

# Generating Text-based Adventure Games

Anna Orosz

A THESIS

In

Data Science



Presented to the Faculties of the University of Pennsylvania in Partial  
Fulfillment of the Requirements for the Degree of Master of Science in Engineering

2021

---

Prof. Chris Callison-Burch

---

Prof. Clayton Greenberg

## Abstract

This thesis presents a method and analysis to generating tools for text-adventure game authors using OpenAI's GPT-3 API.

If the reader is familiar with Zork, Enchanter, Anchorhead, or even Colossal Cave Adventure, text-based adventure games might already sound familiar. To be more specific, text-based adventure games are one of the oldest video game genres, and often considered to be their origin.

Interactive Fiction games are fully text-based simulation environments where a player issues text commands to effect change in the environment and progress through the story. Figure 1 shows an example of such a game.

```
On the table is an elongated brown sack, smelling of hot peppers.
A clear glass bottle is here.
The glass bottle contains:
  A quantity of water.
>w
You are in the living room. There is a door to the east. To the west
is a wooden door with strange gothic lettering, which appears to be
nailed shut.
In the center of the room is a large oriental rug.
There is a trophy case here.
On hooks above the mantelpiece hangs an elvish sword of great antiquity.
A battery-powered brass lantern is on the trophy case.
There is an issue of US NEWS & DUNGEON REPORT dated 28-JUL-80 here.
>get sword
Taken.
>break egg with sword
You rather indelicate handling of the egg has caused it some damage.
The egg is now open.
There is a golden clockwork canary nestled in the egg. It seems to
have recently had a bad experience. The mountings for its jewel-like
eyes are empty, and its silver beak is crumpled. Through a cracked
crystal window below its left wing you can see the remains of
intricate machinery. It is not clear what result winding it would
have, as the mainspring appears sprung.
>_
```

Figure 1: An example for text-based adventure game

These games typically feature a text parser, a user interface that allows the player

to interact with the game solely using typed commands. They also feature a storyline which is mostly linear, although there are multiple possible ways to reach a given ending, and deliberate puzzles within the games.

Text-based games are a form of interactive fiction, the term used to describe a text-only computer game. The term refers to a story-based game that features only text, as opposed to a graphical interface. Text-based adventures were more popular before the introduction of the point-and-click interface, in part because they required fewer resources to implement.

30 years ago, on the subject of fictional text generation, the following exchange occurred in 1992 between Morpheus Nosferatu and Phil Goetz on Usenet, the 1990s version of Reddit:

From: goetz@acsu.buffalo.edu (Phil Goetz)

Subject: Re: Adventure generators (skippable)

Newsgroups: rec.arts.int-fiction

Date: 29 Oct 92 04:40:05 GMT

Sender: nntp@acsu.buffalo.edu

Organization: State University of New York at Buffalo/Comp Sci

morpheus@sage.cc.purdue.edu (Morpheus Nosferatu) writes:

> Has anyone ever worked on, or even heard of, an adventure generator?

>

> I'm not talking about an adventure design language like TADS or Alan,

> but rather a stand-alone adventure generator that produces complete

> adventures, where the user need only give a minimal degree of input,

> such as the level of complexity, type of adventure (mystery, treasure

> hunt, etc.), size of adventure, and so forth?

> ...

> But as anyone ever heard of someone trying to come up with a generator  
> which would produce infocom-style text adventures? I can just imagine  
> what kind of limitations it would have, but I'm curious to know if  
> anyone has tried this, and if so what degree of success they've had.

No. ... The generator you speak of is not written, not being written,  
and not anywhere on the horizon. In 50 years, maybe. In 20,  
definitely not. The problem of writing interesting stories, which  
adhere to someone's definition of a plot (with goal explanations,  
conflict, resolution, complication, climax, etc., all occurring at  
appropriate intervals) is very hard, and I don't expect a solution  
soon. But the problem of writing clever puzzles involves much greater  
creativity, and I have seen NO evidence that ANYBODY has a clue in  
these creativity issues; the most you will find in the field are a  
few vague theories of creativity.

This problem is what Stuart Shapiro calls "AI-complete": Solving it  
would be equivalent to solving all the other problems of AI.

Phil

This thesis shows that Phil Goetz's time estimate was surprisingly accurate, in that  
it illustrates how 30 years after that discussion, language models in 2021 are in fact  
capable of generating creative, human-like fictional texts.

# Contents

<b>1</b>	<b>Introduction</b>	<b>6</b>
1.1	Using large language models to generate text-based adventure games . . .	8
<b>2</b>	<b>Literature Review</b>	<b>9</b>
2.1	Text-based Adventure Games . . . . .	9
2.2	Large Neural Models . . . . .	11
2.3	Few-Shot Learning . . . . .	12
2.4	Fine-Tuning . . . . .	14
<b>3</b>	<b>Facebook’s LIGHT data</b>	<b>16</b>
3.1	Data Structure Examples . . . . .	17
3.1.1	Category (Setting) Schema . . . . .	17
3.1.2	Character Schema . . . . .	19
3.1.3	Object Schema . . . . .	20
<b>4</b>	<b>Generating Descriptions for items and rooms, and personas for characters</b>	<b>23</b>
4.1	Task . . . . .	23
4.2	Data split . . . . .	23
4.3	Methods . . . . .	25

4.3.1	Models . . . . .	25
4.3.2	Few-Shot Learning . . . . .	27
4.3.3	Fine-tuning . . . . .	29
4.4	Subjective Evaluation . . . . .	30
4.5	Results . . . . .	32
4.6	Examples . . . . .	36
4.7	Discussion on Fine-Tuning Experiments . . . . .	40
<b>5</b>	<b>Generating Item Attributes with Curie</b>	<b>41</b>
5.1	Task . . . . .	41
5.2	Natural vs non-Natural Language . . . . .	41
5.3	Models . . . . .	42
5.4	Attributes . . . . .	43
5.5	Objective Evaluation . . . . .	44
5.6	Error Analysis . . . . .	46
5.7	Final thoughts on generating attributes . . . . .	47
<b>6</b>	<b>Conclusion and Future Work</b>	<b>49</b>
<b>7</b>	<b>Acknowledgements</b>	<b>52</b>

# Chapter 1

## Introduction

Text-adventure games - Why are they interesting in Natural Language Processing? As the thesis has previously mentioned, generating fictional text was - for the longest time - thought to be impossible in the world of Natural Language Processing. However, with the invention of bigger and better language models than ever before (such as OpenAI's GPT-3), imitating the creativity and variability of a human's phrasing, terminology and style. The following are some areas that are going to be explored in this thesis:

- What makes a good text-adventure game? Due to the subjective nature of the question, there are several ways to answer this. However, there is one thing that is generally agreed upon: having creative rooms, objects characters that have unique and colorful identifiers enrich the game, which make the creative side of the language model so critical.
- How to generate game-appropriate text? There are a few methods to achieve this goal, of which two are going to be explored further in this thesis.

Fine-tuning, which happens when there is a large amount of training data available and thus makes it possible to retrain the last few layers of the model.

The second method is Few-shot learning, which is possible with only a handful of appropriate examples. Depending on the size and capabilities of the language model, few-shot learning could imitate the competence of Fine-tuning with much less overhead.

- What is the most cost- and time-efficient method to implement for such auto-generation of Text-adventure games? To answer this question, several different engines by OpenAI will be tested, such as Davinci, which is the largest and strongest engine, Curie, which is slightly less powerful yet very effective, and Babbage, the weakest of the three. Since Davinci is the most effective, however, it also comes at a much higher cost and therefore an evaluation of the cost-to-value ratio needs to be done.
- How to evaluate the generated text and attributes? For different tasks, there are different ways to evaluate the performance of the language model.

In the case of text generation, subjective evaluation was applied to be able to differentiate between a human and a language model in terms of their ability to mimic a certain style and their creativity.

However, in the case of item attributes, a traditional binary classification is sufficient, which the thesis will discuss later at length.

- How to generate the game? The thesis presents a strategy to generate such a convoluted and involved fantasy world.

First, descriptions of rooms, characters and objects are to be generated using several different versions of engines, some trained with fine-tuning and/or few-shot learning.

Secondly, item attributes are generated using only few-shot examples due to their



nature of being binary attributes and require much less training of the model.

## 1.1 Using large language models to generate text-based adventure games

The rest of the Thesis is structured as follows:

- Chapter 2 provides a Literature Review that goes over research about language generation and the most advanced models in the field, as well as various techniques, such as few-shot learning and fine-tuning.
- Chapter 3 goes into further detail about the data that was used by this thesis, the LIGHT data from Facebook’s publication ”Learning in Interactive Games with Humans and Text”.
- Chapter 4 outlines the method employed in generating descriptions for various aspects of the game. To assess the the result, subjective evaluation is to be used to analyze the descriptions and compare them to that of humans’.
- Chapter 5 describes the method with which attributes were generated for objects with the Curie model as well as the objective evaluation used for analyzing the results.
- Chapter 6 discusses a summary of the results and potential directions for future work.

# Chapter 2

## Literature Review

### 2.1 Text-based Adventure Games

For any task that calls for storytelling skill, creating a fictional world is the most important. The authors of the paper “Bringing Stories Alive: Generating Interactive Fiction Worlds” concentrate on developing creative fantasy worlds. These are worlds that players interact with using natural language. To create these worlds, the game environment not only needs to be semantically logical, coherent, and persistent but also demands a thorough understanding of everyday rational.

P. Ammanabrolu et al elaborate on an approach that derives a knowledge graph that incorporates vital information regarding world structure such as rooms and items, using existing fictional writing as a template. The graph is needed to extract thematic knowledge and for it to guide a language model that generates the rest of the fantasy world. Similar to this thesis, the authors also chose to do evaluation with human participants to test their neural language model’s ability to derive information and build a knowledge graph and to generate text against human-formulated language. [1]

It is a widely accepted fact that generating creative and logical narrative-style

game worlds is one of the most cumbersome, expensive and time-consuming challenges in natural language processing. The authors of “Generating Interactive Worlds with Text” [4] argue that logic needs to be built into the language model in order for the relationships between different elements of the game (such as locations, characters and objects) to make sense together. A. Fan et al study a method where they generate a fantasy environment utilizing the LIGHT game environment (identical to the data that this thesis used).

They propose a language model that is able to generate convoluted web of rooms, personas, items into a world that is coherent and logical to the player. The goal of the authors is to not only understand the connection between the existing elements but to build upon them, similarly to the goal of this work. Several of the strategies mentioned in the paper ended up in this thesis, since it was shown that the worlds generated with this method are cohesive, diverse. Furthermore they were convincing to the human evaluators when compared to other worlds constructed by neural networks. [4]

Some may argue that the ability to understand and communicate with language is the biggest virtue of human intelligence , similarly to the authors of “Interactive Fiction Games: A Colossal Adventure”. [6] They argue that since interactive fiction games are a perfect combination of logical reasoning, natural language understanding and convoluted action spaces, these worlds provide a great testing environment for language-based autonomous agents.

This is most likely also the reason why Facebook created their LIGHT environment to generate autonomous dialogues with their help, as well. [12] Furthermore, the authors of the paper “What can you do with a rock? Affordance extraction via word embeddings” also applies this approach to a RL (reinforcement learning) agent in a text-based world. [5]

Similarly, TextWorld is a sandbox learning environment for the training and as-

assessment of reinforcement learning agents on text adventure games. Likewise to the goal of this thesis, the motivation behind this work was to enable users of TextWorld to automatically construct new games. The authors of “TextWorld: A Learning Environment for Text-based Games” also point out that it gives the users precise control over the language of generated games as well as the complexity and scope. N. Fulda et al emphasize that TextWorld can not only be utilized for fictional text generation but also be used to study transfer learning as well as generalization. [3]

## 2.2 Large Neural Models

For generating text, I chose the state-of-the-art language model, OpenAI’s GPT-3 [2]. This model was built with the goal to use and improve previous works that have illustrated considerable improvements on many natural language processing applications and standards by training on an enormous amount of content (as large as 45TB of text data) which is then followed by fine-tuning for a specific job. [2]. The model is impressive even by its size: GPT-3 is an autoregressive transformer language model with no less than 175 billion parameters. This is ten times more than any previous dense (non-sparse) language model.

GPT-3 is revolutionary because even though most previous ground-breaking models have been task-agnostic in architecture, they still required task-specific fine-tuning datasets of thousands or tens of thousands of examples – which is unnecessary for humans to do. Fine-tuning in the field of natural language processing refers to the procedure of re-training the last few layers of a language model - that was pre-trained on a huge corpus of text - using own custom data, resulting in that the weights of the initial model are modified to account for the characteristics of the custom data and the task at hand.

However, OpenAI has shown that—by scaling up their language model—task-agnostic, few-shot performance greatly improves. Few-shot learning means that a model is trained on some classes and predict for a new class, which the model has only seen a handful of examples of. To test this capability on text-adventure games, this thesis will dive into some experiments that were conducted on models trained with fine-tuning only, few-shot examples only as well as with fine-tuning and few-shot examples.

In its introductory paper, Brown et al. [2] tested GPT-3 on several different tasks without fine-tuning and only used few-shot examples in their analysis. They found that the model performs consistently and extremely well on many NLP datasets, such as translation, question-answering, and Cloze tasks, as well as several tasks that require on-the-fly reasoning or domain adaptation, such as unscrambling words, using a novel word in a sentence, or performing 3-digit arithmetic.

It is also important to note that the authors describe an experiment after which they concluded that GPT-3 can create news articles that humans find genuinely difficult to distinguish from human-written articles. [2]. This analysis prompted the “subjective evaluation” part of this thesis, where annotators were asked to distinguish between a human-written and 4 different generated texts through crowdsourcing (See Chapter 5).

## **2.3 Few-Shot Learning**

Even before GPT-3, countless large-scale pre-trained language models have demonstrated remarkable abilities as few-shot learners, thus helping Natural Language Processing to make enormous strides in the last few years. Many have found however that they are still yet to be particularly useful in real-life scenarios, since their true capabilities are limited by the model parameters and design.

The authors of Differentiable pRompT (DART) [13] came up with a method which can convert smaller, less capable language models into better models and few-shot learners without having to do any prompt engineering.

This method works by essentially translating natural language processing tasks to the task of a pre-trained language model as well as differentially optimizing not only the target label but also the prompt template with backpropagation. This thesis also has a significant amount of prompt engineering, where several of the above mentioned strategies served as guidance (see more in Chapter 5).

Naturally, after generating text with the help of fine-tuning, a big question remains to be answered: how to evaluate text-based adventure games, which are similar in nature to fictional texts? Natural Language Processing has been going through a revolution in the past decade largely due to amazing advancements in large-scale language models [11]. It is undeniable that they have brought considerable qualitative as well as quantitative advancements in the field of automated language generation.

However, Matiana et al. [9] argue that creating and evaluating of fictional text remains a difficult task, since objective evaluation of machine-generated narrative text may need human-annotated datasets and would be exceedingly expensive. Even so, it is highly likely that such an evaluation would not appropriately assess the logical coherence of a generated text's fictional structure.

However there have been significant advances in contrastive learning [10], with the help of which Matiana et al. [9] present Contrastive Authoring and Reviewing Pairing (CARP). It is a powerful and scalable method for performing qualitatively superior, zero-shot assessment of narrative text, in fact the paper describes a solid correlation between the CARP evaluation and those of humans.

Similarly to the paper's suggestion, this thesis describes human evaluation of machine-generated fiction-like text at length.

## 2.4 Fine-Tuning

A significant work in the field of fine-tuning has been the research about prompting language models with training examples and task descriptions by using the driving force of few-shot learning.

In fact, the authors of the paper "Cutting down on prompts and parameters: Simple few-shot learning with language models" discuss that fine-tuning language models in the few-shot setting can significantly lower the need for prompt engineering. They argue that one achieve competitive accuracy to manually-tuned prompts across a wide range of tasks even if they use null prompts or prompts that contain neither task-specific templates nor training examples.

Through the act of fine-tuning, new parameters are generated for each individual task, however the architects of this model point out that fine-tuning only the bias terms can achieve comparable or better accuracy than standard fine-tuning while only updating 0.1 percent of the parameters. Thus, the memory overhead of fine-tuning can be radically mitigated. [8]

Arguably, previous GPT models that were fine-tuned the traditional way have failed to show good results on natural language understanding. However, with the method of P-tuning, Liu et al. [7] demonstrate that GPTs can be better than or comparable to similar-sized BERTs on natural language understanding tasks.

This method employs trainable continuous prompt embeddings. On the knowledge probing (LAMA) benchmark, the best GPT recovers 64 percent (P@1) of world knowledge without any additional text provided during test time, which substantially improves the previous best by 20+ percentage points [7]. On the SuperGlue benchmark, GPTs achieve comparable and sometimes better performance to similar-sized BERTs in supervised learning. Importantly, we find that P-tuning also improves

BERTs' performance in both few-shot and supervised settings while largely reducing the need for prompt engineering. Consequently, P-tuning outperforms the state-of-the-art approaches on the few-shot SuperGlue benchmark.



# Chapter 3

## Facebook’s LIGHT data

Facebook introduced a large scale crowdsourced text adventure game as a research platform for studying dialogue grounded in a virtual setting [12]. In it, AI agents (or human players) can perceive, emote, and act whilst conducting dialogue with other agents.

Models and humans can both act as characters within the game. Similarly to this thesis, the authors of the LIGHT (Learning in Interactive Games with Humans and Text) data paper conducted experiments for training state-of-the-art generative and retrieval models in the world of text-adventure games. [12]

It contains two files: `light_data.json` and `light_unseen_data.json`, both of which are datasets containing dialogues, which was the main goal of Facebook’s research was focused on generating dialogues. It contains a third file, `light_environment.json`, is a dataset made of the entire world of this text-adventure game. The goal of the thesis is not to primarily generate dialogue, rather to generate situations, filled with specific rooms, characters and objects. The LIGHT environment dataset is a great fit for the tasks in this thesis.

It is structured as a hash table, where some of the keys found are categories, char-

acters, objects, neighbors, etc. It is important to note that each category consisted of several rooms and that each room could only exclusively be found in one category.

Furthermore, all rooms, characters, as well as objects have a myriad of attributes associated with them (binary, for example if a given object is a weapon or not) and of course one or several descriptions (in the case of characters, both a persona and a description came with each character).

To better understand the data structure, let's look at the schema more closely below:

## 3.1 Data Structure Examples

### 3.1.1 Category (Setting) Schema

Text adventure games have a series of locations (sometimes called “rooms”) that a player explores. Each location has a description that is displayed to the user as she enters. Rooms may also contain objects and characters.

The Facebook LIGHT Environment data includes information about locations. The information that is stored about each location includes:

- *setting* - the name of the location
- *description* - a description of the location
- *background* - additional information about the location
- *room\_id* - a numeric identifier for the location
- *category* - a name for the ‘world’ that the location belongs to
- *in\_characters* - characters that are explicitly mentioned listed in the description or the background

- *ex\_characters* - characters that are possibly present but not mentioned directly
- *in\_objects* - things that are explicitly mentioned listed in the description or the background
- *ex\_objects* - things that are possibly present but not mentioned directly
- *neighbors* - the locations that are adjacent to this location.

Here is an example of the data structure for a setting called “An Unfinished Mausoleum” from the “Graveyard” category.

**Example: An Unfinished Mausoleum**

```
{'setting': 'An Unfinished Mausoleum'}
  'description': 'Two-and-a-half walls of the finest, whitest stone
stand here, weathered by the passing of countless seasons. There
is no roof, nor sign that there ever was one. All indications are
that the work was abruptly abandoned. There is no door, nor
markings on the walls. Nor is there any indication that any
coffin has ever lain here... yet.',
'background': "Bright white stone was all the fad for funerary
architecture, once upon a time. It's difficult to understand why
someone would abandon such a large and expensive undertaking.
If they didn't have the money to finish it, they could have sold
the stone, surely - or the mausoleum itself. Maybe they just
haven't needed it yet? A bit odd, though, given how old it is.
Maybe the gravedigger remembers... if he's sober.",
'room_id': 62,
'category': 'Graveyard',
```

```
‘ex_characters’: [204, 75, 156, 720],
‘ex_objects’: [1791, 1792, 439],
‘in_characters’: [203, 203],
‘in_objects’: [1790],
‘neighbors’: [108, 109],
}
```

### 3.1.2 Character Schema

The LIGHT data includes non-player characters (NPCs). These characters have a name, a description, and a persona. A persona is written in the first person, as if the character is introducing herself. Characters can have objects that they are carrying, wearing or wielding.

Here is an example of a character from the dataset. It includes additional information about the type of character (person or creature or animate object), and rooms where the character might be located.

**Example: gravedigger**

```
{‘name’: ‘gravedigger’,
‘char_type’: ‘person’,
‘desc’: ‘You might want to talk to the gravedigger, specially if
your looking for a friend, he might be odd but you will find a
friend in him.’,
‘personas’: ["I am low paid labor in this town. I do a job that
many people shun because of my contact with death. I am
very lonely and wish I had someone to talk to who isn't dead."],
‘corrected_name’: ‘gravedigger’,
‘character_id’: 203,
```

```
‘base_form’: [‘gravedigger’],
‘is_plural’: 0,
‘ex_room_ids’: [100, 349],
‘in_room_ids’: [62],
‘orig_room_id’: 349,
‘carrying_objects’: [890],
‘wearing_objects’: [],
‘wielding_objects’: [],
}
```

### 3.1.3 Object Schema

In addition to locations and characters, text adventure games have objects that players can interact with by picking up and using. Different objects have different uses. For instance, some can be used as a weapon, and some can be worn. What an object can be used for depends on its properties. The LIGHT dataset defines several different properties, which are represented as binary values on each object. These binary values are:

- is container - can be used to store other objects
- is drink - can be drunk
- is food - can be eaten
- is gettable - can be picked up and put into the player’s inventory
- is plural - this is a plural noun
- is surface - other objects can be put onto this object

- is weapon - can be used as a weapon
- is wearable - can be worn like clothing

Objects also have a description. Since the LIGHT data was originally collected from multiple people, many objects have multiple descriptions written by different people. The descriptions are stored in a list, and the 'desc\_entries' variable indicates how long the list is.

Each object has a name, a linguistic base form, and a numeric ID. Here is an example of the data stored for the object 'Legendary Swords'. **Example: Legendary swords**

```
{'name': 'Legendary swords',
 'object_id': 1188
 'base_form': ['sword', 'Sword'],
 'desc_entries': 2,
 'descriptions': ['The sword is very old, you would assume it had
 once belonged to a legendary warrior.',
 "The sword's legend is known by everyone, it is famous
 throughout the land."],
 'ex_room_ids': [],
 'holding_character_ids': [],
 'in_room_ids': [12],
 'is_container': 0.0,
 'is_drink': 0.0,
 'is_food': 0.0,
 'is_gettable': 1.0,
 'is_plural': 1.0,
```

```
'is_surface': 0.0,  
'is_weapon': 1.0,  
'is_wearable': 0.0,  
'link_entries': 1}
```

For the experiments in the next chapter, I use the LIGHT data to train a text generation system to generate a description given the name of a location, or the name of a character, or the name of an object.

# Chapter 4

## Generating Descriptions for items and rooms, and personas for characters

### 4.1 Task

One of the main tasks in this thesis was generating descriptions for rooms as well as attributes and personas for the characters. Given the name, the language model was tasked with creating descriptions that not only closely resemble but outperform human-written descriptions. These descriptions not only have to be of similar length but also stylistically match fiction-like adventure games.

### 4.2 Data split

To make sure that the data that was used during the fine-tuning of the model or utilized as one of the few-shot examples, I chose to do the traditional 80-10-10 split.



- 80 percent of the data was used for training purposes a.k.a. for fine-tuning the GPT-3 model Curie
- 10 percent of the data was used to give the model few-shot examples (Davinci, as well as Curie and Babbage)
- and the remaining 10 percent was used to generate descriptions for.

The three “datasets”, or rather parts of the dataset that this thesis is mostly concerned with, had vastly different sizes/lengths. Let’s look at how the size of the train/dev/test set differed for each of these three aspects:

- Rooms:

train: 532

test: 66

dev: 63

- Characters:

train: 1405

test: 175

dev: 175

- Objects:

train: 2770

test: 346

dev: 346

The rooms had the smallest number of examples, and therefore the fewest number of examples that was at my disposal to train / fine-tune with, namely 532 name-description pairs (or in terms of fine-tuning, prompt-completion pairs).

Characters had slightly more, 1405 examples that were available for training as well as 175 examples to be used for few-shot learning. However, 175 examples exceeded to the number of maximum tokens that any one model could take in as input, which is why the number of available examples in the dev dataset had to be reduced when used in fine-tuning.

However, objects provided the best possibilities to fine-tune with and to test the capabilities of this (potentially) powerful method with. 2770 prompt-completion examples was about 6 times as much as the number of examples available in rooms and twice as much as what was available in characters. Therefore, my initial expectation was that the object-fine-tuned Curie would perform the best out of all fine-tuned models.

## 4.3 Methods

### 4.3.1 Models

For this task several different models were chosen in order to generate descriptions with.

- First, Davinci was an obvious choice, being the strongest, largest and best performing GPT-3 model. For calling this model through the API, I did the following: I created a prompt or "input" by selecting a handful of descriptions for each of the rooms, objects and characters datasets and gave that to the model. One thing to note is that I had to adjust the stop sequence the token length, etc. for the model to behave appropriately. To get the model to generate the descriptions, the name of the room/setting, object or character had to take place as the last line. Since the model was given few-shot examples where the description ended with a stop sequence, the model did not "over-generate" and generated texts of

appropriate length with the same stop sequence appended to it.

- Secondly, Curie was chosen as a candidate to be fine-tuned and held in direct comparison to Davinci with. Similarly to Davinci, the settings had to be carefully set such that the API call returned the generated text in the expected format and of appropriate length.
- In addition to the Fine-tuned Curie model, I also added an extra layer of “training” in the form of few-shot learning, during which I gave the model an extra layer of examples to create descriptions for characters, rooms and objects.
- Lastly, Babbage was chosen a sort of “control” model. Being significantly weaker than the other two, having been trained with less parameters, etc. it was expected that it would perform considerably worse than both Davinci and Curie. Of course, the same settings and data was given to Babbage as the other two, such that “equal footing” was ensured for all three models.

In Google Colab, where most of the research for this thesis was completed, the API call to OpenAI’s model for text generation looked like the following:

```
response = openai.Completion.create(  
    engine="davinci",  
    prompt=input + start_sequence + '\n',  
    temperature=0.7,  
    max_tokens=64,  
    top_p=1,  
    frequency_penalty=0.5,  
    presence_penalty=0,  
    stop=["==="])
```

Here, engine represented the chosen model, the prompt consisted of the few-shot learning examples (if relevant to the experiment) as well as the `start_sequence` which was the prompt for each example.

The max number of tokens was chosen to be 64, since the original descriptions were fairly short, as well. Furthermore, I found that even by increasing this number to 256 for example, the model mostly generated shorter texts, similar length to the original descriptions.

### 4.3.2 Few-Shot Learning

To generate text without fine-tuning the model, few-shot learning was required for both Davinci, and the control model Babbage.

To be able to come up with a format for the input, I utilized OpenAI's Playground feature. First, only name-descriptions were input with a couple of lines in between each example. However, after several trial and error iterations, it was decided that adding a specific stop sequence, namely "[===]" followed by two newlines, would serve best for our purposes. For more detail, please see the API call for each model in the next section, where the only difference between calls was the setting for the model parameter.

For reference, the few-shot learning examples looked the following when they were used as input to the models:

Cottage safe-room

Stocked from floor to ceiling, food and medicine line the western wall. The room is painted matte grey and contains a chest to the left of the food and medicine. Three chairs and a table with a lamp are what is left in the middle of the safe-room. Dimly lit, and a steady water supply coming from the well outside of the cottage.

===

#### Cottage entryway

A welcome mat adorns the floor at the foot of the door. A rack to hang hats and coats is on the left hand side. Right above, a fan lazy spins about on a hot summer day. A single bulb of light illuminates the small hall that the entrance leads to. The entrance smells of the fresh outdoors.

===

#### Unexplored jungle

Located outside of civilization, the jungle is a vast unknown. Many plants never seen, many beasts never discovered. However, what is known is that one cannot survive alone in this jungle. There are many poisonous trees and plants and it is impossible to see the sun once in it. Just an incredible amount of vast, dense and impassable vegetation.

===

#### Abandoned Mine

As shown above, each example begins with the name of a room (or a character or character), followed by the description after a newline character. Each example is separated by a stop sequence, which in the case of this thesis is “===”.

### 4.3.3 Fine-tuning

To test the fine-tuning capabilities, I chose to fine-tune Curie to be held against the high standards of Davinci. Specifically, 80 percent of the available dataset in `light_environment.json` was formatted such that each object, room and character was split into a “prompt”, which was the name of the respective subject/object and a “completion”, which was the description of each of the character, object and room.

Furthermore, during the first iteration of training, I ran into a problem where the model would generate a longer-than-expected text and because of the limitation of 256 tokens, would stop mid-sentence. To mitigate this problem, a stop sequence was added to the end of each “completion” string, such that the model would not generate longer than appropriate text. After this fix, the model generated text that was of similar length to the original examples and to the ones that Davinci generated.

For reference, please see an example for one prompt-completion pair used for fine-tuning below.

```
{"prompt": "The rectory", "completion": "This room is quite small  
and cramped. It's about the size of maybe three wooden carts,  
which is to say, it's very small. There are boxes all over the place  
and many candles and other church accessories. There are several  
big robes hanging next to what looks like a very small closet.  
Some candles shed an eerie light on the room, flickering softly.  
There is a small cabinet with several religious tapes and records  
and a few books. A book case is near and contains many  
common religious texts.\n===\n\n"}
```

Furthermore, it is important to point out that the examples had to be formatted

to be in a .jsonl data format, which the OpenAI's fine-tuning API expected.

After fine-tuning, the resulting model had to be called the following way:

```
response = openai.Completion.create(  
    model='curie:ft-ccb-lab-members-2021-11-30-22-34-52',  
    prompt=start_sequence,  
    temperature=0.7,  
    max_tokens=256,  
    top_p=1,  
    frequency_penalty=0.5,  
    presence_penalty=0,  
    stop=["==="]  
)  
  
return response.choices[0].text
```

## 4.4 Subjective Evaluation

In total, 4 different models were used for generating descriptions for rooms, characters and objects. These models were:

- Davinci with few-shot learning
- Fine-tuned Curie
- Fine-tuned Curie with few-shot learning
- Babbage with few-shot learning

Additionally, for testing purposes, the original description (text) was added in the testing phase.

For collecting human-annotated data, which arguably is the most accurate evaluation of generated natural language we used Amazon’s Mechanical Turk as a user interface, but we recruited our own annotators rather than relying on crowd workers. Our participants were mostly students in Professor Callison-Burch’s Artificial Intelligence class (CIS 521) who were offered extra credit to participate. In total 45 students participated rating a total of 66 locations, 175 characters and 346 objects.

To set up the experiment, each description was collected and written to a CSV file, in this order: original, Davinci with few-shot learning, Fine-tuned Curie, Fine-tuned Curie with few-shot learning, Babbage with few-shot learning for each of the rooms, characters and objects datasets.

These CSV were then uploaded to Mechanical Turk and 3 separate batches were released. The annotators were then given the name of the subject and the five description in the same order as described above. Several methods for the ratings were considered, such as merely selecting the “best” option out of all descriptions, ranking all 5 of them or even rating different aspects of each description.

However, the final format for annotating the generated texts was the following: For each description, the annotator had to rate the description on a 5-point Likert scale (with 5 being the best).

The tasks were set such that each example was shown to 5 separate annotators to ensure agreement between them and weed out any outliers.

The batches looked like the following:

- *rooms* had 66 unique examples and resulted in 460 separate Mechanical Turk tasks
- *characters* had 175 unique examples and resulted in 350 separate Mechanical Turk tasks



- *objects* had 346 unique examples and resulted in 330 separate Mechanical Turk tasks

## 4.5 Results

To evaluate the results of the Amazon Mechanical Turk annotation tasks, there are several analyses that were conducted for this thesis.

- Firstly, the number of 1, 2, 3, 4 and 5 ratings were counted for each individual version (Tables 4.1, 4.3, & 4.5). It was the initial thought that the description with the most number of 5's (the highest rating) may be a good indicator for the quality of said description, however after some deliberation and analysis, I decided against relying this oversimplified metric.
- Secondly, I decided to compare the fine-tuned Curie (without conducting any few-shot learning) against all other versions, to be able to get a simple 1-vs-1 comparison (Tables 4.2, 4.4, & 4.6). Here, for each data point (which was the rating of a specific room/character/object by a specific annotator) the ratings of a pair of descriptions was compared to each other. More specifically, when the original description and the one with fine-tuned Curie was being considered, there was a very simple comparison conducted: was the rating of one higher than the other? Then, the cases where fine-tuned Curie was rated higher vs when it was rated lower and finally when it was rated equally were counted and divided over the overall count to arrive at a percentage for each case.

In the case of the room dataset, the results (Table 4.1) were somewhat surprising and unexpected. For example, the number of 5 ratings were the exact same for the original description and the one generated by Babbage, which was included in the

Rooms					
Sum	Original	Davinci with few-shot learning	Fine-tuned Curie	Fine-tuned Curie with few-shot learning	Babbage with few-shot learning
1	11	6	125	31	22
2	45	49	63	64	67
3	135	123	68	127	126
4	139	168	93	123	115
5	130	114	111	115	130

Table 4.1: [Number of 1-5 ratings received for each model in the room dataset]

analysis as a control model. Even more surprising and disappointing was the fact that Davinci and Fine-tuned Curie with few-shot learning performed worse than the original and Babbage and Fine-tuned Curie without few-shot learning scored even lower on the number of 5 ratings received.

Therefore, I decided to conduct the second analysis, where pairs of descriptions were compared to each other individually, rather than look at all descriptions at once (Table 4.2). Through this analysis, I found that fine-tuned Curie performed worse against all 4 other descriptions. However, what was different in this metric is that the original version as well as the Davinci-generated descriptions performed strongest against Fine-tuned Curie. Both Babbage and Fine-tuned Curie with few-shot learning were slightly weaker according to this metric, however they were still stronger than Fine-tuned Curie itself.

For characters, I was able to observe slightly different results (Table 4.3). When the number of 5 ratings were observed, Davinci performed stronger than any of its counterparts. Fine-tuned Curie with few-shot learning and Babbage surprisingly performed similarly, with Fine-tuned Curie still lagging behind. Surprisingly, the original description received the least number of the highest ratings.

Rooms				
metric	Original vs Curie	Davinci vs Curie	Curie w/ few-shot vs Curie	Babbage vs Curie
fine-tuned Curie better	27.61	27.83	33.26	29.78
other model better	52.61	50	46.52	46.96
equal	19.78	22.17	20.22	23.26

Table 4.2: [Percentages representing when the observed model (fine-tuned Curie) performs better, worse or equal]

Characters					
Sum	Original	Davinci with few-shot learning	Fine-tuned Curie	Fine-tuned Curie with few-shot learning	Babbage with few-shot learning
1	36	26	44	40	55
2	65	70	67	62	73
3	91	81	83	90	72
4	97	83	84	80	74
5	61	90	72	78	76

Table 4.3: [Number of 1-5 ratings received for each model in the character dataset]

Similarly to the rooms dataset, a comparison-analysis was also conducted for characters and got slightly more promising results (Table 4.4). To be more specific, Fine-tuned Curie performed better than both the original description and Babbage, the control model.

This improvement and the fact that the results resemble the expected results better can be attributed to the fact that for characters, there were about 3 times more training samples to fine-tune the model with than in the case of rooms.

Furthermore, Fine-tuned Curie performed comparatively to both Davinci and Fine-tuned Curie with few-shot learning, albeit still under-performing when compared to

Characters				
metric	Original vs Curie	Davinci vs Curie	Curie w/ few-shot vs Curie	Babbage vs Curie
fine-tuned Curie better	36.57	33.43	32	38.57
other model better	34.29	39.43	34.86	33.71
equal	29.14	27.14	33.14	27.71

Table 4.4: [Percentages representing when the observed model (fine-tuned Curie) performs better, worse or equal]

those.

Objects					
Sum	Original	Davinci with few-shot learning	Fine-tuned Curie	Fine-tuned Curie with few-shot learning	Babbage with few-shot learning
1	36	26	44	40	55
2	65	70	67	62	73
3	91	81	83	90	72
4	97	83	84	80	74
5	61	90	72	78	76

Table 4.5: [Number of 1-5 ratings received for each model in the object dataset]

Since the object dataset (Tables 4.5 & 4.6) included the most training samples available for fine-tuning (6 times as much as in the case of rooms) it was expected that object-fine-tuned Curie would perform best out of the three experiments that was conducted in this portion of the thesis.

Fittingly, even though the original description received the highest number of 5 ratings, the second highest count for 5 ratings went to Fine-tuned Curie. Compared to these two, Davinci, Fine-tuned Curie with few-shot learning as well as Babbage underperformed.

Objects				
metric	Original vs Curie	Davinci vs Curie	Curie w/ few-shot vs Curie	Babbage vs Curie
fine-tuned Curie better	27.27	47.88	51.21	52.12
other model better	39.39	22.73	22.12	23.03
equal	33.33	29.39	26.67	24.85

Table 4.6: [Percentages representing when the observed model (fine-tuned Curie) performs better, worse or equal]

However, after looking at the other metric and way of analyzing, the results seemed even more promising. To be more precise, Fine-tuned Curie outperformed all other available options. It did significantly better than the control model Babbage, the Fine-tuned Curie with few-shot examples as well as Davinci. Though when compared to the original, human-written descriptions the difference was not nearly as big, it still confidently outperformed this version, as well.

## 4.6 Examples

Let's look at some of the best-rated (Table 4.7) and worst-rated (Table 4.8) room descriptions (overall) below:

Old Queen Farm		
Engine	Generated description	rating
Original	Once a fairly profitable farm, the Old Queen Farm is now a deteriorated old shack surrounded by uninhabitable fields with soil incapable of growing even weeds. The farm house shows signs that is being occupied by wild animals, and the roof is beginning to cave inwards.	5
Davinci with few- shot learning	The Old Queen Farm is a dilapidated farmhouse with a tiny little garden. It has a barn and stables for animals. It has a long driveway and meadow behind it. The farm has been vacant for over a year, but the grass hasn't grown in the meadow or by the house or by the barn or stables, because of some magic that keeps it from growing. The house is empty but perfectly clean, because it's being maintained by magic.	4
Fine- tuned Curie without few- shot learning	The old queen farm is a large, grassy field that is surrounded by several oak trees. There are two large, open barns on opposite sides of the field. The old wooden fence is beginning to show signs of age and wear.	5

Old Queen Farm		
Engine	Generated description	rating
Fine-tuned Curie with few-shot learning	This farm has been in the family for many generations and is very well known. The farm is enormous, stretching as far as the eye can see. It's barns are huge and full of every imaginable animal, and there are wagons with supplies coming and going all day. The fields are green with growing crops and cows grazing. There is a huge castle in the distance, but it seems to be out of place here.	5
Babbage	This farm is the oldest one in the kingdom. It's run by the queen and her butler, a former soldier who has been transformed into a servant. The queen and her new assistant, the butler, are constantly arguing about what to do with it.	5

Table 4.7: [Description generated by each model for Apple trees, which received one of the highest overall scores]

Apple trees		
Engine	Generated description	rating
Original	the apple trees are a beautiful color and very nice to look at they are easy on the eyes and attract visitors as a result	2
Davinci with few-shot learning	The apples are ripe and red, ready to be plucked by the lucky who find them.	1

Apple trees		
Engine	Generated description	rating
Fine-tuned Curie without few-shot learning	lined the outside of the orchard. The apples were red and yellow and orange. Delicious smells filled the air.	3
Fine-tuned Curie with few-shot learning	The trees are old, but young in comparison to the people who love them.	1
Babbage	The apple tree is not right next to a field of apple trees, nor is it in a forest. The apple trees are clear in the distance, but not close enough for the apples to be easily picked.	2

Table 4.8: [Description generated by each model for Apple trees, which received one of the lowest overall scores]

Upon analyzing these two examples, one difference that stands out instantly is that the longer and more elaborate description was the one that received one of the highest ratings whereas the description with the lowest rating received some of the lowest. This begged the question whether the length of descriptions are a good indicator for human



ratings for descriptions, which can be seen in Figure 4.1. Here, a very slight positive correlation can be observed between the length and the ratings.

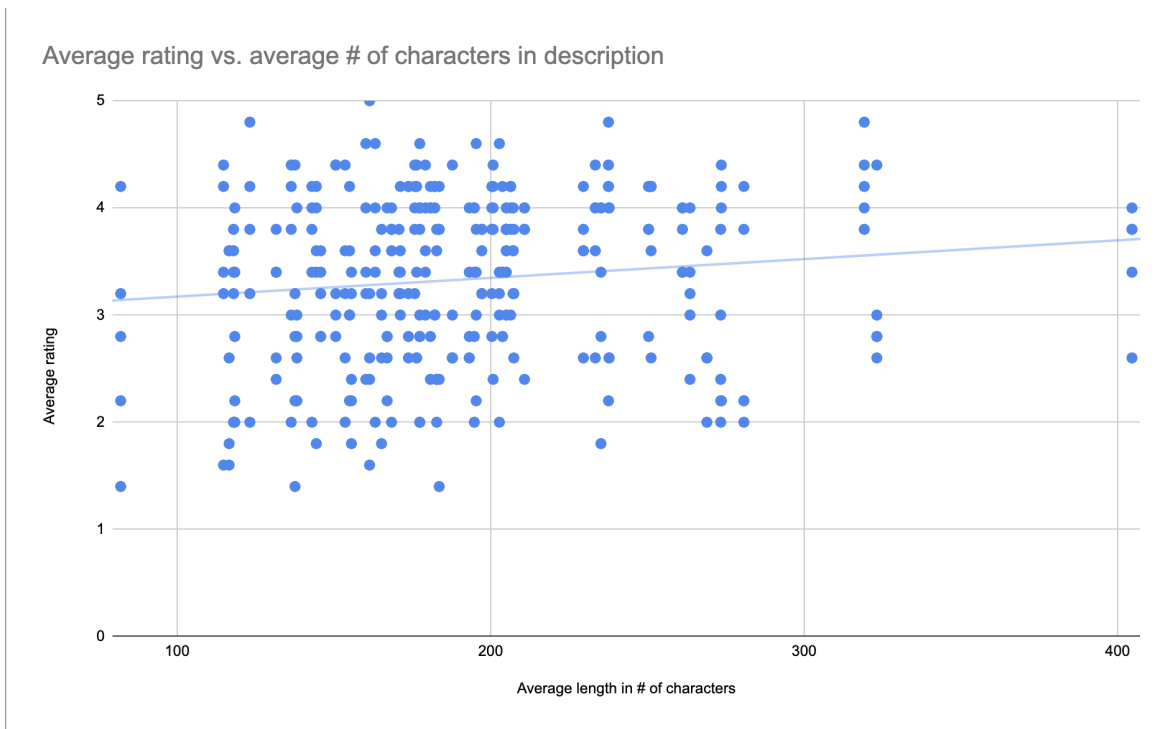


Figure 4.1: Scatterplot to compare ratings against length of descriptions

## 4.7 Discussion on Fine-Tuning Experiments

Overall, the results upon fine-tuning Curie with different sizes of training datasets, it can be concluded that the larger the dataset is, the more understanding the model will gain about the task and the higher its performance will be, more specifically the rooms, characters and objects training dataset was 500, 1500 and 3000 respectfully which clearly showed in the results, as well.

# Chapter 5

## Generating Item Attributes with Curie

### 5.1 Task

In this section of the thesis, we will focus on creating item attributes for the text-adventure games, by narrowing our focus on the objects dataset. We continue to use OpenAI’s GPT-3 for this part of the thesis, since a high level of language understanding is needed for the model to comprehend the name and description pairs (the input) and generate appropriate binary labels (the output).

### 5.2 Natural vs non-Natural Language

Another aspect for generating text was the aspect of prompt-engineering. First and foremost, the importance of prompt engineering was inspected, specifically the theory that OpenAI’s GPT-3 performs best when the prompt is formulated in natural language rather than in “computer language” due to the fact that the model itself was

trained on a massive amount of natural language. To demonstrate what this thesis considers as natural language vs computer language, the reader should consider the following: "This is not a weapon." vs "is\_weapon = False". Furthermore, the original dataset included attributes that were called `is_gettable` which is difficult for a language model generated with natural language to understand, since the words are not common or necessarily even exist in any other context in the English language.

## 5.3 Models

To generate binary attributes for rooms/categories, characters and objects, two separate models were called to test the difference between their abilities. These two were namely Curie and Davinci. Before diving into the details of the difference between their performances, there are already some aspects to consider, there clearly are advantages and disadvantages to both of these versions of GPT-3:

- Curie is a rather powerful and still very fast model. Even though its strength is not necessarily understanding complicated text, it should be perfectly capable of understanding the basic description of an object/character or room and assign an appropriate attribute to it, or better yet, decide if it has the specific attribute. Specifically, Curie is remarkably great at summarization which is why good results are expected from this engine.
- Davinci, however, is the most advanced and powerful model by OpenAI's GPT-3. This engine is a great choice for tasks and application that require a lot of understanding of the content. It is important to note that Davinci is an especially great option because it needs less instruction. However, some of the cons to consider are the facts that it is slower than Curie and also much more expensive. Since the experiment requires a lot of attribute generation, I had to

be mindful of the costs during the writing of the thesis such that the costs would not add up too much by calling the OpenAI API with the Davinci engine.

## 5.4 Attributes

To generate attributes, the first experiment that was conducted was to generate all attributes at once. These included for the object dataset:

- `is_verbatim`
- `is_drink`
- `is_food`
- `is_gettable`
- `is_plural`
- `is_surface`
- `is_weapon`
- `is_wearable`

However, the initial results were poor, in that after inputting few-shot examples into the OpenAI API call, the model did not succeed in generating all expected attributes in the expected format. Due to this, this thesis focused on generating item attributes one at a time and conduct a binary analysis upon generating these.

Next, it was important to change the attribute names into appropriate natural language texts, as per the reasoning in the section above. In order to fit this, the attributes were changed the following way, in the order of appearance above:

- This object contains other objects / This object does not contain other objects
- This is a drink / This is not a drink
- This is food / This is not food
- This object can be picked up / This object cannot be picked up
- There are multiples of this object / There is only one of this object.
- This is a surface / This is not a surface
- This is a weapon / This is not a weapon
- This can be worn / This cannot be worn

## 5.5 Objective Evaluation

Finally, the generated attributes were to be evaluated to analyze the engines' performance, as well as the effectiveness of the above mentioned prompt engineering.

For each attribute, few-shot learning was utilized when calling the OpenAI API with both the Davinci and Curie engines.

Upon generating attributes, true positives, false positives, true negatives and false negatives were added up individually. With the help of these numbers, the recall, precision and recall could easily be calculated the following way:

- Precision =  $\frac{TP}{TP+FP}$
- Recall =  $\frac{TP}{TP+FN}$
- Accuracy =  $\frac{TP+TN}{TP+FP+FN+TN}$

food			
metric	Curie	Davinci	Majority class baseline
precision	3.25	2.63	0.0
recall	80	80	N/A <sup>a</sup>
accuracy	48.28	26.96	41.6

Table 5.1: [Results of predicting the binary food attribute by Curie and Davinci]

<sup>a</sup>Since there were no true positive or false negatives, division by zero occurs when calculating recall

weapon			
metric	natural language	non-natural language	Majority class baseline
precision	23.26	16.47	0.0
recall	54.05	100	N/A <sup>a</sup>
accuracy	64.68	16.47	94.83

Table 5.2: [Results of predicting weapon attribute w/ natural vs non-natural language]

<sup>a</sup>Since there were no true positive or false negatives, division by zero occurs when calculating recall

where TP and TN stand for True Positive and True Negative, respectively, and FP and FN stand for False Positive and False Negative, respectively.

The difference between the two engines is clear after analyzing the results for the food attribute (Table 5.1). Davinci performs better in predicting item attributes. However, the difference might not be significant enough to counteract the fact that not only is Davinci slower, but much more expensive than Curie, as well.

Furthermore, it is very important to note the difference between results when the engine is given text in natural language vs in non-natural language. More specifically, numerically the difference in the case of the weapon attribute can be found in Table 5.2.

Obviously, the difference is very significant, highlighting the importance of using

natural language, which was conducted with the rest of the attributes in this thesis, as well. The reason why natural language performs worse in terms of recall (29 percent) compared to non-natural language (100 percent) is most likely due to the fact that the non-natural language model predicted weapon for most of the cases and therefore did not miss any objects that are truly weapons, whereas the natural language-trained model made more educated predictions.

## 5.6 Error Analysis

The first thing that stood out when analysing the results of food in Table 5.3, is that in both cases, the meat was classified as negative, as in non-food, when in fact according to commonsense, it is food.

However, after some further exploration, it became clearer what the problem may be: there are multiple different 'meat' objects in the dataset, with different descriptions. Therefore, even though there are some cases where the 'meat' object was classified correctly as food, in some cases, due to its description, both Davinci and Curie classified as negative.

Furthermore it is also interesting to point out that there are very few false negatives in the case of Davinci, in fact that was only 1 occurrence in around 300 examples. This could be due to the fact that the model tends to classify objects as food rather than non-food. This contradicts the fact that it is a skewed dataset, with a much larger percentage of non-food items than food.

However, for the sake of few-shot learning, both food and non-food items and their descriptions had to be included. The fact that the models were trained on a "balanced" training set for an imbalanced test set, could explain this phenomena. It can be observed from the results in Table 5.4 that when item attributes are generated

food		
classification	Curie	Davinci
true positive	<i>ingredient</i>	<i>vegetable</i>
true negative	<i>bin</i>	<i>Lamp</i>
false positive	<i>rack</i>	<i>statue</i>
false negative	<i>meat</i>	<i>meat</i>

Table 5.3: [Classification of food attribute by Curie and Davinci respectively]

weapon		
classification	natural language	non-natural language
true positive	<i>rock</i>	<i>rock</i>
true negative	<i>coin</i>	<i>N/A</i> <sup>a</sup>
false positive	<i>treasure chest</i>	<i>Lamp</i>
false negative	<i>fan</i>	<i>sword</i>

Table 5.4: [Classification of weapon attribute using natural vs non-natural language]

---

<sup>a</sup>there were no true negatives present

with a model that was given few-shot examples not in natural language, the model tends to only predict positive classification, in other words, in an overwhelming number of situations, it predicts that the object is a weapon. This is interesting, especially due to the fact that most objects are not a weapon, the object dataset is in fact an imbalanced dataset with most objects being non-weapons. However, the model is unable to gain that understanding because this information was not given to it in natural language and is therefore unable to deduce that information.

## 5.7 Final thoughts on generating attributes

As witnessed by the metrics above, it can be concluded that OpenAI’s GPT-3 though does not necessarily expect natural language, but since it was built with massive



amounts of natural language, it performs better and the one can unleash its biggest potential when the prompt is in natural language, as well.

Furthermore, as per my analysis, the difference between the performance of Curie and Davinci is not that significant to justify the added time and cost of using the superior engine, Davinci and therefore the thesis concludes that Curie is the best option for this task.

# Chapter 6

## Conclusion and Future Work

Since the publication of OpenAI's GPT-3, natural language processing has witnessed a revolution in the form of the renaissance of text generation. The technology has impressed and positively surprised the entire field and its vast array of applications are still being discovered, one being the fictional text generation, where text-based adventure games only cover a small portion of this genre.

This thesis explored some of the strategies to perfect prompt engineering as well as investigated the importance of fine-tuning and few-shot learning. Furthermore, OpenAI's engines were put to the test to understand the difference between their capabilities.

My main findings in this thesis include the following:

1. While Davinci is undoubtedly the most advanced and most powerful engine of OpenAI's GPT-3 model, with sufficient fine-tuning or few-shot learning, Curie could rise up to the level of Davinci for a specific task. Furthermore, Curie is much more cost-efficient which is an important factor to consider when working with large amounts of data. It is also important to mention that so far (as of December 2021), pre-training Davinci was not available through OpenAI's API.

2. Prompt engineering is crucial when using both fine-tuning and few-shot learning methods.
  - (a) In Chapter 4, the thesis explained how formatting the fine-tuning and few-shot learning examples was crucial in making sure that the model generates in the appropriate format, in the expected length and style.
  - (b) As demonstrated in Chapter 5, since GPT-3 was trained on massive amounts of natural language data, it is what the model expects and performs best with.
3. It also became clear through the error analysis in Chapter 5, that having a similarly balanced training and test set is beneficial in achieving the expected results. It is unrealistic to expect good results when training on a balanced dataset and testing on an imbalanced one.
4. As discussed at length in Chapter 2, creative fictional writing requires a deep understanding of the given world. This can be achieved by fine-tuning, due to which Fine-tuned Curie performed so well during description generation.

Some of the future work related to this thesis could explore the following:

1. In the case of description generation, there could be different scales for different attributes of each description during evaluation, such as: “description is stylistically very bad, bad, average, good or excellent”.
2. Furthermore, instead of only asking about the “quality” of the description in some abstract sense the annotators could be asked questions in the realm of:
  - (a) How accurate is this description?
  - (b) How close does it actually represent the real thing being described

- (c) How much does this description make you want to continue to play the game or ask more questions to the system?
- (d) How intriguing is this description?
- (e) How detailed is this description? (sometimes one option scored lower, because it messed up a detail but another option was given lower score because it didn't give any.)

Contrary to Phil Goetz's 1992 prediction, generating Text-based Adventure games is not only possible less than 30 years later, but flourishing.

This work demonstrated ways that could provide support to text adventure game developers, or even replace them in the future. Natural Language Processing and more importantly the branch of fictional text generation is developing and making strides faster than ever and it is my utmost pleasure that I could become a tiny part of this revolution.

# Chapter 7

## Acknowledgements

I would like to express my deep and sincere gratitude to Professor Christopher Callison-Burch for always going out of his way to provide support and providing crucial advice throughout the writing of this thesis. He has taught me the methodology to carry out the research needed for this thesis and to present the results as clearly as possible. It was a great privilege and honor to work and study under his guidance and to have learned invaluable lessons from him which I will carry throughout the rest of my professional career.

I would also like to thank Lara Martin, PhD, for supporting me through this long and challenging process. From her initial vote of confidence, to her insightful feedback on my writing, she has continued to stick by my side while giving me a level of encouragement I am doubtful I ever deserved. This project would not be what it is today without her support.

I want to extend a special thanks to Daphne Ippolito, who has been role model for me from the start. I may never have believed in myself without her voice of encouragement and I doubt this thesis would have been written without her reassurance that I should pursue my research goals.

I would also like to mention the members of Professor Chris Callison-Burch's lab for their many hours spent answering my questions and listening to me rant about my workload during Friday night lab dinners.

The students in the Artificial Intelligence course (CIS 521) also made this project possible with their contribution of completing subjective evaluations on Amazon Mechanical Turk in record time, in less than a day.

I would like to thank Professor Clayton Greenberg for granting me the opportunity to seek the Data Science degree which led me to writing this thesis and allowed me to deepen my knowledge and understanding in this most fascinating field.

Furthermore, I need to mention my amazing friend Meghana Chilukuri, without whose continual encouragement I would never have believed in myself enough to apply for the Data Science program or dared to go on the long and challenging road of completing a thesis.

Last but not least, I would like to thank my ex-colleague and long-time friend Gábor Kovács, without whose guidance I would not have survived the rigorous curriculum at Penn. I will forever be grateful for his countless hours of mentorship and supportive friendship.

You all are the reason I am now able to graduate with two degrees, a Bachelor of Arts in Mathematics and Computer Science as well as a Master of Science in Engineering in Data Science, all while completing the research required for this thesis.

# Bibliography

- [1] Prithviraj Ammanabrolu, Wesley Cheung, Dan Tu, William Broniec, and Mark Riedl. Bringing stories alive: Generating interactive fiction worlds. 2020. URL <https://arxiv.org/abs/2001.10161>.
- [2] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. 2020. URL <https://arxiv.org/abs/2005.14165>.
- [3] Marc-Alexandre Côté, Ákos Kádár, Xingdi Yuan, Ben Kybartas, Tavian Barnes, Emery Fine, James Moore, Matthew Hausknecht Ruo Yu Tao, Layla El Asri, Mahmoud Adada, Wendy Tay, and Adam Trischler. Textworld: A learning environment for text-based games. 2018. URL <https://arxiv.org/abs/1806.11532>.
- [4] Angela Fan, Jack Urbanek, Pratik Ringshia, Emily Dinan, Emma Qian, Sidharth Karamcheti, Shrimai Prabhumoye, Douwe Kiela, Tim Rocktaschel, Arthur

- Szlam, and Jason Weston. Generating interactive worlds with text. 2019. URL <https://arxiv.org/abs/1911.09194>.
- [5] Nancy Fulda, Daniel Ricks, Ben Murdoch, and David Wingate. What can you do with a rock? affordance extraction via word embeddings. 2017. URL <https://arxiv.org/abs/1703.03429>.
- [6] Matthew Hausknecht, Prithviraj Ammanabrolu, Marc-Alexandre Côté, and Xingdi Yuan. Interactive fiction games: A colossal adventure. 2019. URL <https://arxiv.org/abs/1909.05398>.
- [7] Xiao Liu, Yanan Zheng, Zhengxiao Du, Yujie Qian, Zhilin Yang, and Jie Tang. GPT Understands, Too. 2021. URL <https://arxiv.org/abs/2103.10385>.
- [8] Robert Logan, Ivana Balazevic, Eric Wallace, Fabio Petroni, Sameer Singh, and Sebastian Riedel. Cutting down on prompts and parameters: Simple few-shot learning with language models. 2021. URL <https://arxiv.org/abs/2106.13353>.
- [9] Shahbuland Matiana, JR Smith, Ryan Teehan, Louis Castricato, Stella Biderman, Leo Gao, and Spencer Frazier. Cut the CARP: Fishing for zero-shot story evaluation. 2021. URL <https://arxiv.org/abs/2110.03111>.
- [10] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. Learning transferable visual models from natural language supervision. 2021. URL <https://arxiv.org/abs/2103.00020>.
- [11] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. Ex-



ploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, (21), 2020. URL <https://www.jmlr.org/papers/volume21/20-074/20-074.pdf>.

[12] Jack Urbanek, Angela Fan, Siddharth Karamcheti, Saachi Jain, Samuel Humeau, Emily Dinan, Tim Rocktäschel, Douwe Kiela, Arthur Szlam, and Jason Weston. Learning to speak and act in a fantasy text adventure games. page 673–683, 2019. URL <https://aclanthology.org/D19-1062>.

[13] Ningyu Zhang, Luoqiu Li, Xiang Chen, Shumin Deng, Zhen Bi, Chuanqi Tan, Fei Huang, and Huajun Chen. Differentiable prompt makes pre-trained language models better few-shot learners. 2021. URL <https://arxiv.org/abs/2108.13161>.