Generating Two Sentence Horror Stories using Novel Dataset

Arvin Ohanian
arvinoha@usc.edu

Yusufu Feierdawusi
feierdaw@usc.edu

Divam Kesharwani
kesharwa@usc.edu

Vidisha Kudalkar
kudalkar@usc.edu

Mahmoudreza Dehghan
mrezad@usc.edu

Introduction

As AI models improve in the field of NLP, we have seen major strides in novel text generation. However, one of the areas which we noticed models having a particularly hard time with was generating scary stories. Most stories would be generic and oftentimes would repeat the same sections in different stories. Our goal is to create a model that can generate stories that are both varied and scary. The first parameter we need to tackle however is how long the stories are going to be. We decide to have the stories be two sentences due to the dataset we use. By scraping Reddit's r/TwoSentenceHorror subreddit, we create a database of two sentence horror stories. Our solution plans to solve the problem of poor horror story generation by using a new dataset that we create to bolster the models understanding of horror.

Related Work

Text generation is crucial for question answering, translation, summarization, conversation, etc. Language generation ranges from simple template-based systems that uses rules to generate natural language text to machine-learned systems that have a complex understanding of grammar. Various strings of works are based on template based models and they improve the language generation by using the statistical methods [4,5,6]. The early versions of language generation use data driven or rule based methods.

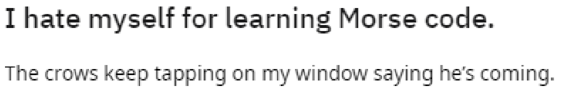
Larger datasets became available in the last few years and it led to the progress of several language generation tasks. Deep neural network models have become popular recently as language generation is possible by training DNN models on a large dataset.

[8] describes the transformer architecture which uses an encoder-decoder implemented using the self-attention mechanism that is being adopted by new language generation systems. Some researchers are also working on generating text that has better flow [9,10].

GPT 3 [3] is a language model created by OpenAI which was trained on hundreds of billions of words. GPT3 can generate longer text which is almost similar to the human generated texts. GPT-3 is a general purpose language model and can be applied to a wide variety of tasks. GPT-3 has proven capable of generating poetry and articles, computer code for web interfaces, and product or job descriptions[2]. The authors of [1] have developed a method to automatically generate an email reply using GPT3. Due to the robustness and generality of GPT3, we are using it for generating horror stories.

Novel Dataset Creation

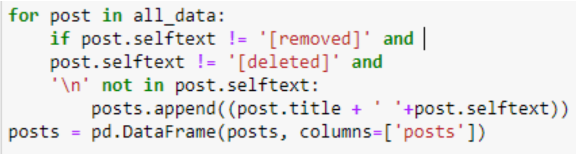
Our dataset is curated from scraping reddit data through its API. The API allows us to specify which subreddit we want to take posts from, as well as what part of the post we want. In our case, we only want the title and the body of the post since in the r/TwoSentenceHorror subreddit, the format people use involves the first sentence being in the title and the second sentence being the body as seen in the image below.



I hate myself for learning Morse code.

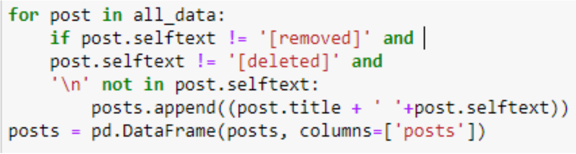
The crows keep tapping on my window saying he's coming.

Using the API we scrape ten thousand of the most recent posts from the subreddit and combining the titles and post bodies gives us ten thousand examples of two sentence horror stories for our model to train on. Cleaning the data was crucial since we realized that a significant portion of the data we were scraping was deleted or removed content. This happens when a moderator or the user who posted decides to remove the post for either breaking the subreddit rules or simply because the user wants it removed. After removing these posts from our database we find that we lose approximately half of the scraped data. Since we decided that we want to train on ten thousand examples, we therefore scraped about 21,000 posts.



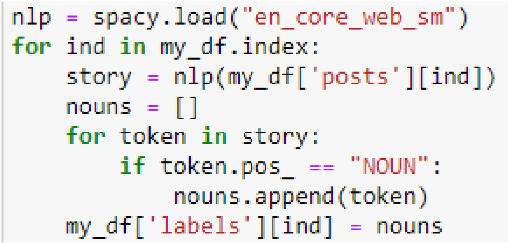
```
for post in all_data:
    if post.selftext != '[removed]' and |
    post.selftext != '[deleted]' and
    '\n' not in post.selftext:
        posts.append((post.title + ' '+post.selftext))
posts = pd.DataFrame(posts, columns=['posts'])
```

Once we had obtained the posts, we needed labels that could be used during the training process.



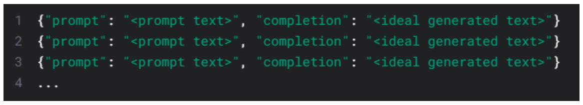
```
for post in all_data:
    if post.selftext != '[removed]' and |
    post.selftext != '[deleted]' and
    '\n' not in post.selftext:
        posts.append((post.title + ' '+post.selftext))
posts = pd.DataFrame(posts, columns=['posts'])
```

Our model takes two prompt words which it uses to generate a two sentence horror story based on the given words. Therefore, we needed a heuristic that could describe the posts most accurately. One method tried was simply taking all the nouns from the post and using two of them. This ended up working, but would tend to create stories that weren't scary since the model would try to use the nouns in favor of making the story scary and ultimately failed to produce the results we wanted. We then tried using a keyword extractor that would analyze the post and extract only the most important two words to use as labels. This also failed however since most keyword extractors tend to label phrases as well as words. This meant that the keyword extractor would take the most important phrases from the post which would be several words, rather than just the most important words. The keyword extractor also extracted single words but they tended to perform worse than simply taking the nouns from the posts. The keyword extractor tended to favor words such as 'with' or 'and' which wouldn't be viable prompt words for our model. Finally, we decided to try using the first and last noun of the post as our labels. This would make the prompt words usually take significant words from both the first and second sentences, resulting in words that described the post in its entirety much better.



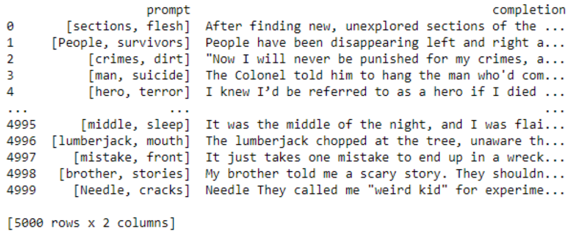
```
nlp = spacy.load("en_core_web_sm")
for ind in my_df.index:
    story = nlp(my_df['posts'][ind])
    nouns = []
    for token in story:
        if token.pos_ == "NOUN":
            nouns.append(token)
    my_df['labels'][ind] = nouns
```

Once we had both our labels and our completion sentences, we were ready to do training. One final step however was to format the database properly for fine tuning on GPT-3. This included adding 'prompt' in front of all the labels and then having the associated story have 'completion' added in front of it. Stop words were also necessary since the model would need to know what separates every example. For this we simply used '\n', indicating a new line.



```
{"prompt": "<prompt text>", "completion": "<ideal generated text>"}
{"prompt": "<prompt text>", "completion": "<ideal generated text>"}
{"prompt": "<prompt text>", "completion": "<ideal generated text>"}
...
```

Here is an example of the prompt/completion format.



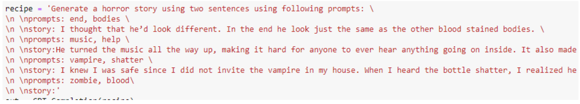
```
                 prompt                                         completion
0       [sections, flesh]  After finding new, unexplored sections of the ...
1     [People, survivors]  People have been disappearing left and right a...
2          [crimes, dirt]  "Now I will never be punished for my crimes, a...
3          [man, suicide]  The Colonel told him to hang the man who'd com...
4          [hero, terror]  I knew I'd be referred to as a hero if I died ...
...                   ...                                                ...
4995      [middle, sleep]  It was the middle of the night, and I was flai...
4996 [lumberjack, mouth]  The lumberjack chopped at the tree, unaware th...
4997     [mistake, front]  It just takes one mistake to end up in a wreck...
4998   [brother, stories]  My brother told me a scary story. They shouldn...
4999     [Needle, cracks]  Needle They called me "weird kid" for experime...

[5000 rows x 2 columns]
```

Example of what the database looks like.

Few Shot Learning and Fine Tuning GPT-3

The tuning process for GPT-3 begins by choosing the appropriate model. We chose the best available model, davinci, to do our tuning on. The training process itself is done on console using the OpenAI API and takes in our prompts as input and the generated stories as reference completions. Using prompt learning, a process by which the base davinci model remains mostly static while using our new data to adjust only some parameters, the final model is able to understand natural language while being fine tuned in generating stories like we want. What we found was that using this prompt learning in combination with few shot learning produced the best result. Few shot learning is a method of training that uses multiple examples to give the model an understanding of what structure we would like to see. For example, we gave the model a few examples of two sentence horror stories and their respective prompt words, and then asked the model to generate new stories based on updated prompt words. This approach resulted in many more outputs that used only two sentences rather than three or four sentences which was much more common without using few shot learning.

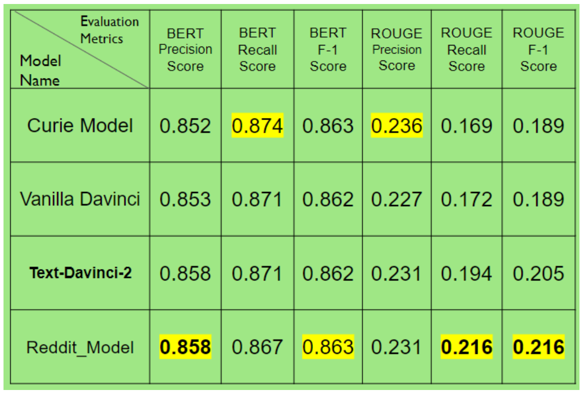


```
recipe = 'Generate a horror story using two sentences using following prompts: \
\n \nprompts: end, bodies \
\n \nstory: I thought that he’d look different. In the end he look just the same as the other blood stained bodies. \
\n \nprompts: music, help \
\n \nstory:He turned the music all the way up, making it hard for anyone to ever hear anything going on inside. It also made
\n \nprompts: vampire, shatter \
\n \nstory: I knew I was safe since I did not invite the vampire in my house. When I heard the bottle shatter, I realized he
\n \nprompts: zombie, blood\
\n \nstory:'
```

Here is an example of the prompt we used to do our training with

Baseline Models and Evaluation

To evaluate the model we needed a comparison to other models. We decided to take other OpenAI models such as curie, text davinci 2, and vanilla davinci. Curie is a weaker model than davinci used to make summaries, text davinci 2 is the best model currently available on OpenAI, and vanilla davinci is the model we used to train our model except this time it doesn’t have our reddit data tuning. These three models were all trained in the same way as our Reddit Model and used as benchmark testing. The benchmark we chose to employ were BERT and ROUGE which measure how coherent the stories are in terms of meanings of words and grammar. Also, these measures also take into account how related the first and second sentences are, such as whether the second sentence is related to the first. BERT and ROUGE both use the reference data that we used for training as the reference as well for each model.

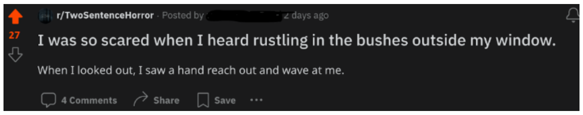


Evaluation Metrics / Model Name	BERT Precision Score	BERT Recall Score	BERT F-1 Score	ROUGE Precision Score	ROUGE Recall Score	ROUGE F-1 Score
Curie Model	0.852	0.874	0.863	0.236	0.169	0.189
Vanilla Davinci	0.853	0.871	0.862	0.227	0.172	0.189
Text-Davinci-2	0.858	0.871	0.862	0.231	0.194	0.205
Reddit_Model	**0.858**	0.867	0.863	0.231	**0.216**	**0.216**

Here are the final results of our tests. We measured the precision, recall, and f1 score for each model on both ROUGE and BERT. We can see that our model outperformed all other models on four of the metrics and lost to the curie model on two metrics. Surprisingly, the best model on OpenAI, Text-Davinci-002 didn’t get the highest score on any of the metrics.

Human Evaluation

The BERT and ROUGE scores are good metrics for sentence structure and overall meaning but they don’t tell us how scary the stories are or whether our model outperforms the others in terms of how scary the stories it generates are. For this we decided that we needed human evaluation of how scary the stories of each model are. Therefore we decided that we could post the generated stories on the r/TwoSentenceHorror subreddit and take the average upvotes for each model. In the end, we found that the highest upvote average was our model with an average of about 15 upvotes while all other models had an average of below 10. This showed us that our model did improve the quality of the stories and made scarier stories overall. Our highest upvoted post has 27 upvotes and is from our tuned model as well which also lets us know that our model outperformed the other models in the human evaluations.



r/TwoSentenceHorror · Posted by [redacted] 2 days ago

27

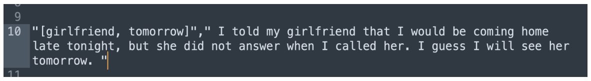
I was so scared when I heard rustling in the bushes outside my window.

When I looked out, I saw a hand reach out and wave at me.

This is our highest upvoted post during the human evaluations and is generated by our reddit model.

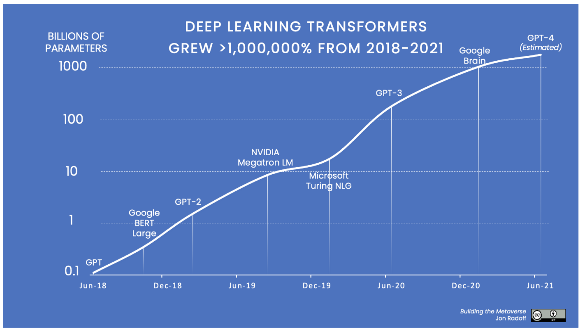
Results and Discussion

Based on the evaluation metrics compared to other baseline models, we can conclude that our model is viable and a competent generative model. Additionally, with the results of the human evaluation from reddit, we can see that the stories generated from our model are generally scary, which is a metric that is difficult to evaluate objectively. However, the output results are not fully accurate either, there were few sentences produced by the model that did not really make sense as a story or that it was not a scary story at all. As we can see from the example below that this is just an event that has occurred without any implication of horror, but uses the two prompts to construct this story.



```
"[girlfriend, tomorrow]"," I told my girlfriend that I would be coming home
late tonight, but she did not answer when I called her. I guess I will see her
tomorrow. "
```

In order to improve the quality of the model's results, we have a few adjustments we can make in the future. The first one is to improve the quality of our novel dataset. We currently collect the top rated posts from the subreddit r/TwoSentenceHorror, we can add a layer of manual inspection to this process, meaning that we can subjectively rate each datapoint sentence based on how scary it is. We can then remove sentences using this baseline as a reference. Another way of improving the quality of our model is to fine-tune a model that outperforms GPT-3 generative models. Thus, we can use the new GPT-4 generative model that is said to be released sometime in 2023.



Based on the comparison chart above we can see that the GPT-4 model has trained on well over one trillion parameters and will most likely outperform any GPT-3 models. It is also said that GPT-4 models will be available to fine-tune as soon as they are released. This means that we do not have to fine-tune the older versions of the model like we did with GPT-3 Text-Davinci [7].

References

[1] Thiergart, J., Huber, S., & Übellacker, T. (2021). Understanding Emails and Drafting Responses--An Approach Using GPT-3. *arXiv preprint arXiv:2102.03062*.

[2] Dale, R. (2021). GPT-3: What's it good for? Natural Language Engineering. https://doi.org/10.1017/S1351324920000601

[3] Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J. D., Dhariwal, P., ... & Amodei, D. (2020). Language models are few-shot learners. *Advances in neural information processing systems*, *33*, 1877-1901.

[4] M. Sporleder, C.—Lapata. Discourse chunking and its application to sentence compression. In Proceedings of HLT/EMNLP, pp. 257–264, 2005.

[5] K. Steinberger, J.—Jezek. Sentence compression for the lsa-based summarizer. In Proceedings of the 7th International Conference on Information Systems Implementation and Modelling, pp. 141–148, 2006.

[6] D. Knight, K.—Marcu. Statistics-based summarization – step one: Sentence compression. In In Proceeding of The 17th National Conference of the American Association for Artificial Intelligence, pp. 703–710, 2000.

[7] Alberto Romero, GPT-4 Is Coming Soon. Here's What We Know About It. https://towardsdatascience.com/gpt-4-is-coming-soon-heres-what-we-know-about-it-64db058cfd45, 2022.

[8] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In NeurIPS, pp. 5998–6008, 2017

[9] Lili Yao, Nanyun Peng, Ralph M. Weischedel, Kevin Knight, Dongyan Zhao, and Rui Yan. Plan-and-write: Towards better automatic storytelling. In CoRR, volume abs/1811.05701, 2018. URL http://arxiv.org/abs/1811.05701

[10] Hannah Rashkin, Asli Celikyilmaz, Yejin Choi, and Jianfeng Gao. Plotmachines: Outlineconditioned generation with dynamic plot state tracking. In arxiv, 2020.

Links:

Link to our code: https://github.com/ArvinOhanian/Generating-Two-Sentence-Horror-Stories-using-Novel-Dataset

https://beta.openai.com/docs/guides/fine-tuning
Guide to fine tuning models using OpenAI

https://beta.openai.com/docs/quickstart
Quickstart guide for training OpenAI models using their API.

https://beta.openai.com/docs/api-reference/fine-tunes
Fine tuning API and training resource.

https://www.reddit.com/dev/api/
Documentation for how to use the reddit api to access subreddit data. This is what we used to create our dataset.

Distribution of Work

Arvin Ohanian: Wrote code for scraping reddit data with API and creating dataset. Did training for all models. Helped write the final report.

Yusufu Feierdawusi: Wrote code for metrics evaluation and collected reddit feedback based on our outputs. Helped scraping reddit data and creating a dataset. Helped write the final report.

Divam Kesharwani: Helped in scrapping reddit data and checking for the results of generated stories by posting on reddit. Also helped write the final report.

Vidisha Kudalkar: Helped in generating dataset, training the models and writing the reports.

Mahmoudreza Deghan: Helped with the evaluation methods, organized the dataset into excel. Help with writing the final report.