

# **Exploring GPT-2 and It's Implementation**

*Submitted in fulfillment of the `J` component for the Course*

## **NATURAL LANGUAGE PROCESSING CSE4022 SLOT A2**

*by*  
**Saksham Saini**  
**17BCB0094**

**Under the guidance of Prof. Sharmila Banu**

**School of Computer Science and Engineering**  
**VIT, Vellore**



June 7, 2020

<b>Contents</b>	<b>Page No.</b>
<b>1. Problem statement</b>	<b>3</b>
<b>2. Motivation</b>	<b>3</b>
<b>3. Dataset description</b>	<b>4</b>
<b>4. Modules and Introduction</b>	<b>7</b>
4.1 Architecture	9
4.2 Training	10
4.3 Implementation	13
<b>5. List of challenges faced and Snapshots of errors and fixes</b>	<b>16</b>
<b>6. References</b>	<b>17</b>
<b>7. Appendix</b>	<b>18</b>

**Repo link:** <https://github.com/saqsham/GPT-2-Expo>

## 1. Problem statement

- There are many text generation models and apps are available but most of them don't work as intended due to design or some holes in the model
- In the year 2019, OpenAI corporation released a paper and developed an improved version of gpt known as gpt-2, it's basically the code for the paper "[Language Models are Unsupervised Multitask Learners](#)".
- gpt-2 is one of the powerful text generation tool, since it's release many more tools were developed over it as a base, one of them is [Transformer](#) which's a very powerful tool too among people/researchers who practice NLP.
- This is one of the interesting [article](#) which was written before the paper got open to public.

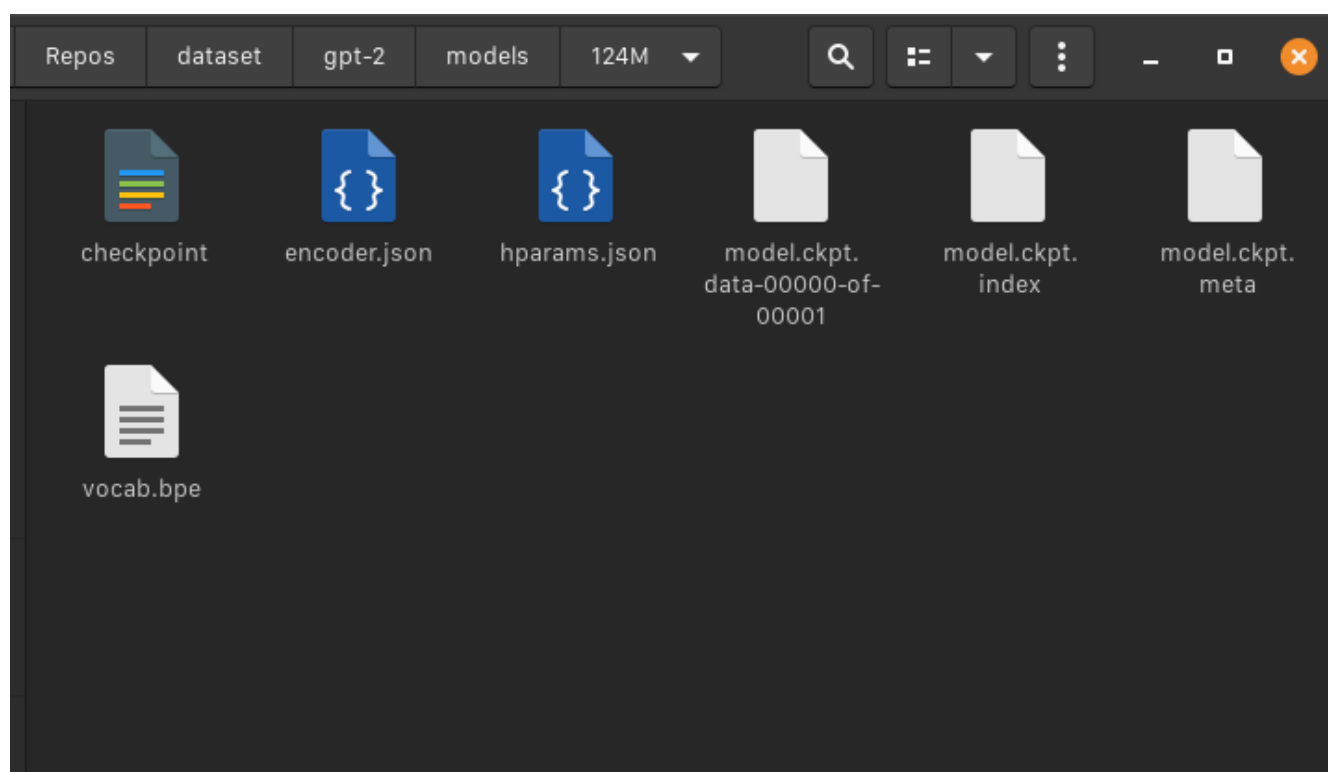
## 2. Motivation

- I found out about gpt-2 through reddit, specifically this [page](#), here I found out that only bots are allowed to chat. I went through the posts and comments, most of them made sense, sentences were grammatically correct and then I started to wonder how was that possible.
- I started looking at the names of the bots and most of them had have this common string in them 'gpt-2'
- Later I searched through the internet and also discussed with my senior, he guided me a bit, and so I thought of doing this as a project.

### 3. Dataset description

Model – 124M (500 MB)

- Hardware restriction: to use model large 774 and largest 1558 it requires more than 4GB GPU GDDR5 Memory



*Fig 1: Contents in Model 124M*

### Dataset – reddit\_titles.csv (1.1 MB)

- The dataset contains 16000 lines of titles taken from reddit

	A	B
2787	DokkanBattleCommunity	Invader Goku
2788	DokkanBattleCommunity	Ignore the low effort, I just thought it was funny
2789	DokkanBattleCommunity	But Sire...
2790	DokkanBattleCommunity	When you get Cell instead of Gohan but don't have good Cell Saga units or a good team for him but you try to make the best of h
2791	DokkanBattleCommunity	Can't wait to see LR Gohans active skill when super star Patrick appears
2792	Doom	I paid Jesus to recite the Slayer's Testament.
2793	Doom	Hmpf hmpf
2794	Doom	The is only one way
2795	Doom	*Gauss Cannon Noises*
2796	Doom	Keeping the hype going
2797	Doom	"Rating pending"
2798	Doom	BUT MOM
2799	Doom	Well, this was funny. Unexpected mod.
2800	Doom	This is hilarious.
2801	Doom	Old Doom Cartoon
2802	Doom	N(o)stalgia(c) post time
2803	Doom	It sets the mood okay?
2804	Doom	S p e e d
2805	Doom	Everyone on November 22
2806	Doom	Mick Gordon changed my entire taste in music
2807	Doom	I-.....uh....what?
2808	Doom	What is this abomination!!!!???
2809	Doom	Anon asks an interesting question.
2810	DotA2	Congratulations to the Champions of The International 2019!
2811	DotA2	Last day best picture lol
2812	DotA2	I won Dota 2 Cosplay Championship at TI9!
2813	DotA2	tims statement on the chinese crowd
2814	DotA2	Please Ban Mason.
2815	DotA2	Gabe, seen sitting on a stone ball in the streets of Shanghai
2816	DotA2	People think Dota2 is some dead niche moba ran by a couple of sweaty pigs in their basement because that's exactly how Valve
2817	DotA2	The Morph Cycle
2818	DotA2	Married an offline OD picker I met in a unranked pub, AMA
2819	DotA2	rtz vs VG
2820	DotA2	QG fanart ti9 1920*1080
2821	DotA2	When a non-chinese team makes a play
2822	DotA2	update 7.23
2823	DotA2	Winter Wyvern's body should go UP when her wings go DOWN and vice versa. But in the game, it is the exact opposite of this ar
2824	DotA2	The last thing QG does before back to home
2825	DotA2	Twitch dota2ti channel - Don't type "tiananmen square" or you'll get banned
2826	DotA2	Nahaz with the correct response to Addgerall "drama"
2827	DotA2	Saw this on twitter. I had to screenshot and share
2828	DragaliaLost	I drew Edelgard, Byleth, and Hubert in Dragalia Life style!
2829	DragaliaLost	Ranzal and Rena having fun together-
2830	DragaliaLost	Qdetta, Summer Incarnate
2831	DragaliaLost	tfw

Fig 2: Dataset contents

## Dataset Generation

Getting titles from reddit using google [BigQuery](#)

- Using BigQuery is limitedly free

```
#standardSQL
WITH
  subreddits AS (
    SELECT
      subreddit
    FROM
      `fh-bigquery.reddit_posts.2019_08`
    WHERE
      score >= 5
      AND subreddit NOT IN ("me_irl",
        "2meirl4meirl",
        "anime_irl",
        "furry_irl",
        "cursedimages")
    GROUP BY
      subreddit
    ORDER BY
      APPROX_COUNT_DISTINCT(author) DESC
    LIMIT
      1000 )

SELECT
  subreddit,
  REGEXP_REPLACE(REGEXP_REPLACE(REGEXP_REPLACE(REGEXP_REPLACE(title, '&', '&'),
    '<', '<'), '>', '>'), '0', '') as title
FROM (
  SELECT
    subreddit,
    title,
    ROW_NUMBER() OVER (PARTITION BY subreddit ORDER BY score DESC) AS score_rank
  FROM
    `fh-bigquery.reddit_posts.2019_08`
  WHERE
    subreddit IN (SELECT subreddit FROM subreddits) )
WHERE
  score_rank <= 18
ORDER BY subreddit
```

## 4. Modules and Introduction

- There are currently 4 models which were released by OpenAI, small being 124M, medium 355M, large 774M and the largest being 1558M.
- GPT-2 models' robustness and worst case behaviors are not well-understood. As with any machine-learned model, carefully evaluate GPT-2 for your use case, especially if used without fine-tuning or in safety-critical applications where reliability is important.
- The dataset GPT-2 models were trained on contains many texts with [biases](#) and factual inaccuracies, and thus GPT-2 models are likely to be biased and inaccurate as well.
- To avoid having samples mistaken as human-written, we recommend clearly labeling samples as synthetic before wide dissemination. The models are often incoherent or inaccurate in subtle ways, which takes more than a quick read for a human to notice.
- The `'keyword_encode.py'` script which attempts to extract the keywords in an unsupervised manner (although you can provide your own keywords if you have them). The methodology is as follows for each text document:
  1. Extract the keywords from each document as "keywords" using spaCy, which both tokenizes keywords and tags their parts-of-speech.
    - \* Only nouns, verbs, adjectives, and adverbs are extracted. Nouns use the raw version of the word (for best user experience when they input them manually) while the other POS use the lemmatized versions (to reduce overfitting but still provide information).
    - \* Proper nouns, named entities, and compound nouns count as their own keyword.
    - \* Pronouns and stop words are excluded from keywords.
    - \* Keywords are deduped.
  2. Prepare the keywords in such a way that the document text is generated conditionally on the keywords.
    - \* Normalize the keywords (replace spaces/punctuation w/ dashes). The keywords are not case-normalized for best user experience when specifying keywords.
    - \* Shuffle the order of the keywords to prevent GPT-2 from cheating and learning when the order of the keywords should be written in the document proper.

\* For each set of processed keywords in a document, create `repeat` random combinations (default: 3) of the keywords. This serves as a data augmentation of sorts, and prevents the model from overfitting on a given set of keywords.

\* For each combination above, select a random number of \*up to\* `max\_keywords` (default: 3), which are then shuffled, to prevent the neural network from a) learning the number of keywords as a hint to the length of the text and b) the order of the keywords in the resulting text.

3. Write the keywords, then the document for each generated set of keywords.

\* The documents are processed in batches with ray; after each batch is encoded, the batch is shuffled before writing to reduce leakage.

- The default case (passing a CSV of `titles`) generates `keywords`, and outputs a `.txt` of keywords and titles.
- The `keyword\_decode.py` script contains functions for decoding bulk-generated encoded texts (e.g. generated through gpt-2-simple, albeit the native truncation is recommended in that use case). `decode\_texts()` will extract the text from each of the specified taxonomic sections for the provided list of texts, and `decode\_file()` can extract and decode all texts and write to a file.
- The encoding is tokenized using spaCy for more robust keyword tokenization and parallelized using ray in order to massively speed up encoding on large datasets.

Packages/Libraries used:

- gpt-2-simple
- spacy>=2.1.0,<2.2.0
- tqdm
- ray
- [https://github.com/explosion/spacy-models/releases/download/en\\_core\\_web\\_sm-2.1.0/en\\_core\\_web\\_sm-2.1.0.tar.gz](https://github.com/explosion/spacy-models/releases/download/en_core_web_sm-2.1.0/en_core_web_sm-2.1.0.tar.gz)
- aiohttp
- psutil
- setproctitle
- grpcio
- tensorflow=1.14.0



## 4.1 Architecture

The actual Transformer architecture GPT-2 uses is very complicated to explain (here's a [great lecture](#)). For the purposes of finetuning, since we can't modify the architecture, it's easier to think of GPT-2 as a black box, taking in inputs and providing outputs. Like [previous forms of text generators](#), the inputs are a sequence of tokens, and the outputs are the probability of the next token in the sequence, with these probabilities serving as weights for the AI to pick the next token in the sequence. In this case, both the input and output tokens are byte pair encodings, which instead of using character tokens (slower to train but includes case/formatting) or word tokens (faster to train but does not include case/formatting) like most RNN approaches, the inputs are "compressed" to the shortest combination of bytes including case/formatting, which serves as a compromise between both approaches but unfortunately adds randomness to the final generation length. The byte pair encodings are later decoded into readable text for human generation.



```

Loading dataset...
100%|████████████████████████████████████████████████████████████████████████████████|
dataset has 294176 tokens
Training...
[10 | 358.92] loss=0.09 avg=0.09
[20 | 703.59] loss=0.09 avg=0.09
[30 | 1076.53] loss=0.09 avg=0.09
[40 | 1451.88] loss=0.06 avg=0.08
[50 | 1832.38] loss=0.07 avg=0.08
^C^Cinterrupted
Saving checkpoint/run1/model-53
(python35-tf) Sun Jun 07 [~/Uni/sem_6/NLP/Project]

```

Fig 4: Training dataset part 2



Fig 5: CPU and RAM usage while training the dataset



Fig 6: Monitor after training was completed

### 4.3 Implementation

Generating titles in a batch or one by one

```
2020-06-07 15:09:44.298767: W tensorflow/compiler/jit/mark_for_compilation_pass.cc:1412] (One-time warning): Not using XLA:CPU for cluster because envvar TF_XLA_FLAGS=--tf_xla_cpu_global_jit was not set. If you want XLA:CPU, either set that envvar, or use experimental_jit_scope to enable XLA:CPU. To confirm that XLA is active, pass --vmodule=xla_compilation_cache=1 (as a proper command-line flag, not via TF_XLA_FLAGS) or set the envvar XLA_FLAGS=--xla_hlo_profile.
Loading checkpoint checkpoint/run1/model-63
WARNING:tensorflow:From /home/saqsham/anaconda3/envs/python35-tf/lib/python3.5/site-packages/tensorflow/python/training/saver.py:1276: checkpoint_exists (from tensorflow.python.training.checkpoint_management) is deprecated and will be removed in a future version.
Instructions for updating:
Use standard file APIs to check for files with this prefix.
WARNING:tensorflow:From /home/saqsham/anaconda3/envs/python35-tf/lib/python3.5/site-packages/gpt_2_simple/src/sample.py:17: add_dispatch_support.<locals>.wrapper (from tensorflow.python.ops.array_ops) is deprecated and will be removed in a future version.
Instructions for updating:
Use tf.where in 2.0, which has the same broadcast rule as np.where
What is your most disturbing/creepy's story about someone you love?

Which piece of advice can you give someone that will literally change the future of their life?
What is the dumbest or most stupid thing that a child has ever said in a public restroom?
Why does your brain think you're a psychopath?
What's something that will go viral in the next 4-6 years that you think will be forgotten?
What's something you learned that you would never forget?
What is something we should all be afraid of and how do we respond?

What's it like to be in a relationship?
(python35-tf) Sun Jun 07 [~/Uni/sem_6/NLP/Project]
```

Fig 7: generating titles or text (kinda emo texts)

Generating using encoding in bulk

Running the script:

The screenshot shows a VS Code editor on the left and a Ray Dashboard on the right. The VS Code editor displays the file explorer with a project named 'train\_posts.py'. The file 'train\_posts.py' is open, showing code that imports 'ray' and 'ray.init' with specific memory settings. The code defines a function 'encode\_keywords' that takes a CSV file path and an output path, and uses 'ray.encode' to process the data. The terminal shows the command 'python train\_posts.py' being executed, and the output indicates that the script is running successfully, starting up 16 workers and encoding the data.

The Ray Dashboard on the right shows the 'MACHINE VIEW' tab. It displays a table of Ray nodes and their resources. The table has columns for Host, Workers, Uptime, CPU, RAM, Disk, Sent, Received, Logs, and Errors. The first row shows the master node 'saqsham-popos' with 23 workers and 8 cores. The subsequent rows show the workers, each with 1 CPU and 16 GB of RAM. The dashboard also shows the progress of the Ray job, with a table of tasks and their status. The tasks are 'Encoder generate\_encoded\_text()' and 'Encoder', and they are all in the 'Running' state.

Fig 8: running train\_encode\_text.py

## Output

reddit\_titles\_encoded.txt



```

data > reddit_titles_encoded.txt
1 <|startoftext|>`destiny2~^#D1toD2 fly~@0h how the time flies... #D1toD2<|endoftext|>
2 <|startoftext|>`destiny2~^#D1toD2 fly~@0h how the time flies... #D1toD2<|endoftext|>
3 <|startoftext|>`jordanpeterson~^Warner-Bros~@Warner Bros get it<|endoftext|>
4 <|startoftext|>`celebs~^Comedian Iliza-Shlesinger~@Comedian Iliza Shlesinger<|endoftext|>
5 <|startoftext|>`nostalgia~^simpsons every-simpsons~@After every simpsons episode<|endoftext|>
6 <|startoftext|>`nostalgia~^simpsons~@After every simpsons episode<|endoftext|>
7 <|startoftext|>`destinythegame~^the-shaders collection~@I screenshotted all of the shaders in
8 <|startoftext|>`baseballcards~^~@This is totally me with baseball lol<|endoftext|>
9 <|startoftext|>`popheads~^~@Rolling stone names shawn mendes and camilla cabello's seniorita pe
10 <|startoftext|>`celebs~^Comedian Iliza-Shlesinger~@Comedian Iliza Shlesinger<|endoftext|>
11 <|startoftext|>`popheads~^performance camilla seniorita~@Rolling stone names shawn mendes and c
12 <|startoftext|>`mexico~^a-su-primo-Big-Bird~@Abelardo visito a su primo Big Bird<|endoftext|>
13 <|startoftext|>`mexico~^~@Abelardo visito a su primo Big Bird<|endoftext|>
14 <|startoftext|>`celebs~^Comedian Iliza-Shlesinger~@Comedian Iliza Shlesinger<|endoftext|>
15 <|startoftext|>`germanshepherds~^Day Alright~@The Day I Knew He Was Alright<|endoftext|>
16 <|startoftext|>`cats~^my-soul straight soul~@My little Clementine always looks like she's star
17 <|startoftext|>`fifacareers~^sorry little~@I'm sorry little one<|endoftext|>
18 <|startoftext|>`completeanarchy~^a-commune capitalism let~@Fuck capitalism, let's start a comm
19 <|startoftext|>`popheads~^~@Rolling stone names shawn mendes and camilla cabello's seniorita pe
20 <|startoftext|>`baseballcards~^lol baseball-lol~@This is totally me with baseball lol<|endoftext|>
21 <|startoftext|>`tiktokcringe~^sound come this-sound~@that's how this sound really comes out.<|
22 <|startoftext|>`nostalgia~^Netflix The-Wii~@Netflix's On The Wii<|endoftext|>
23 <|startoftext|>`germanshepherds~^Day Alright~@The Day I Knew He Was Alright<|endoftext|>
24 <|startoftext|>`destinythegame~^collection sets~@I screenshotted all of the shaders in my coll
25 <|startoftext|>`mexico~^Abelardo primo Big-Bird~@Abelardo visito a su primo Big Bird<|endoftext|>
26 <|startoftext|>`completeanarchy~^~@Fuck capitalism, let's start a commune<|endoftext|>
27 <|startoftext|>`tiktokcringe~^sound~@that's how this sound really comes out.<|endoftext|>
28 <|startoftext|>`nostalgia~^simpsons episode every-simpsons~@After every simpsons episode<|endo
29 <|startoftext|>`mexico~^amigos almorzaron~@Ya almorzaron amigos?<|endoftext|>
30 <|startoftext|>`destinythegame~^sets~@I screenshotted all of the shaders in my collection on a
31 <|startoftext|>`cats~^look~@My little Clementine always looks like she's staring straight into
32 <|startoftext|>`completeanarchy~^fuck~@Fuck capitalism, let's start a commune<|endoftext|>
33 <|startoftext|>`sonicthehedgehog~^Tracy-Yardley draw~@Something Tracy Yardley was recently com
34 <|startoftext|>`nostalgia~^Netflix The-Wii~@Netflix's On The Wii<|endoftext|>
35 <|startoftext|>`baseballcards~^lol baseball-lol~@This is totally me with baseball lol<|endoftext|>
36 <|startoftext|>`sonicthehedgehog~^Tracy-Yardley draw~@Something Tracy Yardley was recently com
37 <|startoftext|>`fifacareers~^sorry little~@I'm sorry little one<|endoftext|>
38 <|startoftext|>`nostalgia~^Netflix The-Wii~@Netflix's On The Wii<|endoftext|>
39 <|startoftext|>`tiktokcringe~^sound this-sound~@that's how this sound really comes out.<|endof
40 <|startoftext|>`destiny2~^~@0h how the time flies... #D1toD2<|endoftext|>
41 <|startoftext|>`cats~^~@My little Clementine always looks like she's staring straight into my
42 <|startoftext|>`jordanpeterson~^~@Warner Bros get it<|endoftext|>
43 <|startoftext|>`mexico~^amigos almorzaron~@Ya almorzaron amigos?<|endoftext|>
44 <|startoftext|>`fifacareers~^sorry little~@I'm sorry little one<|endoftext|>
45 <|startoftext|>`germanshepherds~^Day Alright~@The Day I Knew He Was Alright<|endoftext|>
46 <|startoftext|>`jordanpeterson~^Warner-Bros~@Warner Bros get it<|endoftext|>
47 <|startoftext|>`mexico~^amigos almorzaron~@Ya almorzaron amigos?<|endoftext|>
48 <|startoftext|>`sonicthehedgehog~^~@Something Tracy Yardley was recently commissioned to draw.
49 <|startoftext|>`self~^pace Eye-of-the-Tiger~@A SUV pace me this morning while I was running. U
50 <|startoftext|>`designerreps~^Gucci-Snake-Ring~@[W2C] Gucci Snake Ring<|endoftext|>
51 <|startoftext|>`corgi~^anyone an-extra-corgwich~@i got an extra corgwich, if anyone's hungry<|
52 <|startoftext|>`books~^ago~@Exactly one year ago I took up reading, today I finished my 250th
53 <|startoftext|>`books~^exactly 250th my-favorites~@Exactly one year ago I took up reading, tod

```

Fig 9: reddit\_titles\_encoded.txt file having the generated text



Decoding the .txt file

After decoding the reddit\_titles\_encoded.txt file

```

data > reddit_titles_decoded.txt
1 Oh how the time flies... #D1toD2
2 =====
3 Oh how the time flies... #D1toD2
4 =====
5 Warner Bros get it
6 =====
7 Comedian Iliza Shlesinger
8 =====
9 After every simpsons episode
10 =====
11 After every simpsons episode
12 =====
13 I screenshotted all of the shaders in my collection on all three Majestic Solstice sets so you
14 =====
15 This is totally me with baseball lol
16 =====
17 Rolling stone names shawn mendes and camilla cabello's seniorita performance as the worst perform
18 =====
19 Comedian Iliza Shlesinger
20 =====
21 Rolling stone names shawn mendes and camilla cabello's seniorita performance as the worst perform
22 =====
23 Abelardo visito a su primo Big Bird
24 =====
25 Abelardo visito a su primo Big Bird
26 =====
27 Comedian Iliza Shlesinger
28 =====
29 The Day I Knew He Was Alright
30 =====
31 My little Clementine always looks like she's staring straight into my soul
32 =====
33 I'm sorry little one
34 =====
35 Fuck capitalism, let's start a commune
36 =====
37 Rolling stone names shawn mendes and camilla cabello's seniorita performance as the worst perform
38 =====
39 This is totally me with baseball lol
40 =====
41 that's how this sound really comes out.
42 =====
43 Netflix's On The Wii
44 =====
45 The Day I Knew He Was Alright
46 =====
47 I screenshotted all of the shaders in my collection on all three Majestic Solstice sets so you
48 =====
49 Abelardo visito a su primo Big Bird
50 =====
51 Fuck capitalism, let's start a commune
52 =====
53 that's how this sound really comes out.

```

Fig 10: decode texts from the encoded txt file

## 5. List of challenges faced and Ssnapshots of errors and fixes

- Initially I used the latest version of tensorflow, since it was a better version. Later I read the errors and searched on the internet about the errors that were showing up and hence later I found out that I supposed to use tensorflow 1.14 or 1.15, till today gpt-2-simple doesn't support tensorflow 2
- Second was more of a restriction on time and to make the data availability easier, to create my own dataset if I was using traditional method of scraping it would have took a consoderable amount of time, then I found about Google BigQuery and hence used it in my project.
- Third error was to make the dataset sfw and kinda socially acceptable, when first time I ran the text generation script my “bot” was using all kinds of not-so-socially-acceptable words, so then I tried to minimize those by removing the ‘bad’ subreddits and was continously testing it on the BigQuery output.

2/3/20		
11:21 AM	✓	#standardSQL WITH subreddits AS ( SELECT subreddit FROM `fh-bigquery.reddit_posts.2019_08` WHERE score >... <a href="#">↕</a>
11:20 AM	✓	#standardSQL WITH subreddits AS ( SELECT subreddit FROM `fh-bigquery.reddit_posts.2019_08` WHERE score >... <a href="#">↕</a>
11:02 AM	✓	#standardSQL WITH subreddits AS ( SELECT subreddit FROM `fh-bigquery.reddit_posts.2019_08` WHERE score >... <a href="#">↕</a>
10:08 AM	✓	#standardSQL WITH subreddits AS ( SELECT subreddit FROM `fh-bigquery.reddit_posts.2019_01` WHERE score >... <a href="#">↕</a>
10:03 AM	✓	#standardSQL WITH subreddits AS ( SELECT subreddit FROM `fh-bigquery.reddit_posts.2019_01` WHERE score >... <a href="#">↕</a>
9:59 AM	✓	#standardSQL WITH subreddits AS ( SELECT subreddit FROM `fh-bigquery.reddit_posts.2019_01` WHERE score >... <a href="#">↕</a>
9:55 AM	✓	#standardSQL WITH subreddits AS ( SELECT subreddit FROM `fh-bigquery.reddit_posts.2019_01` WHERE score >... <a href="#">↕</a>
9:54 AM	✓	#standardSQL WITH subreddits AS ( SELECT subreddit FROM `fh-bigquery.reddit_posts.2019_02` WHERE score >... <a href="#">↕</a>
9:53 AM	✓	#standardSQL WITH subreddits AS ( SELECT subreddit FROM `fh-bigquery.reddit_posts.2019_04` WHERE score >... <a href="#">↕</a>
9:52 AM	!	#standardSQL WITH subreddits AS ( SELECT subreddit FROM `fh-bigquery.reddit_posts.2019_04` WHERE score >... <a href="#">↕</a>
9:47 AM	✓	#standardSQL WITH subreddits AS ( SELECT subreddit FROM `fh-bigquery.reddit_posts.2019_02` WHERE score >... <a href="#">↕</a>
9:44 AM	✓	#standardSQL WITH subreddits AS ( SELECT subreddit FROM `fh-bigquery.reddit_posts.2019_02` WHERE score >... <a href="#">↕</a>
9:42 AM	✓	#standardSQL WITH subreddits AS ( SELECT subreddit FROM `fh-bigquery.reddit_posts.2019_02` WHERE score >... <a href="#">↕</a>
9:39 AM	✓	#standardSQL WITH subreddits AS ( SELECT subreddit FROM `fh-bigquery.reddit_posts.2019_02` WHERE score >... <a href="#">↕</a>
9:33 AM	✓	#standardSQL WITH subreddits AS ( SELECT subreddit FROM `fh-bigquery.reddit_posts.2019_04` WHERE score >... <a href="#">↕</a>
9:31 AM	✓	#standardSQL WITH subreddits AS ( SELECT subreddit FROM `fh-bigquery.reddit_posts.2019_04` WHERE score >... <a href="#">↕</a>
2/2/20		
11:56 AM	✓	#standardSQL SELECT REGEXP_REPLACE(REGEXP_REPLACE(REGEXP_REPLACE(REGEXP_REPLACE(title, '&am... <a href="#">↕</a>
11:55 AM	✓	#standardSQL SELECT REGEXP_REPLACE(REGEXP_REPLACE(REGEXP_REPLACE(REGEXP_REPLACE(title, '&am... <a href="#">↕</a>
11:54 AM	✓	#standardSQL SELECT REGEXP_REPLACE(REGEXP_REPLACE(REGEXP_REPLACE(REGEXP_REPLACE(title, '&am... <a href="#">↕</a>

Fig 11: Queries executed on BiqQuery Google Cloud Platform



## 7. References

- [1]. <https://d4mucfpksyvv.cloudfront.net/better-language-models/language-models.pdf>
- [2]. <https://github.com/openai/gpt-2>
- [3]. <https://github.com/minimaxir/gpt-2-simple>
- [4]. <https://minimaxir.com/2019/09/howto-gpt2/>
- [5]. <http://www.peterbloem.nl/blog/transformers>
- [6]. <https://github.com/google/sentencepiece>
- [7]. <https://github.com/openai/gpt-2/issues/114>
- [8]. <https://github.com/soaxelbrooke/python-bpe>
- [9]. [https://huggingface.co/transformers/model\\_doc/gpt2.html](https://huggingface.co/transformers/model_doc/gpt2.html)
- [10]. <https://colab.research.google.com/drive/1RugXCyDcMvSACYNt9j0kB6zzqRKzAbBn#scrollTo=sUmTooTW3osf>
- [11]. [https://colab.research.google.com/drive/1VLG8e7YSEwypxU-noRNhsv5dW4NfTGce#scrollTo=H7LoMj4GA4n\\_](https://colab.research.google.com/drive/1VLG8e7YSEwypxU-noRNhsv5dW4NfTGce#scrollTo=H7LoMj4GA4n_)
- [12]. [https://colab.research.google.com/drive/1qxcQ2A1nNjFudAGN\\_mcMOnvV9sF\\_PkEb#scrollTo=KBkpRgBCBS2\\_](https://colab.research.google.com/drive/1qxcQ2A1nNjFudAGN_mcMOnvV9sF_PkEb#scrollTo=KBkpRgBCBS2_)
- [13]. <https://console.cloud.google.com/bigquery>
- [14]. [https://www.reddit.com/r/legaladviceofftopic/comments/bxi869/i\\_trained\\_an\\_ai\\_to\\_generate\\_the\\_ultimate/](https://www.reddit.com/r/legaladviceofftopic/comments/bxi869/i_trained_an_ai_to_generate_the_ultimate/)
- [15]. <https://minimaxir.com/apps/gpt2-reddit/>
- [16]. <https://github.com/minimaxir/reddit-gpt-2-cloud-run>
- [17]. <https://github.com/minimaxir/gpt-2-keyword-generation>
- [18]. <https://www.reddit.com/r/SubSimulatorGPT2/>
- [19]. [https://www.reddit.com/r/SubSimulatorGPT2/comments/btfhks/what\\_is\\_rsubsimulatorgpt2/](https://www.reddit.com/r/SubSimulatorGPT2/comments/btfhks/what_is_rsubsimulatorgpt2/)
- [20]. <https://huggingface.co/gpt2>
- [21]. <https://huggingface.co/>
- [22]. <https://huggingface.co/models>
- [23]. <https://github.com/huggingface/transformers>
- [24]. <https://openai.com/blog/better-language-models/>
- [25]. <https://github.com/minimaxir/textgenrnn>

## 8. Appendix

### Directory structure

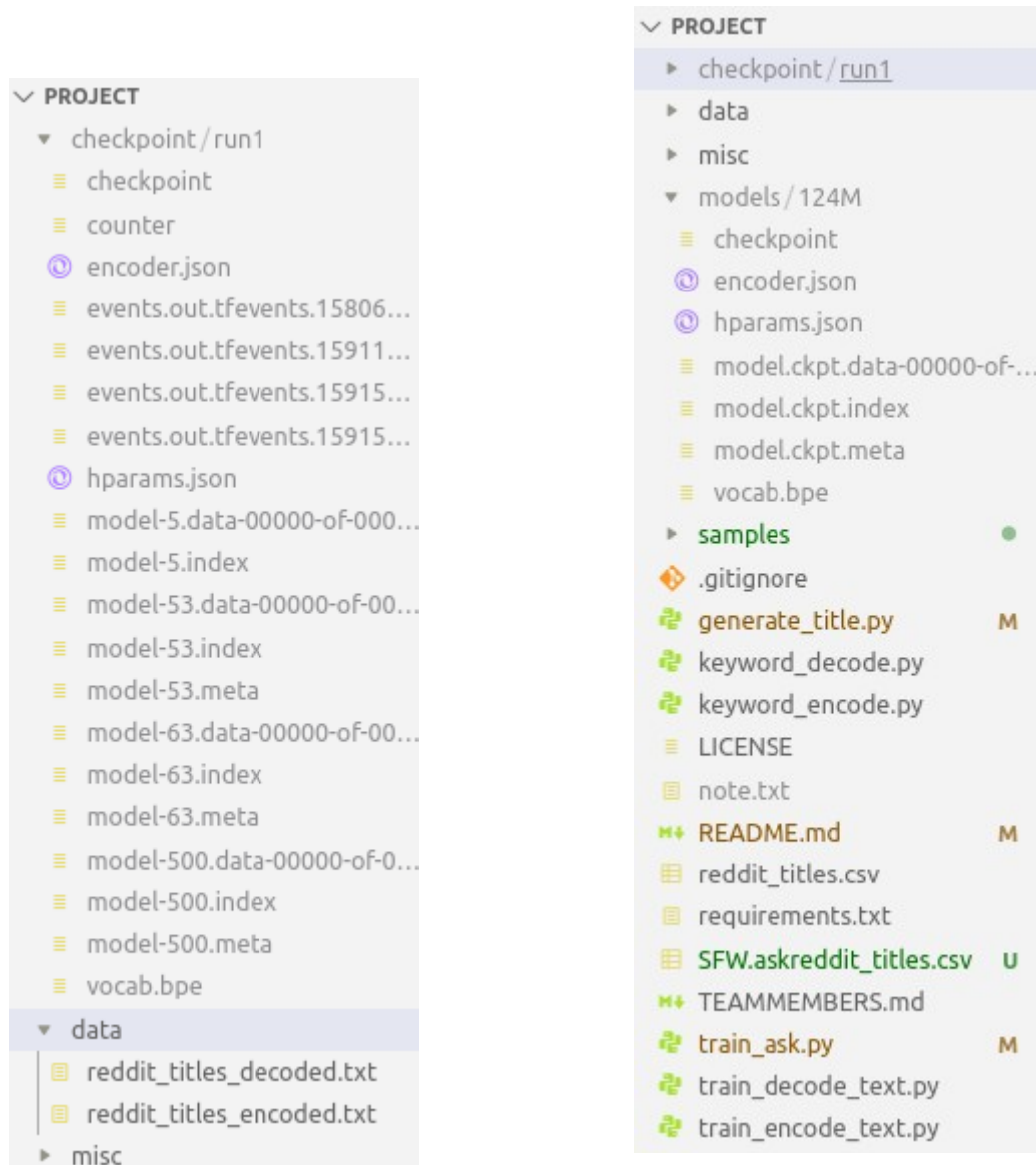


Fig 12: Directory Structure

**train\_ask.py**

```

import gpt_2_simple as gpt2
import os
import requests

# downloads the specified model if not available
# Models:- 124M 355M 774M 1558M
model_name = "124M"
if not os.path.isdir(os.path.join("models", model_name)):
    print("Downloading {model_name} model...")
    gpt2.download_gpt2(model_name=model_name) # model is saved into current
    directory under /models/124M/

file_name = "SFW.askreddit_titles.csv"
if not os.path.isfile(file_name):
    print("dataset not available")

sess = gpt2.start_tf_sess()
gpt2.finetune(sess,
               dataset=file_name,
               model_name='124M',
               steps=500,
               restore_from='fresh',
               run_name='run1',
               print_every=10,
               sample_every=50
               )

```

**generate\_title.py**

```

import gpt_2_simple as gpt2

sess = gpt2.start_tf_sess()
gpt2.load_gpt2(sess, run_name='run1')

gpt2.generate(sess,
               run_name='run1',
               length=100,
               temperature=0.7,
               top_k=40,
               nsamples=10,
               batch_size=10,
               prefix="<|startoftext|>",
               truncate="<|endoftext|>",
               include_prefix=False,
               sample_delim=''
               )

```

**train\_encode\_text.py**

```

from keyword_encode import encode_keywords
import ray

ray.init(object_store_memory=100 * 1000000,
         redis_max_memory=100 * 1000000)

# generate keyword in bulk, default limit is 48000
encode_keywords(csv_path='reddit_titles.csv',
               out_path='data/reddit_titles_encoded.txt',
               category_field='subreddit',
               title_field='title',
               keyword_gen='title')

```

**train\_decode\_text.py**

```

from keyword_decode import decode_file

decode_file(file_path='data/reddit_titles_encoded.txt',
            out_file='data/reddit_titles_decoded.txt',
            sections=['title'],
            start_token="<|startoftext|>",
            end_token="<|endoftext|>")

```

**keyword\_encode.py**

```

import spacy
import csv
import re
import ray
import multiprocessing
from functools import partial
from tqdm import tqdm
from itertools import chain
from random import random, shuffle, randint

DELIMS = {
    'section': '~',
    'category': '\',
    'keywords': '^',
    'title': '@',
    'body': '}'
}

PRONOUN_LIST = ['I', 'Me', 'We', 'You', 'He', 'She',
                'It', 'Him', 'Her', 'Them', 'They']

PRONOUNS = set(PRONOUN_LIST + [x.lower() for x in PRONOUN_LIST])

```

```

def encode_keywords(csv_path, model='en_core_web_sm',
                    category_field=None,
                    keywords_field=None,
                    title_field=None,
                    body_field=None,
                    keyword_gen='title',
                    keyword_sep=', ',
                    dropout=0.5,
                    repeat=3,
                    max_keywords=3,
                    keyword_length_max=20,
                    out_path='csv_encoded.txt',
                    start_token="<|startoftext|>",
                    end_token="<|endoftext|>"):

    data_list = []

    with open(csv_path, 'r', encoding='utf8', errors='ignore') as f:
        reader = csv.DictReader(f)
        for row in reader:
            data_list.append(row)

    shuffle(data_list)

    # https://stackoverflow.com/a/434328
    def chunker(seq, size):
        return (seq[pos:pos + size] for pos in range(0, len(seq), size))

    num_threads = multiprocessing.cpu_count() * 2 # colocate 2 processes per
thread
    print("Starting up {} Workers".format(num_threads))
    encoders = [Encoder.remote(model, category_field,
                                keywords_field,
                                title_field,
                                body_field,
                                keyword_gen,
                                keyword_sep,
                                repeat,
                                max_keywords,
                                keyword_length_max,
                                start_token,
                                end_token,
                                DELIMS,
                                PRONOUNS) for _ in range(num_threads)]

    with open(out_path, 'w', encoding='utf8', errors='ignore') as w:
        pbar = tqdm(total=len(data_list), smoothing=0)
        for chunk in chunker(data_list, num_threads):
            results = ray.get([c.generate_encoded_text.remote(row)
                               for c, row in list(zip(encoders, chunk))])

            # unnest and randomize results
            results = list(chain.from_iterable(results))
            shuffle(results)
            for result in results:
                w.write(result)

```

```

        pbar.update(num_threads)
    pbar.close()

@ray.remote(num_cpus=0.5)
class Encoder(object):
    def __init__(self, model, category_field,
                 keywords_field,
                 title_field,
                 body_field,
                 keyword_gen,
                 keyword_sep,
                 repeat,
                 max_keywords,
                 keyword_length_max,
                 start_token,
                 end_token,
                 DELIMS,
                 PRONOUNS):
        self.nlp = spacy.load(model)
        self.pattern = re.compile('\W+')

        self.category_field = category_field
        self.keywords_field = keywords_field
        self.title_field = title_field
        self.body_field = body_field
        self.keyword_gen = keyword_gen
        self.keyword_sep = keyword_sep
        self.repeat = repeat
        self.max_keywords = max_keywords
        self.keyword_length_max = keyword_length_max
        self.start_token = start_token
        self.end_token = end_token
        self.DELIMS = DELIMS
        self.PRONOUNS = PRONOUNS

    def build_section(self, section, text):
        if text is None:
            return ''
        return self.DELIMS[section] + self.DELIMS[section] + text

    def generate_encoded_text(self, row):
        nlp = self.nlp
        pattern = self.pattern

        # category should be normalized to account for user input
        category = re.sub(
            pattern, '-', row[self.category_field].lower().strip()) if
self.category_field is not None else None

        title = row[self.title_field] if self.title_field is not None else None
        body = row[self.body_field] if self.body_field is not None else None

        if self.keywords_field is None:
            # Generate the keywords using spacy

```

```

# replace smart quotes first for better tokenization
text = re.sub(u'[\u2018\u2019]', '"',
              (re.sub(u'[\u201c\u201d]', "'", row[self.keyword_gen])))
doc = nlp(text)
keywords_pos = [chunk.text if chunk.pos_ == 'NOUN'
                else chunk.lemma_ if chunk.pos_ in ['VERB', 'ADJ',
'ADV']]
                else 'I'
                for chunk in doc
                if not chunk.is_stop
                ]
keywords_ents = [re.sub(' ', '-', chunk.text)
                 for chunk in doc.ents]
keywords_compounds = [re.sub(' ', '-', chunk.text)
                      for chunk in doc.noun_chunks
                      if len(chunk.text) < self.keyword_length_max]

keywords = list(set(keywords_pos +
                    keywords_ents +
                    keywords_compounds) - self.PRONOUNS) # dedupe
else:
    keywords = [keyword.strip()
                for keyword in
row[self.keywords_field].split(self.keyword_sep)]
    keywords = list(set(keywords))

encoded_texts = []
for _ in range(self.repeat):
    new_keywords = keywords
    shuffle(new_keywords)
    new_keywords = " ".join(
        new_keywords[:randint(0, self.max_keywords)])

    encoded_texts.append(self.start_token +
                        self.build_section('category', category) +
                        self.build_section('keywords', new_keywords) +
                        self.build_section('title', title) +
                        self.build_section('body', body) +
                        self.end_token + "\n")

return encoded_texts

```

### keyword\_decode.py

```

import re

DELIMS = {
    'section': '~',
    'category': '\',
    'keywords': '^',
    'title': '@',
    'body': '}'
}

```

```

def build_pattern(sections, start_token, end_token):
    # sections may not be in the correct order: fix it
    key_order = ['category', 'keywords', 'title', 'body']
    sections = [section for section in key_order if section in sections]

    pattern_text = re.escape(start_token) + '(?:.*)'

    for section in sections:
        pattern_text += '(?:{}})'.format(
            re.escape(DELIMS[section] + DELIMS[section])) + '(.*)'

    pattern_text += '(?:.*)' + re.escape(end_token)

    return re.compile(pattern_text, flags=re.MULTILINE)

def decode_texts(texts, sections=['title'],
                 start_token="<|startoftext|>",
                 end_token="<|endoftext|>"):

    # get the index of the group(s) we want to extract
    group_indices = [i + 1 for i, section in enumerate(sections)]

    assert len(group_indices) > 0
    pattern = build_pattern(sections, start_token, end_token)
    if not isinstance(texts, (list,)):
        texts = [texts]
    decoded_texts = []
    for text in texts:
        decoded_text = re.match(pattern, text)
        if decoded_text is None:
            continue
        decoded_text_attrs = tuple(decoded_text.group(i)
                                   for i in group_indices)
        if len(group_indices) == 1:
            decoded_text_attrs = decoded_text_attrs[0]
        decoded_texts.append(decoded_text_attrs)
    return decoded_texts

def decode_file(file_path, out_file='texts_decoded.txt',
                doc_delim=' ' * 20 + '\n',
                sections=['title'],
                start_token="<|startoftext|>",
                end_token="<|endoftext|>"):

    assert len(sections) == 1, "This function only supports output of a single
    section for now."
    doc_pattern = re.compile(re.escape(start_token) +
                             '(.*)' + re.escape(end_token), flags=re.MULTILINE)

    with open(file_path, 'r', encoding='utf8', errors='ignore') as f:
        # warning: loads entire file into memory!
        docs = re.findall(doc_pattern, f.read())

    docs = [start_token + doc + end_token for doc in docs]
    decoded_docs = decode_texts(docs,

```



```
        sections=sections,  
        start_token=start_token,  
        end_token=end_token)  
with open(out_file, 'w', encoding='utf8', errors='ignore') as f:  
    for doc in decoded_docs:  
        f.write("{}\n{}".format(doc, doc_delim))
```