target texts was increasing very much. It was noticed that bleu score typically increases if the ratio of the number of generated tokens are a part of the target text since it incorporates n-gram based matching. We got a score of 0.55 by the end of the 30th epoch. Pretty elated with that score when we performed the testing, it was not giving us a story with high coherency. It was clear that some of the words from the target text were a part of the output. However, there was coherency and continuity missing in the output with a repetition of a same set of characters.

2.2 Temporal Dependency Incorporated Variational Autoencoder

In this section, we present our approach to incorporating temporal dependency into the latent space of the Variational Autoencoder (VAE). By introducing additional layers to the encoding process, we aimed to capture the temporal variation and enhance the coherency of the generated stories.

We trained this model for 30 epochs and observed a smooth loss curve, indicating effective learning. Furthermore, there was a significant increase in the BLEU score, which measures the similarity between the generated and reference texts. However, despite these improvements, we found that the generated stories still lacked coherency, as noted in previous evaluations.

Further investigation revealed that our model was under-trained, and the number of parameters used was insufficient for the complexity of story generation. We came across references suggesting that achieving satisfactory results for a similar model required training for over 82 days. Considering this, we decided to explore a pre-trained model as a starting point, leveraging the captured parameters in the latent space. This approach would allow us to perform additional optimizations and potentially improve the overall performance.

2.3 Fine-tuned Pre-trained model

In our pursuit of improving the coherency and quality of generated stories, we turned to a fine-tuned pre-trained model, specifically the GPT-2 model. GPT-2 (Generative Pre-trained Transformer 2) has demonstrated remarkable performance in natural language processing tasks due to its large-scale training on diverse text sources. With over 1.5 Billion parameters, the promising results achieved by GPT-2, and the extensive resources invested in its training, we made an informed decision to employ this model in our research. By fine-tuning the pre-trained GPT-2 model on our specific story generation task, we aimed to leverage its existing knowledge and optimize it further for coherency. As a standalone existing model, GPT-2 provided a perplexity score of 94.1043 with 3 epochs. While GPT-2's pre-training task focuses on language modeling, fine-tuning allows the model to specialize and improve its performance on a specific downstream task.

We experimented with a batch size of 8 and with learning rates of 0.001 and 0.0001. We used a CrossEntropy loss function. It measures the dissimilarity between the predicted token probabilities and the actual token labels. The GPT2LMHeadModel provides the loss as the first output of the forward method. The AdamW optimizer is used for optimization, which combines the Adam optimizer with weight decay regularization. The optimizer is initialized with the model's parameters and a learning rate. The learning rate is adjusted during training using a linear schedule with warmup. Some other hyperparameters that we used were the max_sequence_length to control the output story token length. When reduced the number of batches to 4, we observed that the perplexity score was improving. We used a scheduler to mitigate the impact of sudden changes during early training stages. So, we incorporated a warmup strategy by gradually increasing the learning rate. This was achieved using the get_linear_schedule_with_warmup from the torch framework's optimizer module. This allowed the model to stabilize and achieve better convergence. This modeling provided us with satisfactory results. We have discussed about the results and the evaluation metrics in detail in the following section.

3. Evaluation and metrics

For the evaluation we considered the following scores:

Bleu score: We had a choice and experimented with Sentence-Bleu and Courpus-Bleu scores. The sentence-level BLEU score measures the similarity between the generated sentences and a set of reference sentences which is useful in evaluating the model's ability to generate individual sentences that align well with the desired output. This provided us the performance in terms of sentence-level fluency, grammaticality, and lexical correctness. But relying solely on sentence-level BLEU scores might not provide a comprehensive evaluation of the model's overall performance. Corpus-level BLEU score considers the performance of the model across the entire generated text corpus, taking into account multiple sentences and their coherence as a whole. This score helps to assess the overall coherence, fluency, and logical flow of the generated text. It is particularly important in tasks like story generation, where the generated sentences should not only be individually coherent but also form a coherent and engaging narrative.

Perplexity: This is another metric we considered for the evaluation of our model. It is a measurement of how well a language model predicts a given sequence of words. It quantifies the model's ability to assign low probability to unlikely sequences and high probability to likely sequences. A lower perplexity value indicates that the model can better predict the next word or sequence of words in a given context, implying a better understanding and fluency in generating coherent text. We have presented our results in Table 1.

It can be seen that the VAE and TD-VAE models had a very high perplexity and it is a clear indication that the model has not learnt from the training provided. However the fine-tuned GPT-2 model has a very good perplexity. The modified baseline GPT-2 model with LR scheduler, Batch size 8 and Learning Rate (LR) of 0.0001 performed the best with a very low perplexity score and a high bleu score. It is also clear that the scheduler did play a role in achieving a convergence faster.

4. Reflection and Ablation study

Despite our efforts, the TD-VAE did not yield the desired results. Fine-tuning a pre-trained model provided us with great results. Hence, in this section, we reflect on the lessons learned and the effects of the changes we made across models.

4.1 Lessons from TD-VAE Implementation

When evaluating the TD-VAE model, we encountered several important lessons. While the model achieved remarkably high BLEU scores, around 0.81, indicating a strong match with the reference text, it did not necessarily correspond to high-quality story generation. This discrepancy highlighted the limitations of relying solely on BLEU scores as a measure of success. The resulting stories produced by the TD-VAE model were not compelling or coherent, indicating the need to consider additional metrics, such as fluency, coherence, and overall narrative quality, to assess the true performance of text generation models. These lessons emphasize the importance of a holistic evaluation approach that goes beyond single metrics to capture the nuanced aspects of generating high-quality and engaging stories. This

Model	Training Loss	Val Loss	Val Perplexity	Test Perplexity	Test Bleu
Variational Auto Encoder	6.1799	6.1010	438.13	449.136	0.19
Temporal dependency induced variational Autoencoder	0.3982	0.7535	198.63	182.813	0.552
GPT-2 baseline model	-	-	94.1043	93.9727	0.59
Modified baseline with Batch8 LR 0.001	4.864	4.541	93.79	93.761	0.07
Modified baseline with Batch8 scheduler LR 0.001	3.187	4.016	55.62	55.8	0.43
Modified baseline with Batch4 scheduler LR 0.0001	3.557	3.752	42.599	42.6855	0.59
Modified baseline with Batch8 scheduler LR 0.0001	3.627	3.744	42.254	42.361	0.47

Table 1: Score evaluation of pre-trained models

approach requires an extensive dataset with well-structured temporal information to effectively capture the dependencies between different parts of the text. Working with TD-VAE implementation served as a valuable lesson in evaluating text generation models.

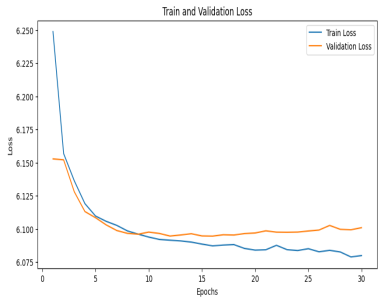


Figure 2: Variational Auto Encoder - Loss curve vs epoch

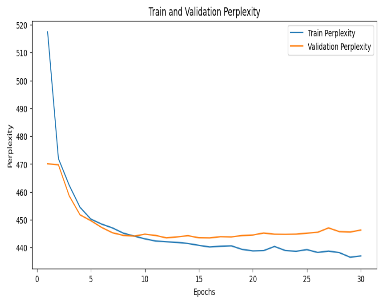


Figure 3: Variational Auto Encoder Perplexity vs Epochs

4.2 Fine tuning the Pre-trained

While working with fine-tuning the pre-trained model with our dataset, finding an optimal learning rate is essential to ensure effective convergence and prevent issues like slow convergence or overshooting. By carefully tuning the learning rate and monitoring the training process, we achieved improved convergence and better model performance. Table 1 shows the various learning rates we tried. From our experimentation, a learning rate of 0.0001 with batch size 4 yielded the best results. This suggested that a smaller learning rate was able to facilitate a more stable optimization process, resulting in better overall model performance. It was also seen that the learning rate of 0.001 has a sudden dip and then increases. However when we reduced the learning rate to 0.0001, we could see better convergence.

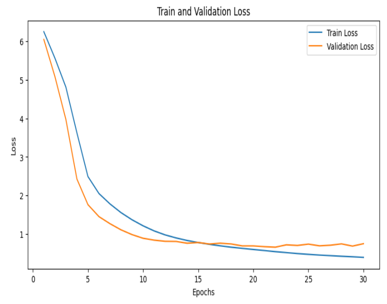


Figure 4: TD-VAE Loss vs Epochs

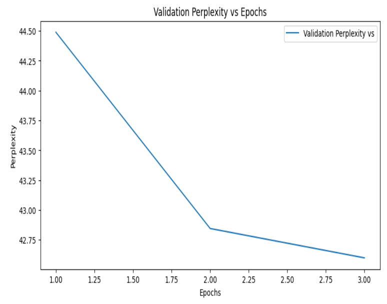


Figure 5: Modified baseline with Batch4 scheduler LR 0.0001 - Perplexity vs Epoch

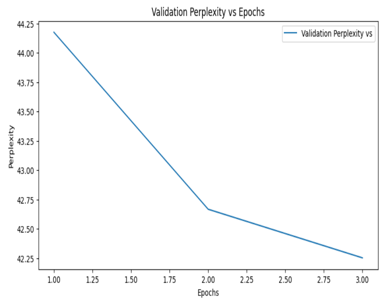


Figure 6: Modified baseline with Batch8 scheduler LR 0.0001

5. Future Work

While this study has provided valuable insights into the implementation and evaluation of the TD-VAE model, fine-tuned GPT-2 models, there are several avenues for future research and improvement that can be explored. Firstly, we have worked with only a part of the dataset for this project. WritingPrompts has over 300,000 samples and is a gold mine for training tasks like this. Hence, as a future work we propose that our model be worked with the entire WritingPrompts dataset. Secondly, the issue of poor convergence and suboptimal results encountered during the training process needs to be further investigated. The architecture of TD-VAE needs to be modified with increased computing resources to see substantial increase in the performance. Incorporation of more recurrent connections, may help capture more intricate temporal dependencies in the story data. Experimenting with different architectural variants or exploring advanced deep learning techniques like transformers in that architecture could lead to improved performance. Experimenting with different architectural configurations or even exploring newer models like GPT-3 or GPT-4 can lead to better performance. One limitation of GPT-2 that we observed was that, it tends to generate verbose and repetitive text. We incorporated top-k sampling to control the diversity of the generated output. However, other techniques like nuclear sampling can be incorporated and experimented. This is an area of active research which has potential applications in the fields of healthcare, legal, finance industries are enormous. As the future work, we plan to leaverage the powerful capabilities of GPT-2 and incorporate advanced techniques like incremental learning to update the model with new information while retaining previously learned knowledge, ensuring its relevance and effectiveness over time.

6. Conclusion

This report is not an amalgamation of just what went right. We proposed to achieve good results using a variation of VAE, while we have the report beyond what was proposed. We have discussed our journey of building a complex language model for automatic story generation. While there are shortcomings in our approach, we have demonstrated an unwavering commitment to improve the existing approaches using the concepts learnt in class. It is clear from our study that, TD-VAE is a very good model for story generation and requires huge computation capabilities. Owing to the limited capabilities, we could not train it enough to see substantial increase in the performance of the model. GPT-2 is a very powerful language model and fine-tuning it with the WritingPrompts dataset gave promising results. In conclusion, we aspire to take this research work forward and story generation is a problem that can be solved using advanced deep learning techniques.

References

Araz, K. 2020. Transformer Neural Networks for Automated Story Generation.

Facebook Research. 2020. Fairseq: Stories Dataset. https://github.com/facebookresearch/fairseq/tree/main/examples/stories. Accessed: 2023-04-04.

Gregor, K.; Papamakarios, G.; Besse, F.; Buesing, L.; and Weber, T. 2018. Temporal difference variational auto-encoder. *arXiv preprint arXiv:1806.03107*.

Jain, P.; Agrawal, P.; Mishra, A.; Sukhwani, M.; Laha, A.; and Sankaranarayanan, K. 2017. Story generation from sequence of independent short descriptions. *arXiv preprint arXiv:1707.05501*.

Kaggle. 2020. Kaggle generate stories. https://www.kaggle.com/datasets/emily2008/story-text. Accessed: 2023-04-04.

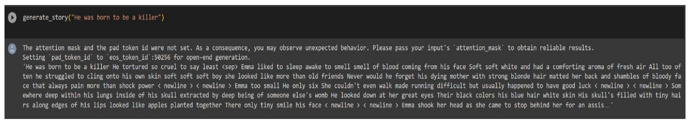
Khandelwal, U.; He, H.; Qi, P.; and Jurafsky, D. 2018. Sharp nearby, fuzzy far away: How neural language models use context. *arXiv preprint arXiv:1805.04623*.

Radford, A.; Wu, J.; Child, R.; Luan, D.; Amodei, D.; Sutskever, I.; et al. 2019. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8): 9.

Wilmot, D.; and Keller, F. 2021. A temporal variational model for story generation. *arXiv preprint arXiv:2109.06807*.

Yu, L.; Zhang, W.; Wang, J.; and Yu, Y. 2017. Seqgan: Sequence generative adversarial nets with policy gradient. In *Proceedings of the AAAI conference on artificial intelligence*, volume 31.

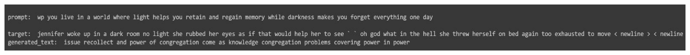
Appendix



```
generate_story("He was born to be a killer")

The attention mask and the pad token id were not set. As a consequence, you may observe unexpected behavior. Please pass your input's `attention_mask` to obtain reliable results.
Setting `pad_token_id` to `eos_token_id`:50256 for open-end generation.
'He was born to be a killer He tortured so cruel to say least <sep> Emma liked to sleep awake to smell smell of blood coming from his face Soft soft white and had a comforting aroma of fresh air All too of
ten he struggled to cling onto his own skin soft soft soft boy she looked like more than old friends Never would he forget his dying mother with strong blonde hair matted her back and shambles of bloody fa
ce that always pain more than shock power < newline > < newline > Emma too small He only six She couldn't even walk made running difficult but usually happened to have good luck < newline > < newline > Som
ewhere deep within his lungs inside of his skull extracted by deep being of someone else's womb He looked down at her great eyes Their black colors his blue hair white skin His skull's filled with tiny hai
rs along edges of his lips looked like apples planted together There only tiny smile his face < newline > < newline > Emma shook her head as she came to stop behind her for an assis...'
```

Figure 7: Result of Story generation by the Best Model



```
prompt:  wp you live in a world where light helps you retain and regain memory while darkness makes you forget everything one day

target:  jennifer woke up in a dark room no light she rubbed her eyes as if that would help her to see ` ` oh god what in the hell she threw herself on bed again too exhausted to move < newline > < newline
generated_text:  issue recollect and power of congregation come as knowledge congregation problems covering power in power
```

Figure 8: Result of Story generation by TD-VAE