NLP Project:
Woby's Spooky Tales

George Washington University
Natural Language Processing - DATS 6312
Group 4
Joshua Ting

https://github.com/justjoshtings/Final-Project-Group4

Introduction

- Recent language models have the ability to produce text in various genres and domains where humans are not aware they are computer generated.
- Horror stories have been an integral part of humanity's outlet to explore and bring to life our collective deepest fears and imagination.

Research Question

Can we create a language model that can generate coherent horror stories for readers who enjoy a good scare?

What Makes a Good Horror Story?

A well written horror story will need to understand:

- The semantics and structure of a particular language
- Different nuanced elements that elicit a response from our primal fight or flight instincts

Prior Similar Works

- In 2017, a group of people leveraged deep learning to create Shelley, a Twitter bot to complete scary stories from Twitter users.

- The actual architecture of Shelley was never released to the public but since this was developed in/prior to 2017, it is likely some sort of RNN architecture.

Corpus Source

Number	SubReddit Name	Link
1	r/nosleep	Link
2	r/stayawake	Link
3	r/DarkTales	Link
4	r/LetsNotMeet	Link
5	r/shortscarystories	Link
6	r/Thetruthishere	Link
7	r/creepyencounters	Link
8	r/truescarystories	Link
9	r/Glitch_in_the_Matrix	Link
10	r/Paranormal	Link
11	r/Ghoststories	Link
12	r/libraryofshadows	Link
13	r/UnresolvedMysteries	Link
14	r/TheChills	Link
15	r/scaredshitless	Link
16	r/scaryshortstories	Link
17	r/Humanoidencounters	Link
18	r/DispatchingStories	Link

Metrics: Perplexity

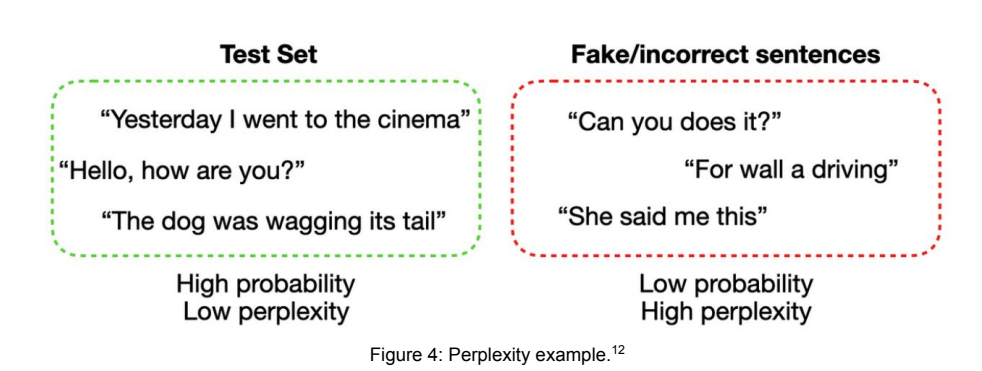
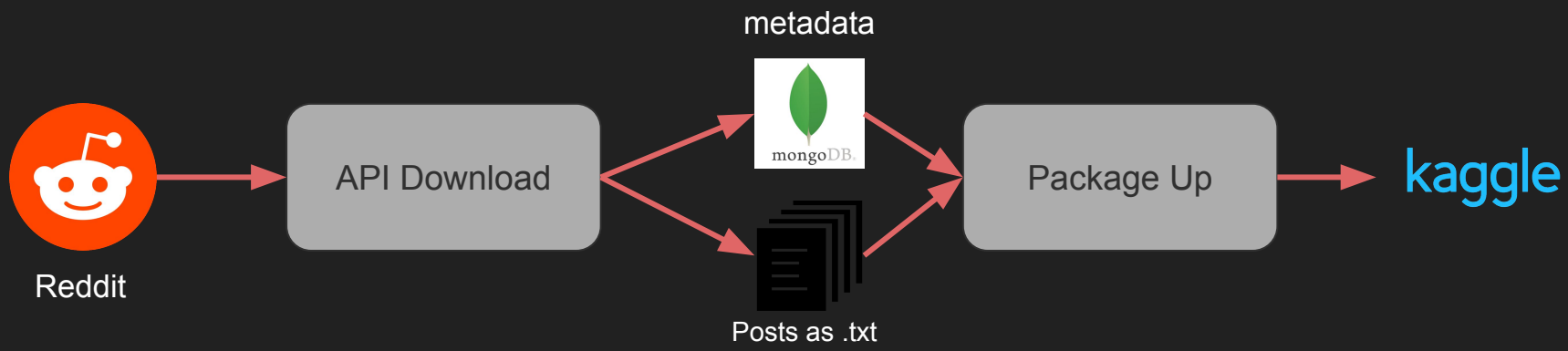


Figure 4: Perplexity example.[12]

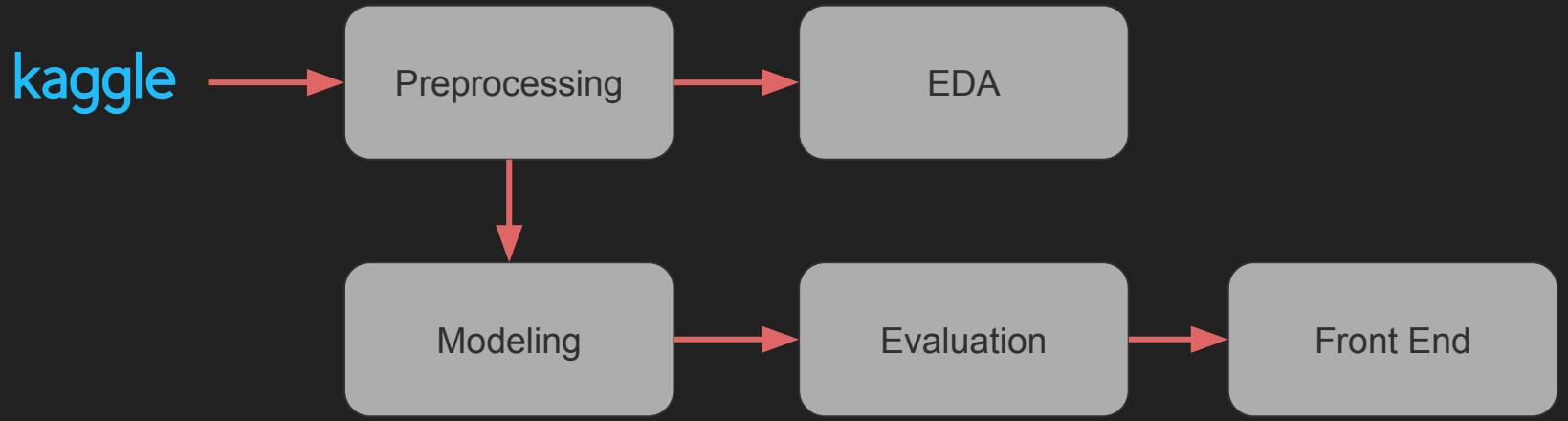
$$P(M) = e^{H(L,M)}$$

where, H is cross entropy loss, M is the true language model, L is generated data

Code Architecture: Data Acquisition Pipeline



Code Architecture: Preprocessing & Modeling Pipeline



Data Acquisition

- Python Reddit API Wrapper (PRAW)[6]
  - Official Reddit API Wrapper
  - Runs into issues of 1000 posts limit per SubReddit available
- Python Pushshift.io API Wrapper (PSAW)[7]
  - Community open-sourced API wrapper
  - Collects more posts than PRAW but can run into issues where certain hosting shards are not online
- We used both in order to have the widest coverage then removed any duplicates afterwards
- ~90 MB, ~15k posts

Data Storage: Local Disk/MongoDB

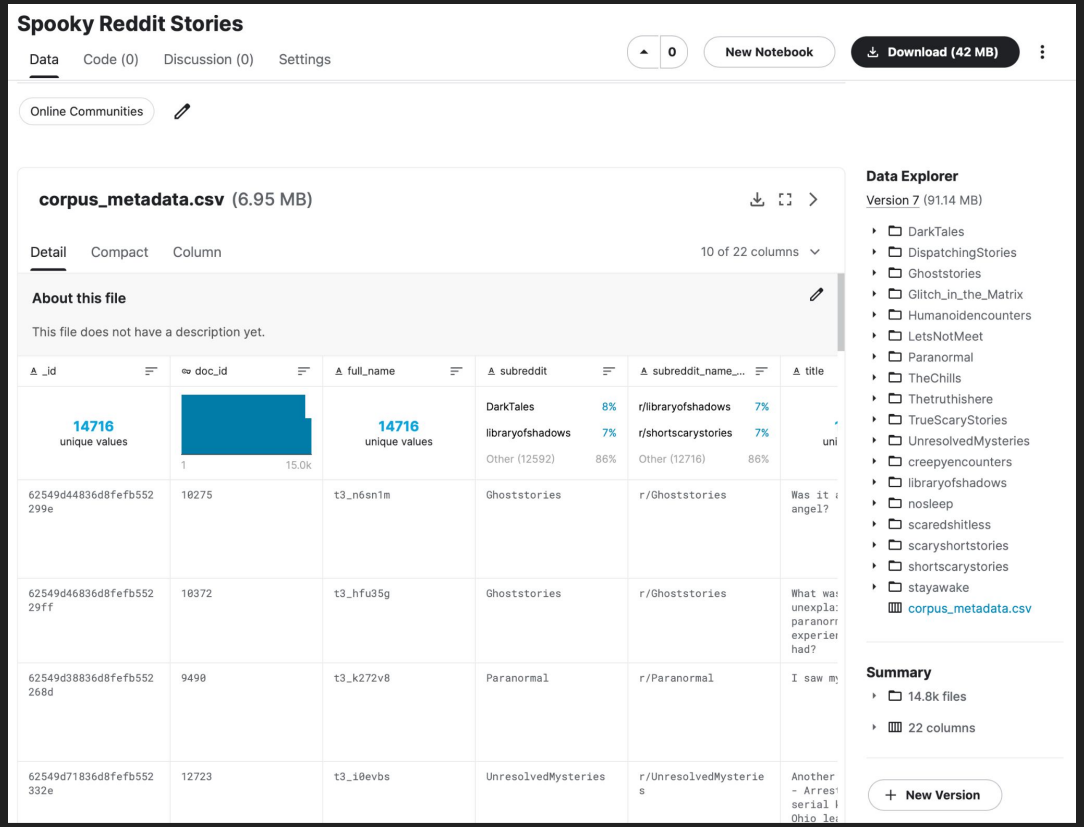
Metadata Saved to MongoDB

```
data_dict = {
            'doc_id': ,
            'full_name': ,
            'subreddit': ,
            'subreddit_name_prefixed': ,
            'title': ,
            'little_taste': ,
            'selftext': ,
            'author': ,
            'upvote_ratio': ,
            'ups': ,
            'downs': ,
            'score': ,
            'num_comments': ,
            'permalink': ,
            'kind': ,
            'num_characters': ,
            'num_bytes': ,
            'created_utc': ,
            'created_human_readable': ,
            'filepath': ,
            'train_test':
      }
```

Post Text Saved as .txt to Disk

```
FINAL-PROJECT-GROUP4
├── Code
├── Woby_Log
├── ...
│
└── Corpus
|     |
|     └──nosleep
|     |     | 1_t3_diuucz.txt
|     |     | 2_t3_dyqd5e.txt
|     |     | ...
|     |
|     └──creepyencounters
|     |     | 5931_t3_i3l009.txt
|     |     | 5931_t3_i3l009.txt
|     |     | ...
|     |
|     └──Ghoststories
|     |     | 9845_t3_jdedeb.txt
|     |     | 9846_t3_hvu2ko.txt
|     |     | ...
```

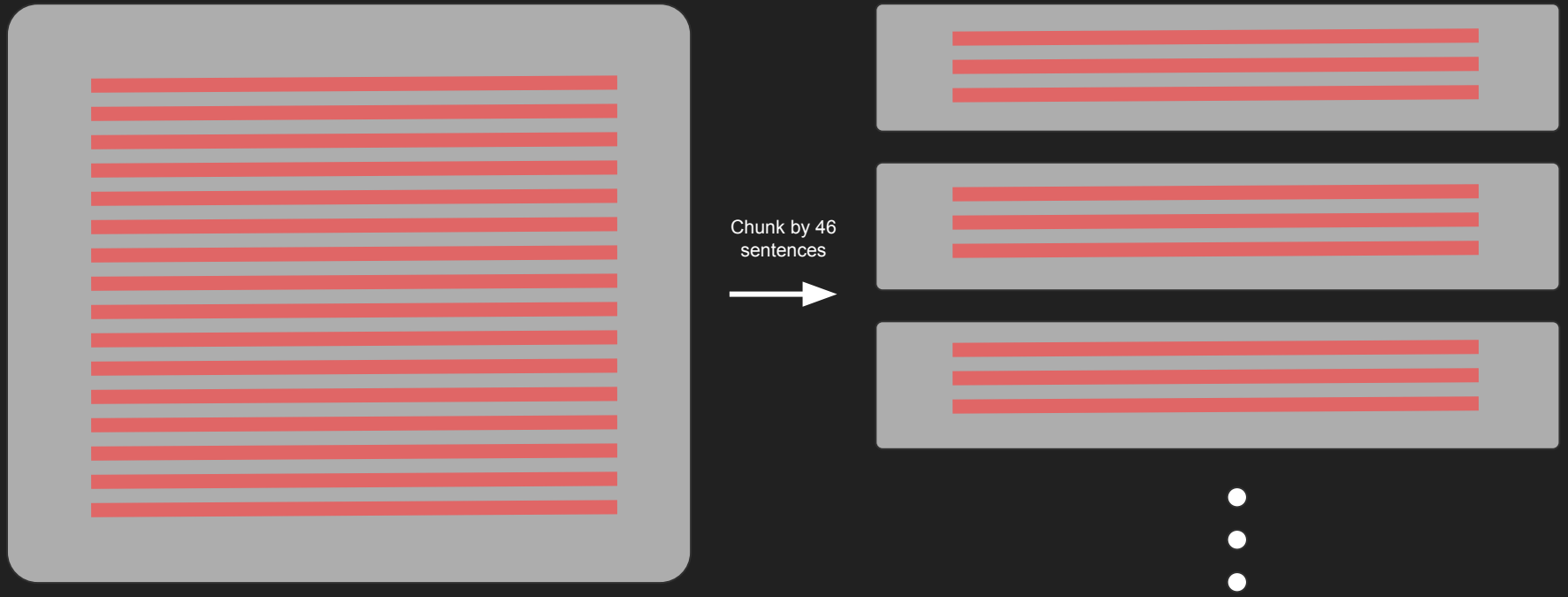
Data Storage: Kaggle



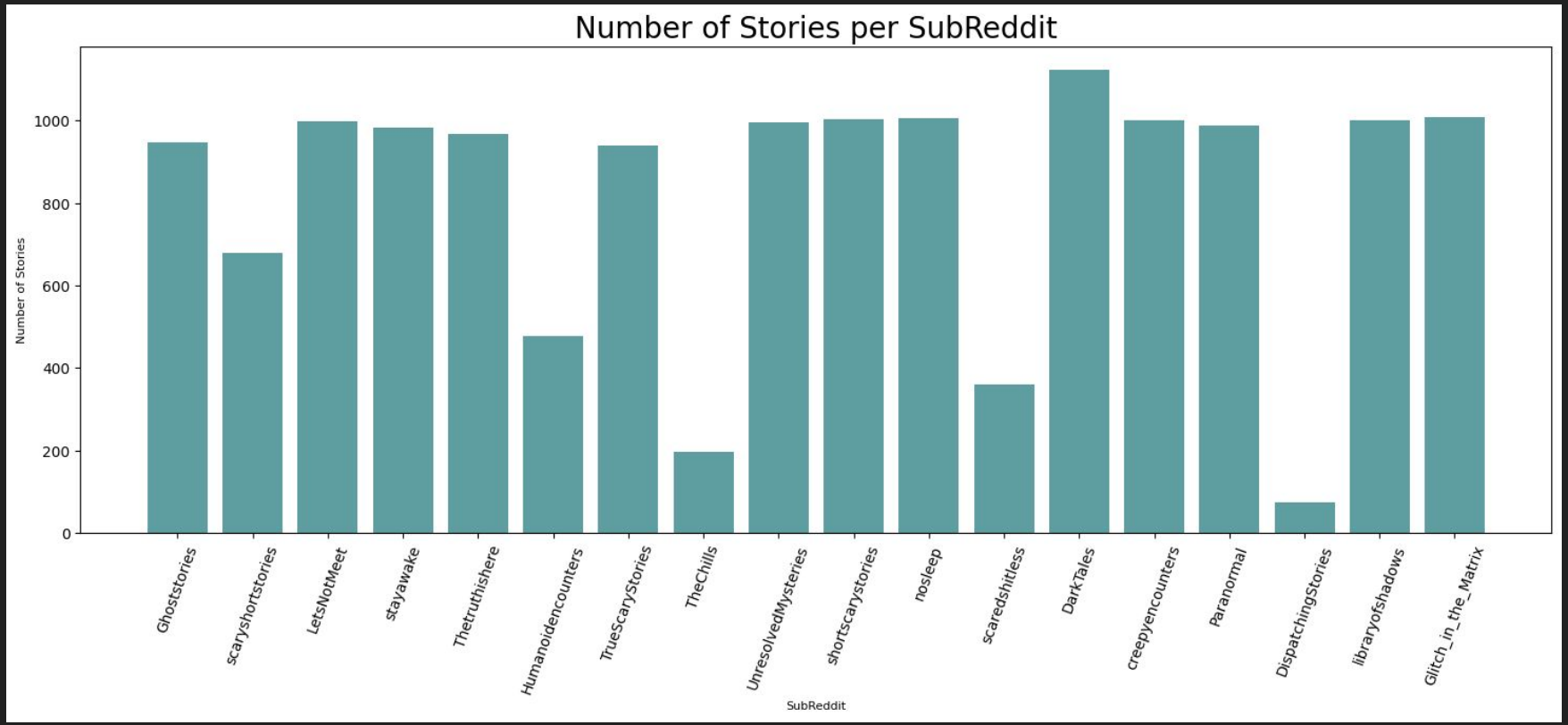
Data Preprocessing: Cleaning

Step	Cleaning Task	Description
1	Remove all text after: “TLDR”, “TLDR:”, “TL;DR”, “TL DR”, “TL DR:”.	TLDR stands for Too Long Didn’t Read and the text that follows often is not part of the actual story.
2	Remove all links	Not relevant for stories.
3	Remove “&amp” and “”	These are HTML elements that are not needed in our corpus.
4	Remove “***” or more *.	These are often used for formatting purposes and are not needed for our corpus.

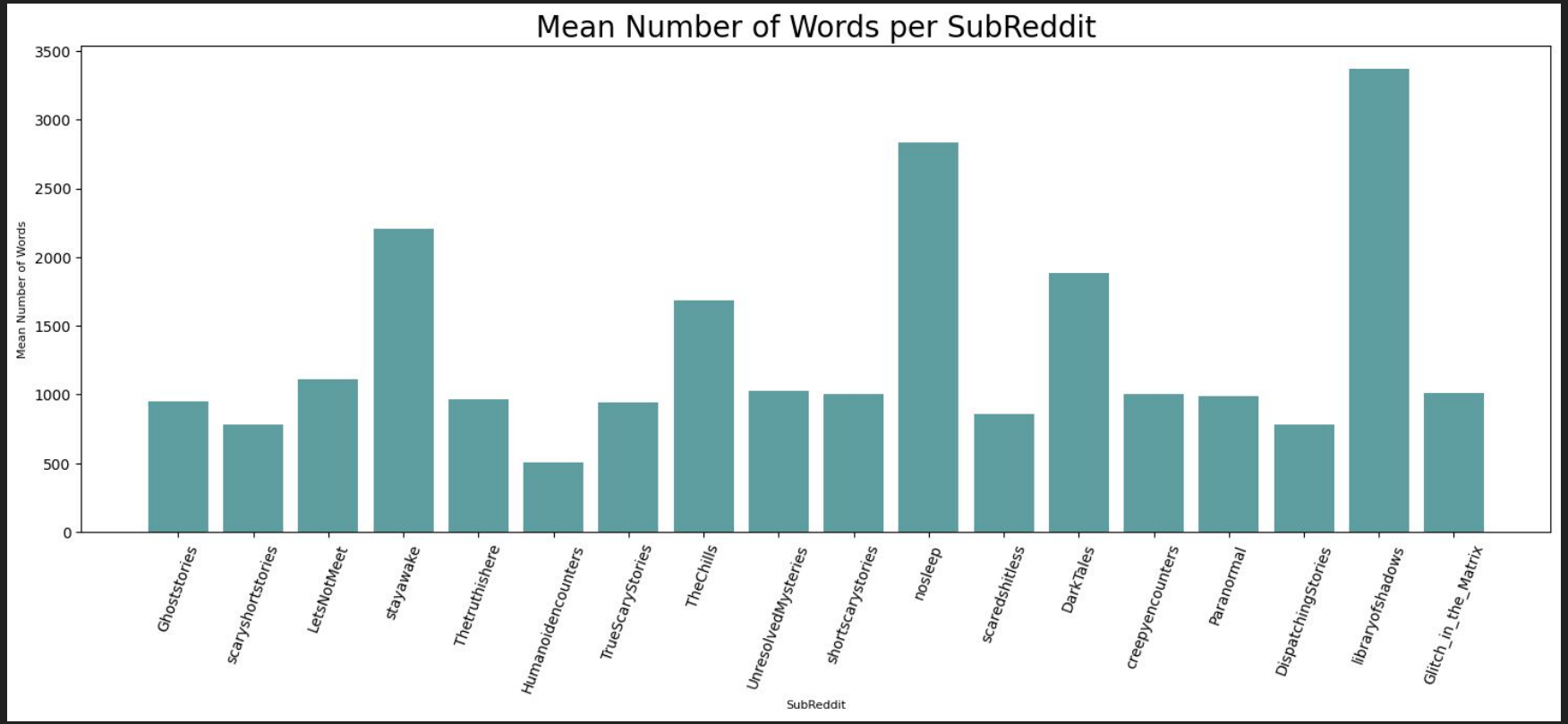
Data Preprocessing: Sentence Chunking



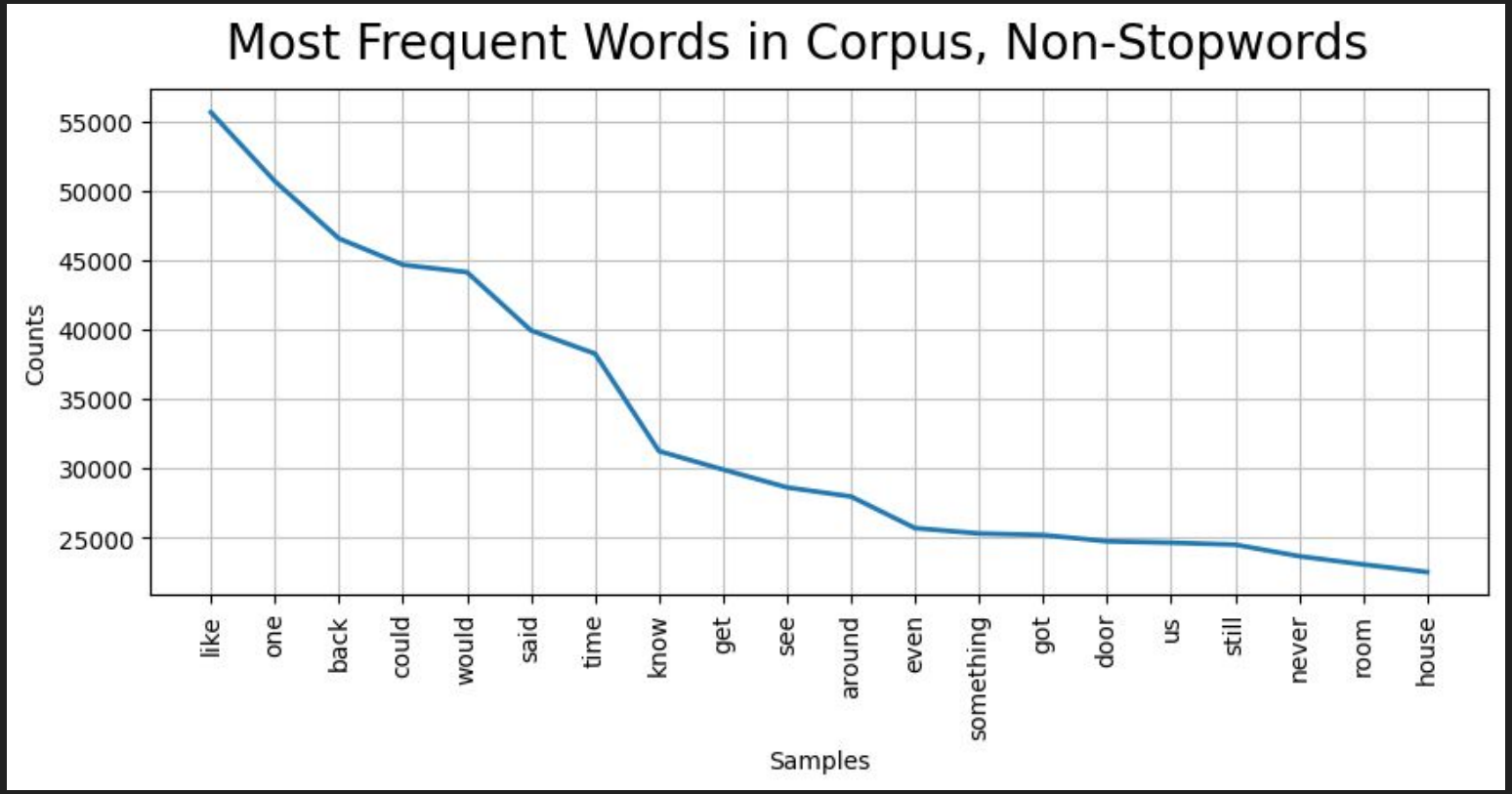
EDA: Number of Stories per SubReddit



EDA: Mean Number of Words per SubReddit



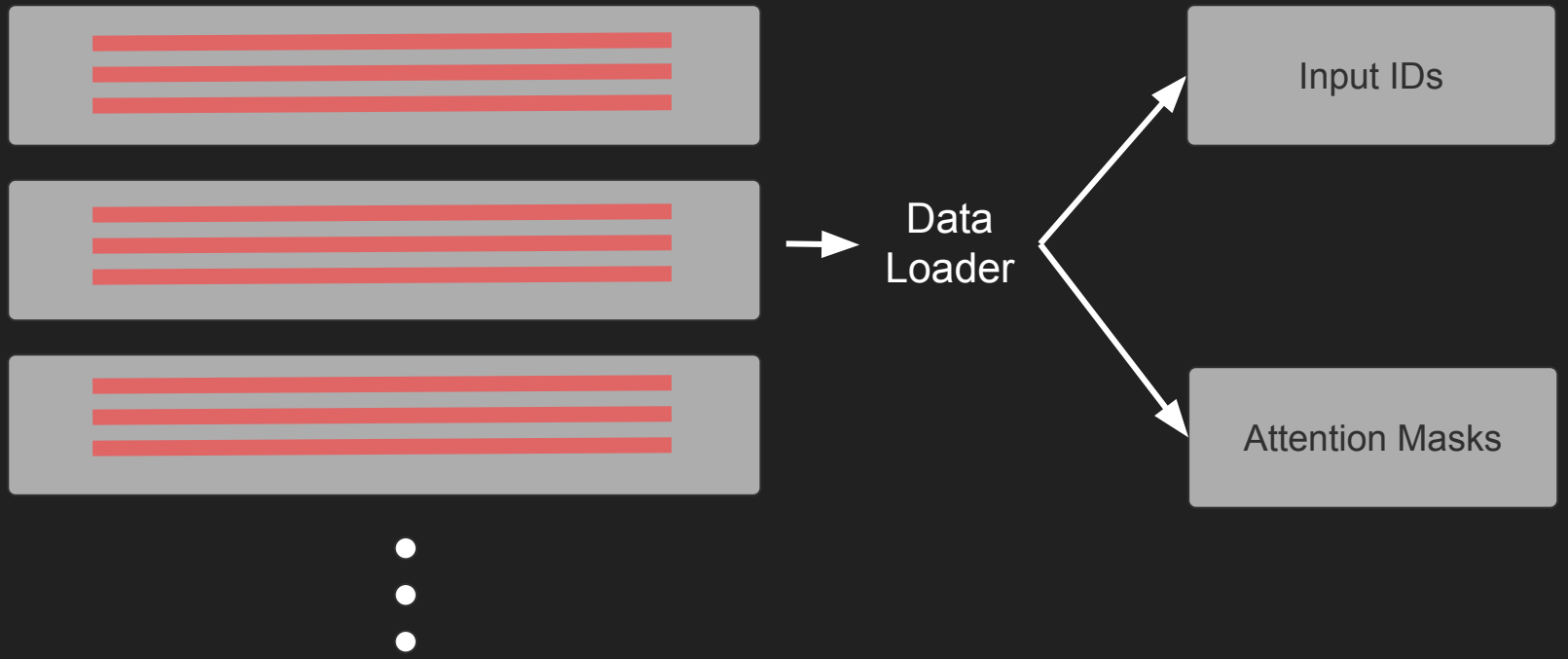
EDA: Most Frequent Words in Corpus



Modeling Options: Autoregressive Transformers

1. **GPT2** fine tuned on our corpus for text generation.[8]
2. **GPT-NEO** fine tuned on our corpus for text generation.[9]
3. A **custom GPT2 variant** pretrained on our corpus then again fine tuned on our corpus for text generation, GPT2Spooky

Data Loader



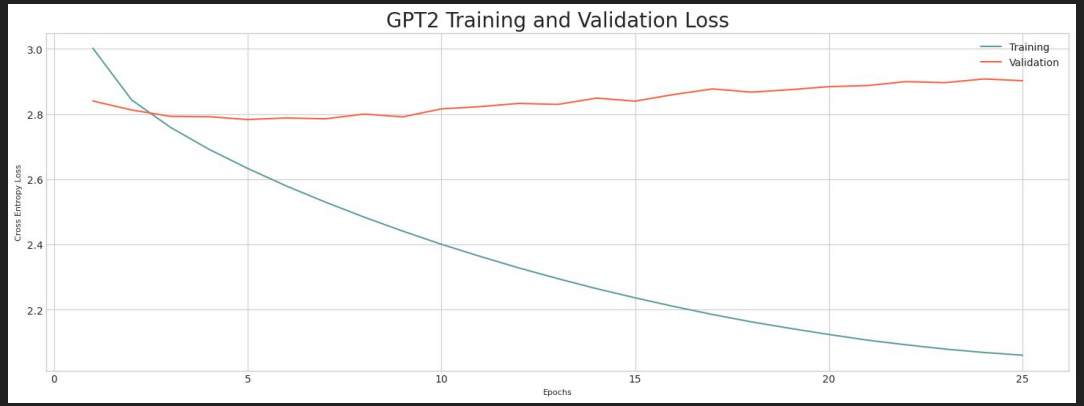
GPT2

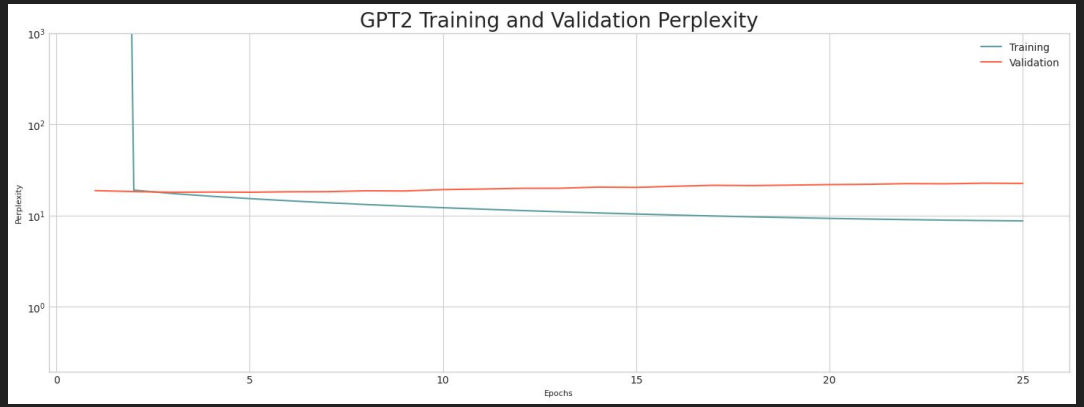
- OpenAI
- 1.5 billion parameters
- Trained on a dataset of 8 million web pages
- Predict the next word
- Hugging Face’s ‘gpt2’ configuration: 117M parameters

GPT2 Fine Tuning

Configuration	Value
Tokenizer	`from transformers import GPT2Tokenizer`
Model Head	`from transformers import GPT2LMHeadModel`
Optimizer	`from torch.optim import AdamW`
Custom Tokens	`bos_token='<|startoftext|>', eos_token='<|endoftext|>', pad_token='<|pad|>'`
Number of Epochs	25
Learning Rate	5e-5
Learning Rate Scheduler	Linear
Batch Size	1
Max Input Length	768 Tokens
Model Type	‘gpt2’
Seed	42
Loss	Cross Entropy
Metric	Perplexity, Equation (3)
Number of Parameters	117M
Pretrained On	8 million web pages

GPT2 Training





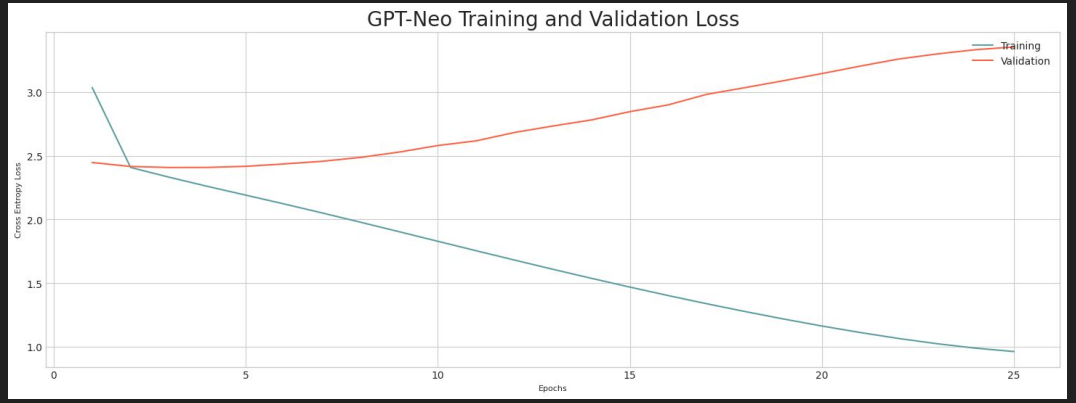
GPT-Neo

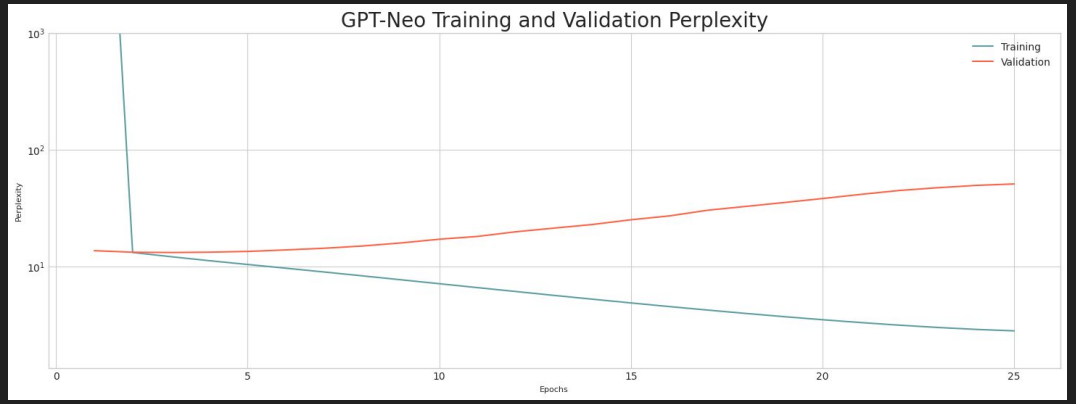
- Released in the EleutherAI/gpt-neo repository by Sid Black, Stella Biderman, Leo Gao, Phil Wang and Connor Leahy
- Similar to GPT2 except that GPT Neo uses local attention in every other layer with a window size of 256 tokens
- Trained on the Pile dataset (>800GB)
- GPT-Neo has several model versions with the largest being 2.7B parameters

GPT-Neo Fine Tuning

Configuration	Value
Tokenizer	`from transformers import GPT2Tokenizer`
Model Head	`from transformers import GPTNeoForCausalLM`
Optimizer	`from torch.optim import AdamW`
Custom Tokens	`bos_token='<|startoftext|>', eos_token='<|endoftext|>', pad_token='<|pad|>'`
Number of Epochs	25
Learning Rate	5e-5
Learning Rate Scheduler	Linear
Batch Size	1
Max Input Length	1024 Tokens
Model Type	‘EleutherAI/gpt-neo-125M’
Seed	42
Loss	Cross Entropy
Metric	Perplexity, Equation (3)
Number of Parameters	125M
Pretrained On	Pile dataset

GPT-Neo Training





GPT2Spooky

- Custom pretrained model on our own corpus
- Uses the same architecture as the first model of GPT2, except it is completely pretrained on just our own custom corpus.

GPT2Spooky Custom Corpus Pretraining

Configuration	Value

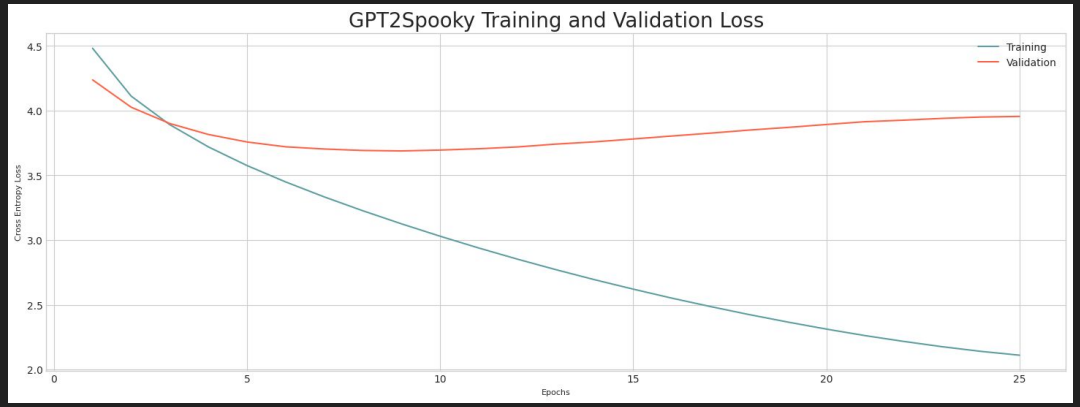
Tokenizer	`from tokenizers import ByteLevelBPETokenizer`
	`from tokenizers.implementations import ByteLevelBPETokenizer`
	`from transformers import GPT2TokenizerFast`

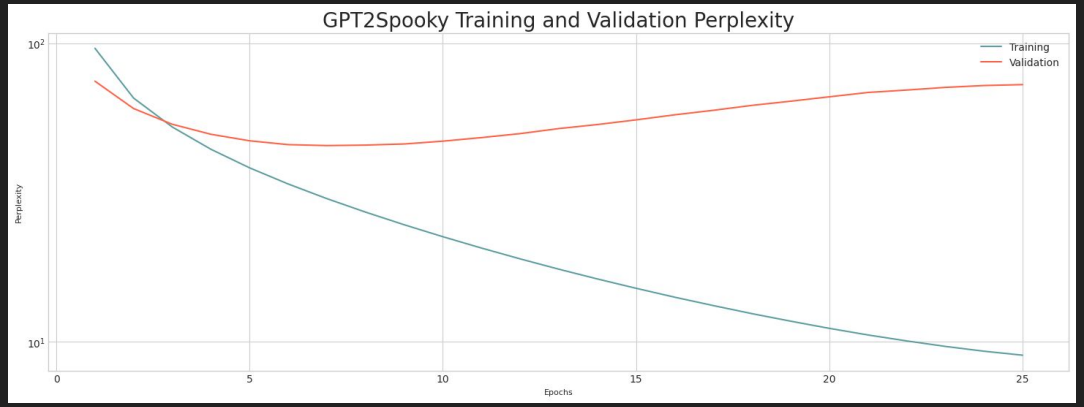
Custom Tokens	`bos_token='<|startoftext|>', eos_token='<|endoftext|>', pad_token='<|pad|>'`
Model Head	`from transformers import GPT2LMHeadModel`
Vocab Size	`8000`
Number of Attention Heads	`12`
Number of Hidden Layers	`6`
Optimizer	`from torch.optim import AdamW`
Number of Epochs	5
Batch Size	64
Max Input Length	512 Tokens
Seed	42

GPT2Spooky Fine Tuning

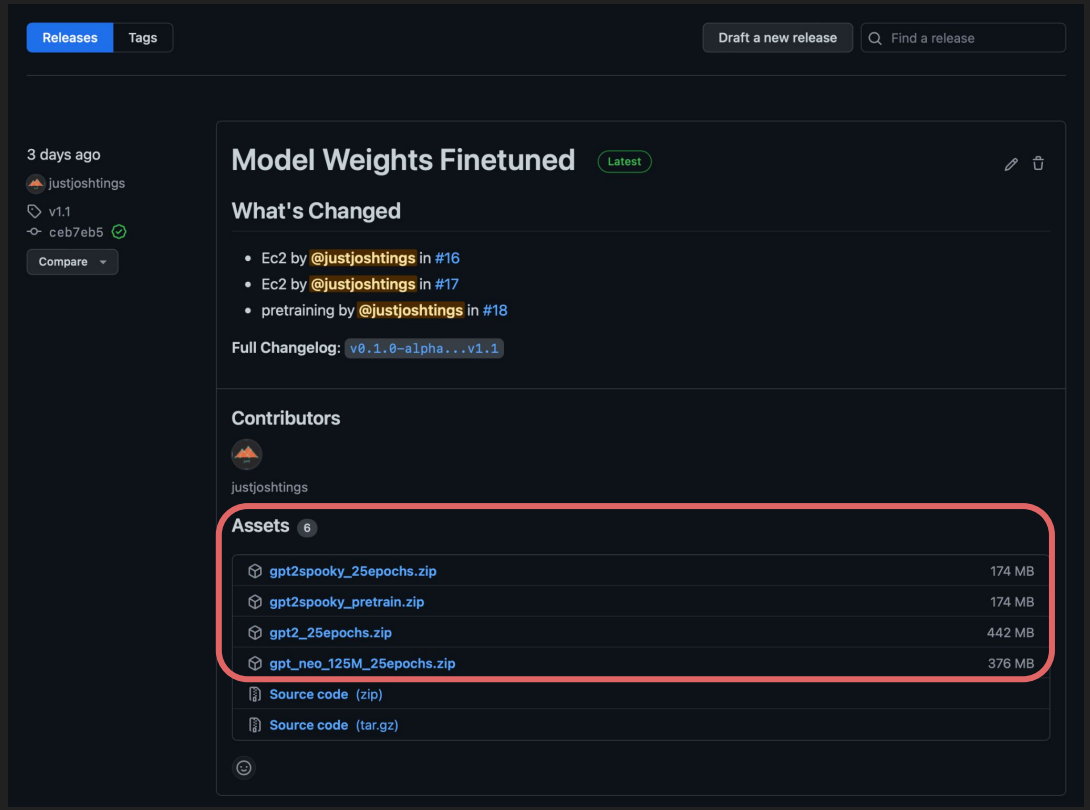
Configuration	Value
Tokenizer	`from transformers import GPT2Tokenizer`
Model Head	`from transformers import GPT2LMHeadModel`
Optimizer	`from torch.optim import AdamW`
Custom Tokens	`bos_token='<|startoftext|>', eos_token='<|endoftext|>', pad_token='<|pad|>'`
Number of Epochs	25
Learning Rate	5e-5
Learning Rate Scheduler	Linear
Batch Size	1
Max Input Length	512 Tokens
Model Type	‘gpt2spooky’
Seed	42
Loss	Cross Entropy
Metric	Perplexity, Equation (3)
Number of Parameters	5M
Pretrained On	Custom Corpus (Scary Stories)

GPT2Spooky Training





Model Distribution



Model Evaluation: Perplexity

After 25 epochs:

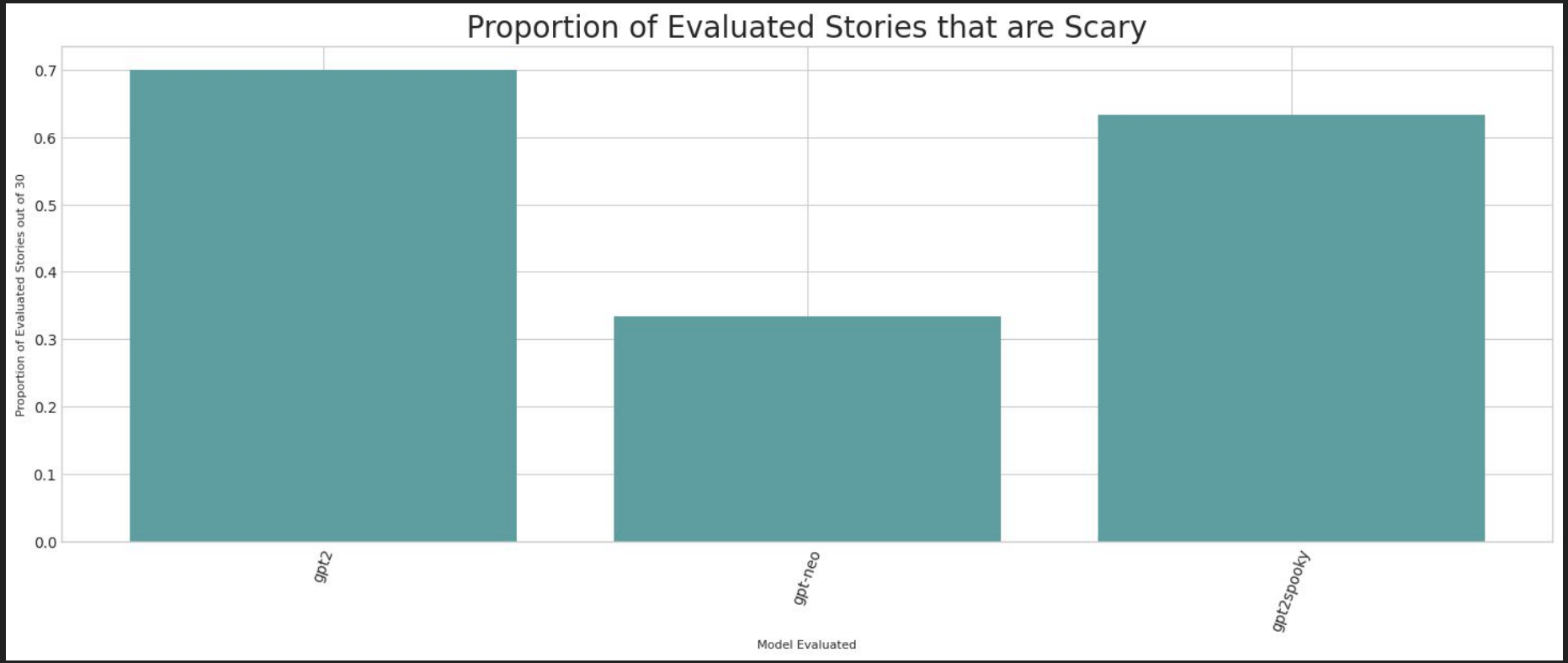
1. GPT2 = 22.5
2. GPT-Neo = 50.7
3. GPT2Spooky = 72.7

Model Evaluation: Human Evaluation

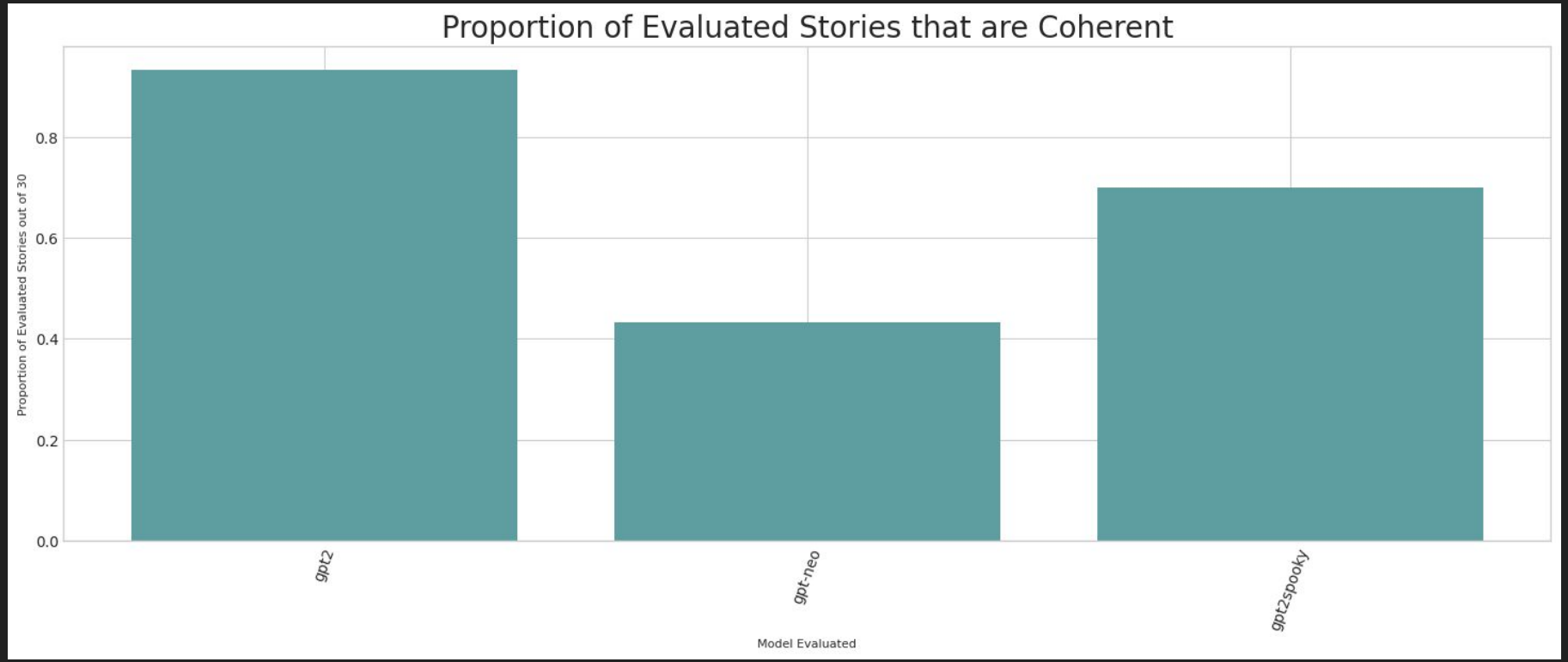
prompts	gpt2_25_generate	gpt_neo_25_generate	gpt2spooky_generate	gpt2_25_scary	gpt_neo_25_scary	gpt2spooky_scary	gpt2_25_choherent	gpt_neo_25_coherent	gpt2spooky_coherent
Lorem ipsum…	Lorem ipsum…	Lorem ipsum…	Lorem ipsum…	1	0	0	0	1	1
Lorem ipsum…	Lorem ipsum…	Lorem ipsum…	Lorem ipsum…	0	1	1	1	0	1

30 prompts total

Model Evaluation: Human Evaluation



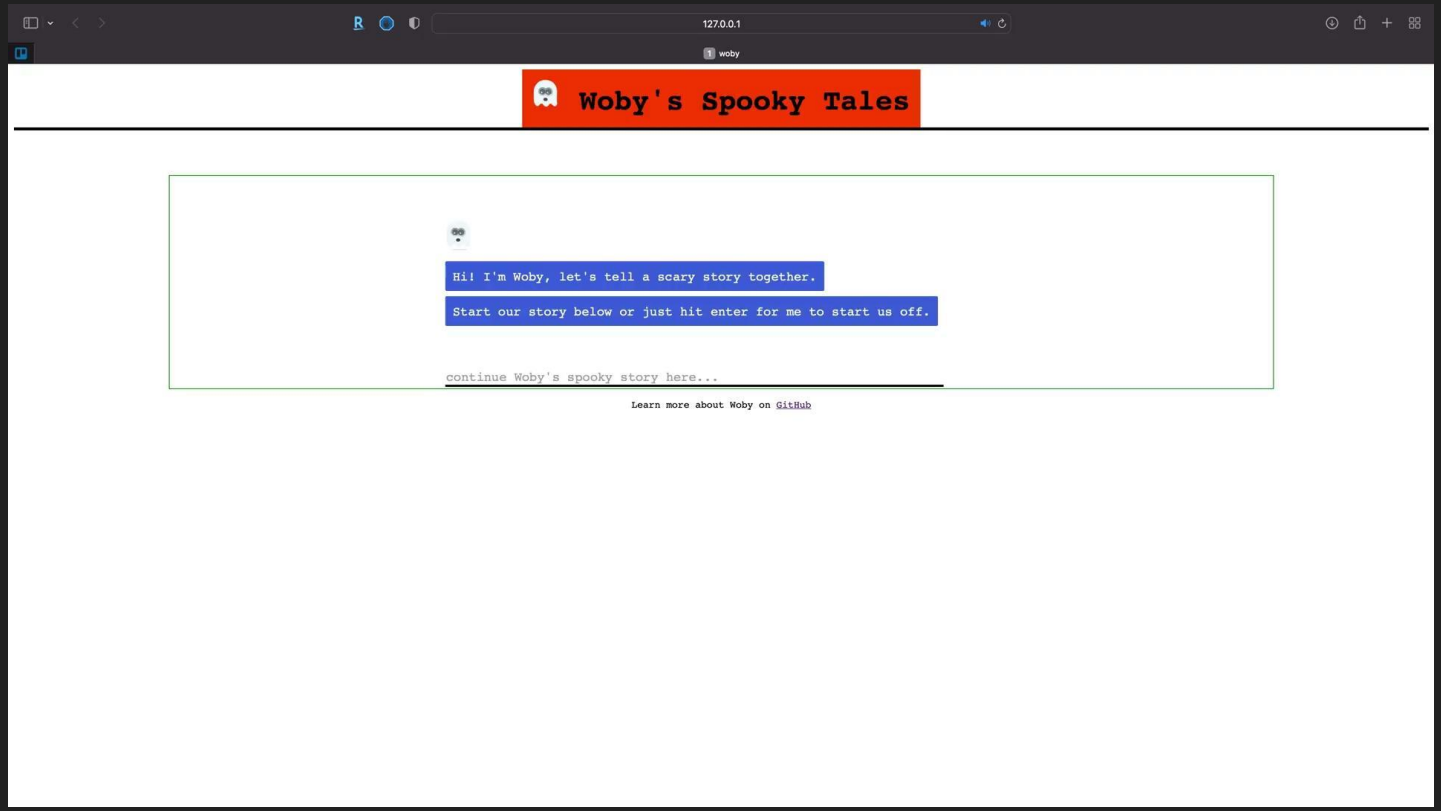
Model Evaluation: Human Evaluation



Conclusion

- Succeeded in our objective to train and compare several models to perform the downstream task of horror story generation
- Best model: GPT2 fine tuned
- Performance of GPT2 and GPT2Spooky are not too far off
- More tweaking and fine tuning, GPT2Spooky may perform close to GPT2 while being much smaller
- Human evaluation is completely subjective and small sample size

Front End Chat Bot - Flask App: Demo



Front End Chat Bot - Flask App: Demo

References

1. Vaswani et. al (2017) Attention is All You Need
2. Glue Benchmark
3. Devlin et. al (2018) BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding
4. Radford et. al (2018) Improving Language Understanding by Generative Pre-Training
5. Shelley: Human-AI Collaborated Horror Stories
6. Python Reddit API Wrapper
7. Python Pushshift.io API Wrapper
8. Hugging Face GPT2
9. Hugging Face GPT-Neo
10. Hugging Face Transformers
11. PyTorch
12. Perplexity in Language Models
13. Hugging Face Transformers Fine Tuning
14. Hugging Face How to Text Generation
15. Hugging Face Text Generation
16. Fine Tune GPT2
17. Flask Chatbot
18. Conditional Text Generation for Harmonious Human-Machine Interaction, Guo et al., 2020
19. Transformers: State-of-the-Art Natural Language Processing, Wolf et al., 2020
20. Transformer-based Conditional Variational Autoencoder for Controllable Story Generation, Fang et al., 2021
21. Improving Neural Story Generation by Targeted Common Sense Grounding, hhmao et al., 2019
22. Evaluation of Text Generation: A Survey, Celikyilmaz et al., 2021