Your Own Personal GPT Report

T-725-MALV, Màlvinnsla - Natural Language Processing

Reykjavik University - School of Computer Science, Menntavegi 1, IS-101 Reykjavík, Iceland

Samuele Pirani

samuele25@ru.is

05. October 2025

Contents

1 Own Personal GPT **3**
1.1 Chosen Selection 3
1.2 Default Model 4
1.3 First Model 6
1.4 Second Model 7
1.5 Third Model 8
1.6 Analysis & comparisons 10

1 Own Personal GPT

The purpose of this document is to outline all the steps required to accomplish the second part of Assignment 2, which involves developing three **small character-level GPT** models using *Karpathy's nanoGPT* repository, trained on a selected corpus. The first section will describe the chosen corpus in detail, providing a clear explanation of its structure and the rationale behind its selection. The following sections present each model developed, starting from the hyperparameter settings and the output generated. Finally, the last section analyzes all the results obtained and the work done behind the scene.

1.1 Chosen Selection

For the purpose of this training experiment, a curated selection of short stories by *"H.P. Lovecraft"*[1][1] has been chosen as the corpus. These stories are known for their distinctive literary style, characterized by archaic vocabulary, elaborate sentence structures, and recurring themes of cosmic horror and existential dread[2]. Originally comprising a broader collection of texts, the dataset was refined to include only narrative prose classified as short fiction, resulting in a corpus of approximately 1.2 million words. The corpus was sourced from a publicly available dataset and distributed in plain text format. Each entry is corresponds to a part of the story (embodied in the *text* cell) , each identified by the source file containing the aforementioned part (embodied in the *meta* cell). Each story can be seen as a concatenation of multiple entries which share the same source file, splitting the overall structure in more tiny pieces, as it possible to see in the Table 1.

This corpus was selected with the intention of enabling the model to learn stylistic and thematic patterns specific to Lovecraftian fiction. The primary objective is to **generate coherent and genre-faithful prose** that emulates the linguistic cadence and atmospheric tension typical of Lovecraft's work.

text	**source**
It is all in that ancestral diary I found; all the hushed innuendoes and furtive tales of things with a blemished eye seen at windows in the night or in deserted meadows near the woods.	{ "source": "unnamable.txt" }
Something had caught my ancestor on a dark valley road, leaving him with marks of horns on his chest and of apelike claws on his back; and when they looked for prints in the trampled dust they found the mixed marks of split hooves and vaguely anthropoid paws.	{ "source": "unnamable.txt" }
Once a post-rider said he saw an old man chasing and calling to a frightful loping, nameless thing on Meadow Hill in the thinly moonlit hours before dawn, and many believed him...	{ "source": "unnamable.txt" }

Table 1: Partial reppresentation of "*Unnamable*" story in the Lovecraft dataset.

[1]https://huggingface.co/datasets/TristanBehrens/lovecraftcorpus

1.2 Default Model

As a starting point for the three subsequent models, a first default model was trained using the standard hyperparameter settings provided in Karpathy repository [2] (as shown in Table 1. The motivation behind this choice lies in understanding the primal generative capabilities of the model over this specific dataset, tuning the settings based on the output produced by each model.

The experimental setup consists of four steps, which remain consistent across all the models:

- **Import and Filtering**: This phase involved importing the dataset into the project and applying a filtering function to reduce its size. The reduction was performed by removing all non-short stories from the original corpus, which contains the entirety of Lovecraft's literary works (as it shown in Listing 1).

```
def filter_short_stories(dataset):
    filtered = [sample for sample in dataset["train"] if any(title in
        sample["text"] for title in short_story_titles)]
    with open("nanoGPT/data/shakespeare_char/lovecraft.txt", "w",
        encoding='utf-8') as f:
         for story in filtered:
             f.write(story["text"] + "\n\n")

def download_corpus():
    if not os.path.exists("nanoGPT/data/shakespeare_char/
         \item lovecraft.txt"):
             ds = load_dataset("TristanBehrens/lovecraftcorpus")
             filter_short_stories(ds)
```

Listing 1: Default implementation of import and filter functions

- **Dataset Prepare**: This practice involves preparing the raw dateset for model training. The dataset is passed as input to `prepare.py` file, which tokenizes the corpus and generates a vocabulary for the subsequent stages. Since the code is already provided by the repository, it is sufficient to execute the script using the following command.

```
subprocess.run('python nanoGPT/data/shakespeare_char/prepare.py', shell=
    True)
```

Listing 2: Example of prepare.py invocation, through the *subprocess* library

- **Training Phase**: This phase initiates the training of the GPT model using the provided dataset and configuration. By adjusting the network's hyperparameters such as *layer count*, *embedding size*, *block size*, and *dropout rate*, it is possible to train models with different architectures and behaviors. The command shown in the section below represents the default implementation used for baseline experimentation To facilitate understanding of the network's hyperparameters, the Table 2 provides a summary of the default settings used in this configuration.

```
    subprocess.run('python nanoGPT/train.py nanoGPT/config/
        train_shakespeare_char.py --device=cpu --compile=False --
        eval_iters=20 --log_interval=1 --block_size=64 --batch_size=12 --
        n_layer=4 --n_head=4 --n_embd=128 --max_iters=2000 --
        lr_decay_iters=2000 --dropout=0.0',shell=True)
```

Listing 3: Command used to train the model in default experiment

[2] https://github.com/karpathy/nanoGPT

Hyperparameter	Value
eval_iters	20
log_interval	1
block_size	64
batch_size	12
n_layer	4
n_head	4
n_embd	128
max_iters	2000
lr_decay_iters	2000
dropout	0.0

Table 2: Default hyperparameter settings

- **Sample Generation**: The model applies the knowledge acquired during the training phase to generate Lovecraft-style text sample. As in the previous steps, the sampling command is already provided in the repository; therefore, it has been included in the Listing 4.

```
subprocess.run('python nanoGPT/sample.py --out_dir=out shakespeare-char
    --device=cpu', shell=True)
```

Listing 4: Command used to start text generation over the acquired Lovecraft text knowledge

By applying all the default steps described above to the dataset using the predefined configuration, the model generates multiple output samples. One representative example among the generated outputs is the following:

Generated Output

The dreaps and presat of the sundess paters, sto told be't Chirlip efor Rhoumb120.

The ofthined him the they nougn for bourt the ounsonry with Sa. Hemed imeld a swill–healled house dener. On had shore danding the lies briddened consuttant and that renser, would with had hellf corrted in melly dow of things, FIn's what Jegan and men the Goulge ats the only Curwas we est was and belemently was canelly whoserable to the days and the alistable stions nough to be reach acch. He gatter tille the s

---

The result shows that, although the default model is capable of structuring sentences in a Lovecraftian style, it ultimately produces meaningless and random words lacking semantic coherence. While the output mimics the cadence and atmosphere typical of Lovecraft's writing, it lacks meaningful content and logical progression. This outcome highlights the limitations of the default configuration, which, although functional, does not yet possess the expressive depth needed to generate coherent literary text. Such constraints are expected given the relatively small embedding size and number of layers used in this baseline setup. The entire process was executed without CUDA integration and completed in approximately three minutes, during which no particular issues were encountered.

1.3 First Model

In light of the limitations observed in the default configuration, a first model was refined from default by applying some modification to default settings, as it is possible to notice in Table 3. Specifically, the *n_layer* parameter **was doubled from 4 to 8**, allowing the model to learn more abstract and hierarchical representations. The **number of attention heads** (*n_head*) **was also increased from 4 to 8**, enabling the model to capture a wider range of contextual relationships between tokens. The **embedding dimension** parameter **was expanded from 128 to 256**, enriching the semantic representation of the vocabulary. Additionally, the *block_size* **was extended from 64 to 128**, allowing the model to consider longer sequences during generation, which improves narrative coherence and the **number of training iterations was raised from 2000 to 5000** to provide the model with more time to converge.

Hyperparameter	Default	Model_1
eval_iters	20	20
log_interval	1	1
block_size	64	**128**
batch_size	12	12
n_layer	4	**8**
n_head	4	**8**
n_embd	128	**256**
max_iters	2000	**5000**
lr_decay_iters	2000	**5000**
dropout	0.0	0.0

Table 3: Settings modification from default model to first model

After adjusting the default training parameters and launching the updated experiment, the first model generated the following result:

```
subprocess.run('python nanoGPT/train.py nanoGPT/config/train_shakespeare_char
    .py --device=cpu --compile=False --eval_iters=20 --log_interval=1 --
    block_size=128 --batch_size=12 --n_layer=8 --n_head=8 --n_embd=256 --
    max_iters=5000 --lr_decay_iters=5000 --dropout=0.0',shell=True)
```

Listing 5: Command with adjusted hyperparameters used to run the train the first model.

Generated Output

The dark, and with the thing man to the countern of blasphemains to like the monstrous depths and slowled took. In the tension on the dispectance down his tince of children will crystation in the floath and dory, since it take in the dark and slopes that the blaspheron with the shadowy far of sunset. Capt. He ind. He had someted, and the descended floor a weaken them walking me broods are of either incidentiation the person which sign with its crazy a wall black in the children pretails slanting

The output generated by the first model reveals a significant improvement over the default configuration. While the text remains surreal and stylistically dense, it exhibits a clearer sentence structure, more consistent thematic elements, and a richer vocabulary. Unlike the baseline output, which consisted largely of invented and meaningless words, this version demonstrates

a partial semantic alignment with the training corpus, indicating that the architectural enhancements have positively impacted the model's generative capacity. However, despite these advancements, the model's performance remains insufficient for generating fully coherent literary text. Semantic inconsistencies persist throughout the output, with frequent grammatical errors, invented words, and abrupt transitions that undermine readability. These limitations indicate that, although the model has begun to internalize stylistic patterns, it still struggles to capture deeper semantic structures and maintain contextual continuity over longer spans.

1.4 Second Model

Following the partial improvements observed in the first model, a second configuration was designed to further expand the model's representational capacity and contextual horizon. The architectural structure was maintained with 8 layers and 8 attention heads, but the **embedding dimension was doubled** from **256** to **512** to allow for more expressive token-level representations and richer semantic encoding. Additionally, the *block_size* **was extended from 128 to 256** to enable the model to capture longer-range dependencies and improve narrative continuity. The number of **training iterations was increased from 5000 to 7000** to provide the model with more time to converge and internalize stylistic and structural patterns. *Dropout* was deliberately kept at 0.0 to observe the model's behavior under full exposure to the training data, following the observation that the first model did not exhibit signs of overfitting despite its limited regularization.

Hyperparameter	**Default**	**Model_1**	**Model_2**
eval_iters	20	20	20
log_interval	1	1	1
block_size	64	128	**256**
batch_size	12	12	12
n_layer	4	8	8
n_head	4	8	8
n_embd	128	256	**512**
max_iters	2000	5000	**7000**
lr_decay_iters	2000	5000	**7000**
dropout	0.0	0.0	0.0

Table 4: Settings modification from first model to second model

After adjusting the training parameters between the first and second model, the second experiment was launched with the intention of evaluating whether the revised configuration could lead to improvements in output quality. In particular, the model generated the following output:

```
subprocess.run('python nanoGPT/train.py nanoGPT/config/train_shakespeare_char
    .py --device=cpu --compile=False --eval_iters=20 --log_interval=1 --
    block_size=256 --batch_size=12 --n_layer=8 --n_head=8 --n_embd=512 --
    max_iters=7000 --lr_decay_iters=7000 --dropout=0.0',shell=True)
```

Listing 6: Command with adjusted hyperparameters used to run the train the second model.

Generated Output

Able things be the old much in his had it is material of the mountry to believe beyond he had astonishing and as it was the Furope of this would follow that no learned. The eight we had come of that he did not all house that the smell to the received of the uniced to the uncover box, which I had blooded by the black no place of howled could have me even a recognized by hint object shunned. He had come from the misfather and handwrite or Outside the say. He had security of the embalming, and tha

The output generated by the second model reveals a shift in behavior compared to the previous configuration. While the text remains semantically unstable, it exhibits a more fluid sentence structure and a noticeable reduction in invented or malformed words. This suggests that the increased embedding dimension and extended context window have allowed the model to internalize more consistent linguistic patterns. However, the overall coherence of the output remains limited with lacks of narrative direction and absent semantic relationships between sentences. Despite the architectural expansion, the model continues to struggle with maintaining logical continuity and thematic development across longer spans of text. Furthermore, the final `validation loss` (**2.5280**) indicates a decline in generalization performance compared to the first model, suggesting that the increased capacity may have led to a form of over-specialization, even in the absence of classic overfitting symptoms (as shown in the Listing 7).

```
iter 6996: loss 0.3826, time 1302.45ms, mfu 0.14%
iter 6997: loss 0.3451, time 1551.52ms, mfu 0.14%
iter 6998: loss 0.4412, time 1171.59ms, mfu 0.14%
iter 6999: loss 0.3876, time 952.83ms, mfu 0.14%
step 7000: train loss 0.3999, val loss 2.5280
```

Listing 7: Validation loss value after the training phase of model 2.

1.5 Third Model

Based on the limitations observed in the second model, particularly its tendency to overfit the training dataset at the expense of generalization, a third configuration was introduced with the aim of restoring balance between representational capacity and training stability (as shown in Table 5). **The embedding dimension was reduced to 384** to mitigate the risk of overfitting and improve efficiency, while a **dropout rate of 0.1 was added to encourage regularization** and reduce reliance on specific training patterns. The other settings remained unchanged except for the number of training iterations which was **scaled back to 5000** to prevent excessive convergence.

The third experiment aimed to evaluate the model's ability to produce more stable and coherent outputs under the revised training regime. The following sample illustrates the linguistic behavior observed in this configuration:

```
subprocess.run('python nanoGPT/train.py nanoGPT/config/train_shakespeare_char
    .py --device=cpu --compile=False --eval_iters=20 --log_interval=1 --
    block_size=256 --batch_size=12 --n_layer=8 --n_head=8 --n_embd=384 --
    max_iters=5000 --lr_decay_iters=5000 --dropout=0.1',shell=True)
```

Listing 8: Command with adjusted hyperparameters used to run the train the third model.

Hyperparameter	Default	Model_1	Model_2	Model_3
eval_iters	20	20	20	20
log_interval	1	1	1	1
block_size	64	128	256	256
batch_size	12	12	12	12
n_layer	4	8	8	8
n_head	4	8	8	8
n_embd	128	256	512	**384**
max_iters	2000	5000	7000	**5000**
lr_decay_iters	2000	5000	7000	**5000**
dropout	0.0	0.0	0.0	**0.1**

Table 5: Settings modification from second to third model

Generated Output

Yes–that he had actually this would daemon better him and arrow to explore for outside. He was not asked no go a primal public book of poten well, and uncourage that was not respondential when screamed it was in his fine end. It actually since to when he wished when he resolved the ships were locked only tangles underscored a little almost that through the silent soul rather core the gods of his meteria a great former early address. He had a large of the almost and some future brastic entrave

The output generated by the third model demonstrates a marked improvement in structural stability and lexical integrity compared to previous experiments. While semantic coherence remains limited, the sentence construction is more controlled, and the frequency of malformed or invented words is further reduced. The model exhibits a greater ability to maintain syntactic rhythm, and several segments suggest an emerging grasp of thematic continuity, albeit inconsistently. Despite residual semantic drift, the overall generative behavior reflects a more disciplined internalization of linguistic patterns. This is consistent with the observed stabilization of the training and validation losses, which converged to `training loss` (**1.1016**) and `validation loss` (**1.3806**) at the last step, indicating improved generalization performance relative to the second model (as shown in Listing 9). The reduction in loss values, coupled with the more restrained output, suggests that the revised configuration successfully mitigated over-specialization and enhanced the model's capacity to produce linguistically plausible sequences.

```
iter 4996: loss 1.1525, time 1610.93ms, mfu 0.06%
iter 4997: loss 1.1622, time 1567.52ms, mfu 0.06%
iter 4998: loss 1.2118, time 1632.57ms, mfu 0.06%
iter 4999: loss 1.2197, time 1609.56ms, mfu 0.06%
step 5000: train loss 1.1016, val loss 1.3806
```

Listing 9: Validation loss value after the training phase of model 3.

1.6 Analysis & comparisons

Modifying the hyperparameters during the training phase of a GPT model has proven to be a crucial step, as they directly affect how efficiently the model learns and how coherent and stylistically consistent its generated text becomes. In this experiment, the most impactful parameters were the **number of layers** and **attention heads**, **the embedding size**, and **the block size**. Increasing these values, as done in the first and second models, allowed the network to capture more abstract relationships and longer contextual dependencies, resulting in text that was stylistically closer to Lovecraft's prose. However, this also showed that simply expanding model capacity can lead to diminishing generalization ability, making the text semantically unstable (as observed in Model 2). To counter this effect, the third configuration focused on achieving a better equilibrium between representational power and training stability, **adjusting the embedding size and introducing a slight dropout rate**, demonstrating that regularization and balance between parameters are often more effective than pure scaling, improving both stability and lexical accuracy.

The size and nature of the corpus also had a major influence on the outcome. Its archaic vocabulary, elaborate syntax, and recurring thematic patterns posed a challenge for small-scale GPTs trained at the character level. The dataset's limited size and stylistic uniformity encouraged overfitting, leading models to reproduce surface-level stylistic elements without achieving full semantic coherence. Moreover, the character-level tokenization made it harder for the models to learn long-range dependencies and consistent narrative logic. As a result, the outputs often reflected Lovecraft's tone and rhythm but lacked meaningful content or sustained context.

Future experiments could explore word-level or subword-level tokenization to enhance semantic coherence and reduce the appearance of invented words. Expanding the corpus, either by including a broader range of Lovecraft's works or mixing texts from similar genres, could improve generalization and narrative consistency. Additionally, performing the training with GPU acceleration would allow for larger models, longer training iterations, and deeper evaluation of how model capacity influences stylistic and semantic fidelity.

References

[1] Dr. Tristan Behrens. Lovecraft corpus - a weird fiction dataset, 2024.

[2] Donovan K. Loucks. The h.p. lovecraft archive, 2024.