Examining Descriptiveness of Narratives Generated using Planning and Large Language Models

An Honors Thesis by Courtney Dozier

Committee Members

Director: Dr. Maria Elena Chavez-Echeagaray
Second Reader: Dr. Victor Benjamin

Barrett, the Honors College
Arizona State University
Fall 2022

Abstract

Narrative generation is an important field due to the high demand for stories in video game design and also in stories used in learning tools in the classroom. As these stories should contain depth, it is desired for these stories to ideally be more descriptive. There are tools that help with the creation of these stories, such as planning, which requires a domain as input, or GPT-3, which requires an input prompt to generate the stories. However, other aspects to consider are the coherence and variation of stories. To save time and effort and create multiple possible stories, we combined both planning and the Large Language Model (LLM) GPT-3 similar to how they were used in TattleTale [25] to generate such stories while examining whether descriptive input prompts to GPT-3 affect the outputted stories. The stories generated are readable to the general public and overall, the prompts do not consistently affect descriptiveness of outputs across all stories tested. For this work, three stories with three variants each were created and tested for descriptiveness. To do so, adjectives, adverbs, prepositional phrases, and suboordinating conjunctions were counted using Natural Language Processing (NLP) tool spaCy for Part Of Speech (POS) tagging. This work has shown that descriptiveness is highly correlated with the amount of words in the story in general, so running GPT-3 to obtain longer stories is a feasible option to consider in order to obtain more descriptive stories. The limitations of GPT-3 have an impact on the descriptiveness of resulting stories due to GPT-3's inconsistency and transformer architecture, and other methods of narrative generation such as simple planning could be more useful.

Introduction

There is a heavy toll on everyone who needs to write a story. Stories require a lot of creativity, thought, and time that could be spent on other endeavors. Teachers that need to create lesson plans to teach students how to read and understand stories could use all new fresh, exciting stories. Teachers could use more stories using modern speech rather than give students old material with words that are hardly ever used nowadays.

In a different realm, video game designers that want to create detailed, immersive games need good stories to ensure their audience is engaged and entertained. By creating dramatic cut scenes with intense dialogue, detailed backstories, and the like, the entire playing experience seems more realistic and well-thought out, which is essential to creating a well-received game. Plus, designers of games that have story branches where characters' actions influence the resulting story could use not just several stories, but several stories based on the same material but varying slightly.

To create all of these stories, especially from scratch, is a monolithic task. Not only that, but generating enough of these stories or variants of these stories would be exponential. To cover this, we need not only a way to save time in story generation, but also a way to create multitudes of stories and variants on them.

That is why generating such stories with tools like planning [10] can be useful, especially if the stories could become automatically generated. The field of planning focuses on finding a sequence of actions (called a plan) that can be done to achieve certain goals given a particular domain. This field happens to be fit for generating stories because of how coherent and believable the plans, and thus the plot of the story, can be. By constraining when certain actions can happen, coherence can be forced in a way that other generation methods fail to do.

There has been a lot of research into the field of planning and all that

it can be used for. Understanding how diverse planning can be enables us to learn from other unrelated problems and design the planning algorithm with multiple resulting fields in mind, in our case video games and education. Planning is also extremely useful for narrative generation [11], especially for constructing believable stories, which we attempt to do in this work. It is important to keep in mind not just story believability but also character believability as well. We also must consider the runtime and possible shortcuts for such planning algorithms to ensure that generating the stories does not take away from planning's time-saving capabilities. To further save time, automating the generation of stories from planning and enabling the generation of any kind of story we want in one central algorithm (domain-independent planning) is ideal. Additionally helpful is noting how far LLMs have come in narrative generation on their own, including procedural generation, while noting their limitations. The new challenge then lies in how to expand on the language of the stories without losing the coherence generated by the planner. In this work, we seek to change the input prompt's descriptiveness and determine whether or not GPT-3's output will be more or less descriptive. While the descriptive measures have shown some possible insight into how descriptiveness can be increased, overall, this work has been another demonstration of the limitations of GPT-3 as a whole.

Literature Review

Currently, there has been a lot of ongoing research into how planning, especially with Planning Domain Dependent Language (PDDL), can be used to generate narratives. There has also been some research into how Large Language Models (LLMs) like GPT-3 by OpenAI [1], which takes an input prompt and continues where the prompt leaves off, can fare in generating stories. Stories generated with planning tend to be more coherent and believable, but need to be somehow translated into natural language. Stories generated with LLMs are already in natural language, but can be repetitive or incoherent. In fact, lack of coherency is one of GPT-3's worst downsides.

One work called TattleTale combines both planning and GPT-3, hoping to take the strengths of both to cover their weaknesses. Nisha Simon and Christian Muise first create a plan of story actions based on childrens' stories, then feed that sequence of actions to GPT-3 line by line with the prompt growing each time, appending the result in an outputted story [25]. This work was successful in generating coherent and interesting stories; however, the stories TattleTale generates fall short in descriptiveness as the style of writing seems plain.

Breadth of Planning

The domain of planning can be used for many different tasks. Masataro Asai and Alex Fukunaga created an architecture called LatPlan that takes image input, autoencodes the input, uses deep learning to determine the PDDL model, and solves the image using planning [2]. This means that a planner is then able to solve an 8-puzzle image by first going through that deep learning step. If a narrative was to be generated solely from images, with an initial image and a goal image, then perhaps we could create a story of images with a similar technique beyond just text. Then narrative generation could extend much more easily beyond text-RPGs, and perhaps some form of rudimentary

animation could be done using LatPlan and planners. However, this paper focuses solely on text-based narratives for proof of concept of descriptiveness.

In addition, narrative generation can extend to many different fields. For instance, with methods like those used by Justus Robertson and R. Michael Young [23], we can extend the narratives generated to actual video games so as to personalize story experiences. It also can extend to reporting disasters on the news as with the work of Tang et al. [26].

In both cases, it is important to consider the audience of such stories as the stories we generate should be easily accessible to that audience. As a result, in this work we looked into readability of the generated stories to determine what grade level the stories would be accessible for in the classroom.

Planning-Generated Narrative Believability

Another important factor beyond considering the potential diversity of the audience of the stories is the believability of said stories. The goal is to construct stories that have coherent plots where characters act in a way that is not necessarily expected, but definitely believable. Taking believability into account will ensure that the stories produced will not seem like nonsense to the potential readers.

In a paper about NPCs in video games, Johansen et al. consider how planning can enhance believability in NPCs and create a set of planning problems for each NPC [11]. Their work suggests that if we split a story into multiple planning problems, we can improve how believable each character's actions are relative to their goals. Given how each character can do one action per turn, a story could be created based on these individual goals. The story would also need to consider the author's goals, as does a planning algorithm called the Glaive Narrative Planner. Glaive not only involves multiple characters having their own individual goals but also nudges the group towards the author's goals [27]. If character and author goals could be created and planning code written for them, and then the resulting plan fed into an LLM, this would improve the story quality. The LLM would not just have better coherence but also better believability, although the complexity of computing the story may increase.

Another multi-agent planner called Sabre works similar to Glaive in that it involves multiple characters and achieving a main author goal, but instead of having character goals ensures that characters do not act against their beliefs [28].

Compared to Glaive, instead of writing character goals, we would need to write constraints on what characters would be willing to do. This would also involve more complexity than single agent planning, but feeding it to an LLM will also improve the LLM's character believability in a different sense. Given more time and computational resources, the above two methods for believability could be implemented within the planning domain that was written for this work. However, for simplicity and to ensure the focus was on descriptiveness, these believability techniques were omitted.

Another work involving character belief accounts for characters not knowing certain aspects about the world [4]. This is actually a really useful way to interpret stories, especially ones that involve characters being in the dark. One such story was analyzed in this paper [17].

A work by Marrella et al. introduces mood-based planning in *The Iliad* by forcing characters to only perform certain actions if their mood towards a certain character suits the action [15]. Adding mood would also increase LLM character believability although we would once again suffer from complexity as it requires complicating the original PDDL problem. Moods would also be a wonderful addition to the descriptive stories generated in this work in further research.

Runtime of Planning for Narrative Generation

As the complexity increases, one method of ameliorating the complexity of the search space is to use Monte Carlo Tree Search as it improves performance when planning [12]. An implementation with Monte Carlo Tree Search would actually be useful to limit the amount of time spent planning as then more time can be freed for the LLM to write the story based on that plan. Another way of simplifying planning problems is to focus only on protagonist and antagonist roles, which according to a user study still left the story clear to readers [19].

Porteus et al. use state constraints to both limit how many states have to be searched through to find the story and also to enforce more control over the narrative [18]. This would also reduce the complexity needed to plan. If more stories are generated using LLMs, implementing a state constraint control to ensure action sequences are exactly what is desired for the story would be extremely useful.

All of these are additional avenues for further research. As the runtime of this project was minimized by using small domains, these techniques were

not fully implemented, but would be very helpful for more complex stories.

Extending and Varying Planning-Based Stories

For extending planning methods to create more domain independent stories, tools like THYPE are useful. THYPE abstracts characters, objects, locations, etc. to general types and provides alternatives to specific instances of said types [21]. This will be useful as researchers seek to automate PDDL planner-based narrative generation, and would enable more stories to be generated with more variation. In this work, we focused on simple variations, but THYPE would open up even more possibilities.

Another idea for creating stories that are domain independent is to use a natural language model to build the planning problem PDDL automatically. This can be done using RNNs as in Christian Muise and Nisha Simon's work [24]. It can also be done by merely writing a plot synopsis and generating the PDDL for that problem automatically [10]. Creating these models automatically would drastically improve their extension in the real world as more people will be able to use these generated stories in more realms at ease.

We also may seek to automate the entire process, as done in a work by Richard A George where soap operas were entirely automatically generated [8]. In this work, we did not automate the initial manual extraction to ensure that the extraction is accurate; however, given more methods to do so, the entire process could easily become automated.

Another issue in narrative story generation is creating variation in stories, to which Porteus et al. create variants based on antonyms for particular actions [20]. Similar to how THYPE created protagonists and antagonists, creating antonyms of actions would vary up the story greatly. Having more variants enables more stories to be generated, which will also help the ease of using narratives in video games or other realms like classrooms. In this work we did not include antonyms for actions, but we did vary where characters start and end.

Story Generation with LLMs

There has also been work with LLMs like GPT-3 to create narratives procedurally, making branches for different variations [7]. While this is useful, it may not always have the structure and coherence that planning-generated

stories do.

Another important note about LLMs is that they can tend toward representing the bias that their original data was trained on. For instance, a study on gender representation bias revealed that even verbs that portrayed power could be associated with masculinity [14]. As such, any narrative generation using LLMs needs to be curated with bias in mind, and GPT-3 may still require some additional constraining in order to prevent it from revealing biased opinions. To ameliorate this, in this work we solely asked for one line from GPT-3 at a time so that it did not go off on tangents. Unfortunately, there may still be more bias that ideally should be curated afterward in post-processing.

Another issue with GPT-3 is that it has a transformer architecture. For Recurrent Neural Networks (RNNs), there exists a problem called the *vanishing gradient effect*. This is where along the line of backpropagation, the values become smaller and smaller, causing the gradient to "vanish" further on out [3]. The transformer architecture, used by GPT-3, avoids this issue by keeping all of the history in mind to be able to read longer input prompt texts [16]. However, this causes GPT-3 to find every detail included in the prompt important, which could affect the results of any generated story, in this work or otherwise.

Combining Planning and LLMs

Christian Muise and Nisha Simon explored feeding planner-generated narratives into GPT-3 in their work TattleTale [25]. To limit repetitiveness and ensure coherence, they limited GPT-3 to only output a single line of text for each action. They also fed GPT-3 the context of the story to begin with so as to help GPT-3 understand who the characters were and why they were attempting to reach certain goals. Their solution did, in fact, produce coherent stories. However, these stories lack elaborate detail. This paper sought to explore how to expand on such coherent GPT-3 stories' descriptiveness without losing coherency.

Approach

The majority of the approach of this work is similar to the method used in Muise and Simon's TattleTale [25], with minor modifications. In this work, the prompts fed to GPT-3 vary in their descriptiveness. In addition, these prompts were written by asking GPT-3 to paraphrase a given action in the story instead of add on to the story as in TattleTale. Additionally, TattleTale fed the output of GPT-3 as the next line's input, which was not done in this work either so as to avoid GPT-3 running off with the story in a different direction. The method of generating stories in this work involves the following:

1. Developing a plan based on a given reference story
2. Converting this plan into input to be fed to GPT-3
3. Capturing the output from GPT-3
4. Analyzing the resulting descriptiveness

Developing a Plan

In order to develop a plan of actions to be done throughout the story, first the actions, characters, items, and locations of the story need to be known. For simplicity, we chose to manually extract these objects and actions from children's stories as they tend to have simple settings and thus few actions, characters, items and locations. This then ensured that the search space for the planner is small enough that plans for each story could be found in a quick amount of time. The stories chosen were "Rapunzel" [9], "The Golden Nugget" [17], and "The Heart of a Monkey" [13]. These were chosen from European, Asian, and African folktales to ensure that the results hold regardless of the origin and culture of the story.

Object Category	Objects Extracted from Story
Type	animal, location
Characters	monkey, shark, sultan, lion, donkey, hare
Locations	cliff, cave, tree, sea, forest, stream
Actions	move, trick, travel together, eat heart

Table 1: Extracting objects from the reference story resulted in these lists, that were compiled into PDDL.

After choosing the stories, to manually extract the objects and actions, we looked for each of these in the story and noted down every new one we saw. Technically there are ways to automatically extract named entities like characters using natural language processing. However, to ensure that no character was missed, we opted for manual extraction. An example of the individual items obtained from manual extraction is in Table 1.

Predicates
connected ?l1 ?l2 - location
in ?a - animal ?l - location
sick ?a
travelling_together ?a1 ?a2 - animal
foolish ?a
tricked_once ?a
heart_eater ?a
want_heart ?a

Table 2: Extracting predicates from the reference story resulted in this PDDL syntax-based list. Each predicate begins with its name followed by its parameters (like ?a) with their parameter types (like animal).

Once the extraction had been done and actions had been determined and defined, we still needed a way to determine a plan based on all of these objects and actions we had identified. We did so by writing these actions, characters, objects, and locations in Planning Domain Definition Language (PDDL). PDDL takes a *domain*, which is the kind of world that the characters live in and how that world works, and a *problem*, which defines the specific names of objects, where characters start at, and where they should end at. Then, a planner could take the PDDL description and solve the problem, finding a

```
(:action trick
    :parameters (?a1 ?a2 - animal)
    :precondition (and
        (exists (?l - location)
                (and (in ?a1 ?l) (in ?a2 ?l)))
        (or (foolish ?a2) (not (tricked_once ?a2)))
        (not (= ?a1 ?a2))
    )
    :effect (and (trust ?a2 ?a1)
        (when (not (tricked_once ?a2))
            (tricked_once ?a2))
        (when (tricked_once ?a2)
            (tricked_twice ?a2))
    )
)
```

Listing 1: PDDL trick action. Given an animal ?a1 that wants to trick another animal ?a2, the animals must be distinct, must be in the same location, and the second animal must either never have been tricked or be foolish enough to be tricked twice. The result is that the second animal trusts the first, and will be tricked once if never tricked or else tricked twice if tricked once. Being tricked more than twice is ignored for the scope of the story.

plan of actions that enables the characters to get to the end.

Once we had manually extracted everything, we could write the actions in the domain file and the objects in the problem file. The actions needed a bit of work to determine what they apply to (parameters), when they can be applied (preconditions), and what happens after they are applied (effects). To do so, we had to create *predicates*, which are certain facts about certain objects in the story that can be true or false. For instance, whether a character is at a location is a predicate that was created for each story. Once we had defined these predicates as in Table 2, we could create the actions by noting when in the story the actions could be applied and using common sense. In Listing 1 is a snippet of the PDDL code for the action for an animal to trick another.

From there, all that was needed to form a plan is where the objects start, and where they should end. Where the objects start is defined in the *initial*

```
(:init
    (in the_monkey tree)
    (in the_shark sea)
    (in the_sultan sea)
    (in the_lion cave)
    (in the_hare cave)
    (in the_donkey forest)
    (sick the_sultan)
    (sick the_lion)
    (connected tree cliff)
    (connected cave forest)
    (connected forest stream)
    (connected cliff sea)
    (connected sea cliff)
    (connected stream forest)
    (connected forest cave)
    (connected cliff tree)
    (foolish the_donkey)
    (want_heart the_hare)
)
```

Listing 2: PDDL initial state for "The Heart of a Monkey". This includes all of the predicates that are true at the beginning of the story. In this case, it includes the current locations of all of the characters, what locations are connected, and the fact that the donkey is foolish and the hare wants the heart for itself

state, and where the objects end is defined in the *goal state* of the problem file. To determine the initial state, we once again examined the reference stories, looking at the setting to determine where characters and objects were and what they had at the start. For the goal state, we examined the ending and similarly determined the locations of characters and objects and what they had at the end. However, we also opted to create different variations on each of the reference stories. This was done so as to have more generated stories by the end, a method that teachers may use if they were to generate stories in this manner. In addition, having these variants would also show that descriptiveness of output does not depend on the story ending. An example

```
:goal (and
    (tricked_twice the_donkey)
    (tricked_once the_monkey)
    (not (heart_eater the_lion))
    (not (heart_eater the_sultan))
    (heart_eater the_hare)
))
)
```

Listing 3: PDDL goal state for "The Heart of a Monkey". This includes all of the predicates that are true at the end of the story. In this variant, the donkey should be tricked twice, the monkey tricked once, and the hare will end up eating a heart, but the lion and sultan will not.

of these initial and goal states for a particular story variant is provided in Listings 2 and 3 respectively.

Now that we had created the initial and goal state, we had constructed the entire problem and domain file, and we could solve these using a PDDL planner, and we opted to use the solver from planning.domains [5]. From there, the planner generated a solution as a sequence of actions that characters would perform, and thus we had generated the plan for the story. The example plan is presented in Listing 4.

```
(move the_hare cave forest)
(trick the_hare the_donkey)
(trick the_hare the_donkey)
(eat_heart the_hare the_donkey)
(move the_sultan sea cliff)
(move the_sultan cliff tree)
(trick the_sultan the_monkey)
```

Listing 4: Plan returned by PDDL solver of the actions to be taken for the story to progress from start to finish.

Converting the Plan to GPT-3 Input

Once we had generated a plan, the next step was to generate the input for GPT-3. First, we created a prompt that would give GPT-3 the context about the way the story world works and the setting and goals that are sought to be achieved. In these prompts, we then created three separate versions: one with simple, short language, one with a few more adjectives and adverbs, and one with even more that is even more descriptive. We could then test for descriptiveness by running each different prompt version and comparing the outputs.

Action	**Sentence example**
(move ?a - animal ?l1 ?l2 - location)	The lion moves from the cave to the forest, accompanied by anyone travelling with the lion.
(trick ?a1 ?a2 - animal)	The hare tricks the donkey.
(move_together ?a1 ?a2 - animal)	The shark and monkey start travelling together.
(eat_heart ?a1 ?a2 - animal)	The lion eats the donkey's heart since the donkey was tricked twice.

Table 3: A series of examples of sentences that each action would be converted to for "The Heart of a Monkey".

After writing the prompts, we then took each action in the plan that was generated and wrote it in natural language. To do so, we took each action in the story's domain file and wrote out a natural language sentence corresponding to that action. In Table 3, an example converted sentence is listed for every action. Then, we created a python program to automatically take the actions in the plan and write out the sentence for each action based on who does the action.

The next step was to feed the prompt as well as the action sentences to GPT-3. In order to curate GPT-3's output and prevent it from repeating itself or going off on tangents, we fed the input as the prompt plus the first action in the plan. Then we looked at the prompt plus the second action and ask GPT-3 for output to append to its previous output, and so on throughout the remaining actions. However, unlike Muise and Simon, instead of feeding the action sentence verbatim, we found it more accurate to ask GPT-3 to

rephrase the sentence. This way, we avoided GPT-3 writing what happens after that action and instead managed to get GPT-3 to write the story in its own words, while still following the action that was given.

```
Here is the context for the story:

Ki wu and Pao shu are two young men who love to work
hard. Ki wu and Pao shu are very good friends and many
people talk of their bounteous virtue. One day Ki wu
and Pao shu are tired after working a long time. The
pathway happens to reward those who are virtuous with
precious golden nuggets and those who are not with
deadly, dangerous snakes. There is also a countryman
who is not virtuous lounging lazily at the spring. To
get to the spring from the city, you have to first go
to the grove, then to the stream, then to the pathway,
and then you arrive at the spring. To be able to rest,
there should be no one else resting there. Ki wu and
will end up losing his virtuousness eventually. Pao shu
and Ki wu will surely end up at the grove. The
countryman will no longer be resting after a long nap
and will end up travelling to the city.

Here is the next action in the story:
The countryman walks from the grove to the city, and all
friends of the countryman that are awake nearby follow.

Paraphrase that action in different, more elaborate words:
```

Listing 5: Example prompt for an action of "The Golden Nugget" story variant 3 prompt2. As can be seem, there is a moderate amount of adjectives and adverbs added for this prompt. The context is first provided, then the next action, and then a prompt for GPT-3 to rephrase the action in different words.

Capturing GPT-3 Output

Now that the plan had been determined and the input prompts had been written, we sent the output to GPT-3 to rephrase the story in its own words. The model used was TEXT-DAVINCI-002 as it is declared by OpenAI to be "the most capable model in the GPT-3 series" [1]. The temperature hyper-parameter controls how likely GPT-3 will output text that is riskier. The temperature was set to 0.7 to avoid too much repetition and leave room for descriptiveness by taking more risks. The maximum number of tokens was left at 256 to allow for longer sentences, but of course most, if not all, of the sentences generated are shorter than that. The rest of the parameters were left to default as modifying them too many of the parameters can cause the resulting output text to, once again, be either extremely repetitive or completely ignore the input prompt.

Feeding the input prompt to OpenAI's software required a simple python program (see Appendix E: Main Python Program) to generate *completions* of each prompt given. Every action required a completion of the given prompt, and the results from each completion were stored in a plain text file of the final story. An example of one such story is provided in Appendix A: Example Story Output at the end of this paper.

Descriptiveness Analysis

Lastly, in order to determine whether more descriptive prompts caused more descriptive output, we needed a measure of descriptiveness. As adjectives, adverbs, prepositional phrases, and subordinating clauses are a good hint as to how descriptive a sentence is, we measured how many were in both the input prompts and output using spaCy's POS tagging feature. spaCy is a Natural Language Processing (NLP) tool that can label every word in a given text with a part of speech (POS) tag [6]. For spaCy, adjectives are represented by tags JJ, JJR, and JJS; adverbs are represented by tags RB, RBR, and RBS. We noted the count of both prepositions and subjunctive clauses using the tag IN.

We also measured the total number of sentences, the total number of words and the average number of words per sentence as a secondary measure of descriptiveness. Each of these descriptiveness measures was outputted as in Listing 6.

As some prompts may be longer than others, and that may influence descriptiveness, we also measured the total number of words in the prompts

```
Stats for "./rapunzel1/prompts/rapunzel1.txt":
Number of sentences: 7
Number of tokens: 80
Average number of tokens per sentence 11.43
Adjectives:                                  2
Adverbs:                                     2
Prepositions/Subordinating Conjuctions:      8
Total:                                      12
```

Listing 6: Output for descriptive stats on an input prompt for the first variant of "Rapunzel."

and output. Afterward, we divided the total of the descriptive words by the number of words to get a measure not reliant on text length.

Combining Steps with an Automated Process

Now that stories could be generated and analyzed, a final step was taken in this work to automate the process from different variants of Planning Domain Definition Language files to the generated plan to GPT-3 and finally to the descriptiveness analysis. Each variant of each story was run through five iterations of GPT-3 stories to create five final stories that could be analyzed and the results averaged. This was done through a simple bash shell script (see Appendix D: Bash Shell Script) that stored all of the resulting stories in an output folder and all of the descriptiveness results in another folder for easy processing. The script could easily be extended to other shell types to enable generating these stories from any operating system that supports python.

Results

The main results of this study are the resulting descriptiveness of the GPT-3 outputs. Recall that descriptiveness reflects how detailed a story is and is measured as the amount of descriptive words like adjectives, adverbs, and prepositional phrases. However, as coherence in a story could be defined not solely just on the plot consistency, but also on the readability, we take a look at how one of the outputted stories fares in terms of readability. Readability measures how easy a text is to read, and is computed based on spelling and grammar mistakes as well as length of words and sentences. The next section covers those readability results, and the following section covers the actual descriptiveness results.

Readability

As the goal of this study was to maintain coherence in the stories, the output for the first "Rapunzel" variant when given PROMPT1 as input was analyzed using readability software [22] in Figure 1. As can be seen, there were two long sentences and an adverb, which influenced the readability score, but otherwise the story was coherent. As longer sentences and more adverbs correspond to more descriptive stories, this particular readability measure will end up lower for more descriptive stories. As descriptiveness is analyzed in the next section, it is important to keep in mind that the readability is reduced for more descriptive stories.

As can be seen in Figure 2, there are multiple readability and reading level scores. The best case is that fifth graders could read the story, and the worst case is closer to ninth grade. Either way, the stories and prompts would need to be adapted further to ensure readability for younger students. Still, the reading levels were high enough to use these generated stories in higher education.

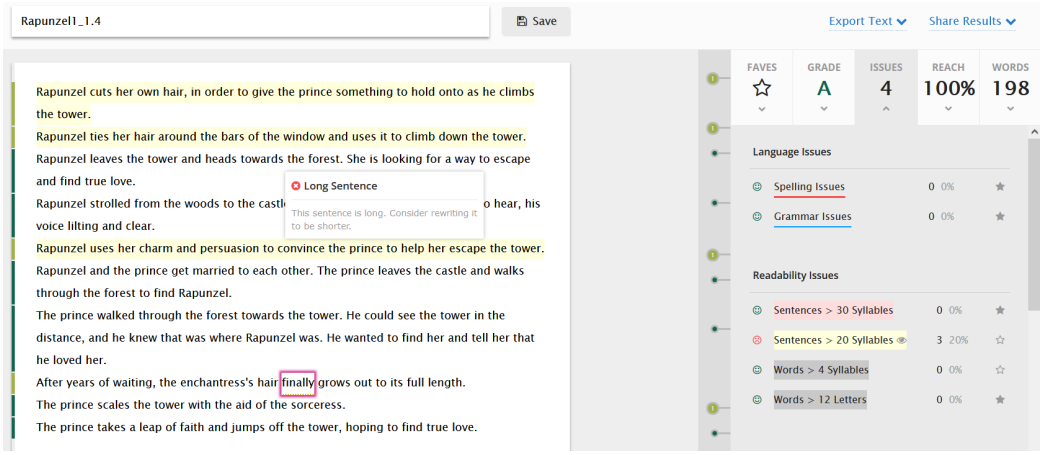

Figure 1: The readability scoring on Readable.com for the fourth iteration of the first "Rapunzel" variant and prompt1. Adverbs count against readability as do long sentences; however, the overall readability grade is an A.

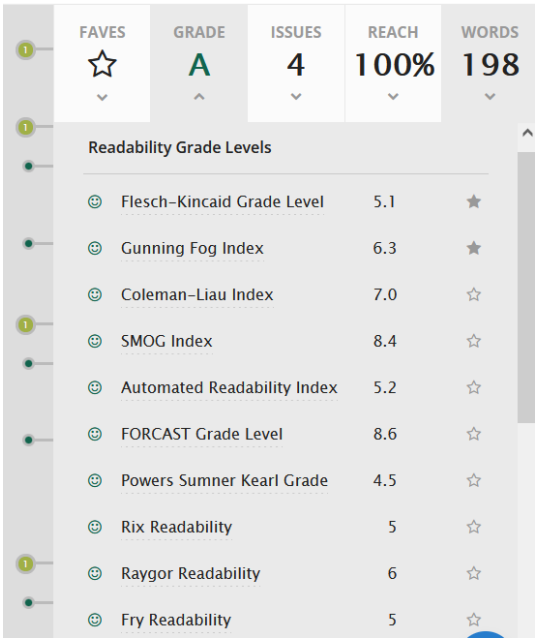

Figure 2: The reading grade level scores for the fourth iteration of the first "Rapunzel" variant and PROMPT1 on Readable.com.

Descriptiveness

In addition to readability, descriptiveness was an important factor studied in generated stories in this work. However, the results for descriptiveness were gathered for all story variants and prompts, unlike for readability, as descriptiveness was the main focus of this work.

After the GPT-3 output was created and the descriptiveness analyzed, the resulting descriptiveness measures were gathered for each prompt of each story variant. The story variants are labeled with a short tag for simplicity: RAP1, RAP2, and RAP3 for the "Rapunzel" variants; GOLD1, GOLD2, and GOLD3 for "The Golden Nugget" variants; and MONK1, MONK2, MONK3 for "The Heart of a Monkey" variants. The prompts were labeled PROMPT1, PROMPT2, and PROMPT3, where PROMPT1 was the least descriptive, PROMPT2 was of medium descriptiveness, and PROMPT3 was the most descriptive.

Subsequently, all of these variants' descriptiveness measures were gathered for each of the five iterations of GPT-3 output stories. These were separated by their prompt and story variant as well. Each of these iterations was then averaged for all of the descriptiveness values. The complete raw data is listed in Appendix B, with data for averages listed in Appendix C.

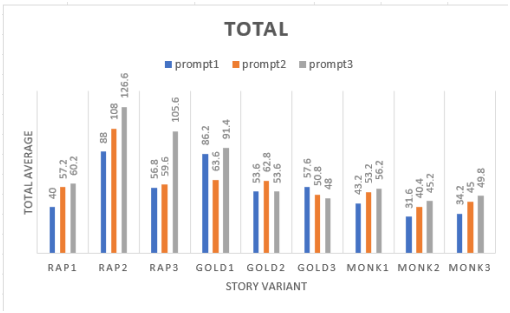

Figure 3: Given the story variant (X-axis) and prompt number (key), a plot of the values of total descriptiveness is shown. These values are taken as averages of every iteration of running GPT-3's story outputs.

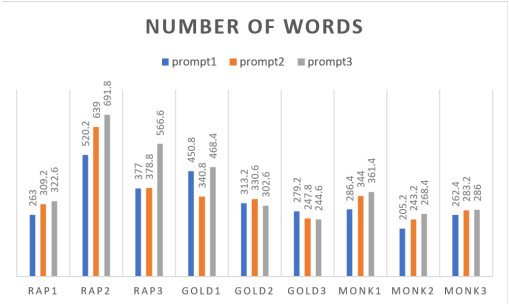

Figure 4: Given the story variant (X-axis) and prompt number (key), a plot of the number of words is shown. These values are taken as averages of every iteration of running GPT-3's story outputs.

total	**prompt1**	**prompt2**	**prompt3**
RAP1	40	57.2	60.2
RAP2	88	108	126.6
RAP3	56.8	59.6	105.6
GOLD1	86.2	63.6	91.4
GOLD2	53.6	62.8	53.6
GOLD3	57.6	50.8	48
MONK1	43.2	53.2	56.2
MONK2	31.6	40.4	45.2
MONK3	34.2	45	49.8

Table 4: Given the story variant (left column) and prompt number, each of the values of total descriptiveness is shown. These values are taken as averages of every iteration of running GPT-3's story outputs.

#words	**prompt1**	**prompt2**	**prompt3**
RAP1	263	309.2	322.6
RAP2	520.2	639	691.8
RAP3	377	378.8	566.6
GOLD1	450.8	340.8	468.4
GOLD2	313.2	330.6	302.6
GOLD3	279.2	247.8	244.6
MONK1	286.4	344	361.4
MONK2	205.2	243.2	268.4
MONK3	262.4	283.2	286

Table 5: Given the story variant (left column) and prompt number, the number of words in the text is shown. These values are taken as averages of every iteration of running GPT-3's story outputs.

Adding more adjectives, adverbs, prepositional phrases, and subordinating clauses in the prompts did result in more of these in the output for each of the "Rapunzel" and "The Heart of a Monkey" story variants. However, this same result did not appear in any of the variants of "The Golden Nugget." The differences in Tables 4 and 5 and Figures 3 and 4 (see above) show

that the total number of these descriptive items was affected by the total number of words outputted by GPT-3. The graphs demonstrate very similar trends, showing this correlation between descriptive words and total number of words. This is an important result as it means that if we want a descriptive story, one way to obtain one is to run GPT-3 and only take the longest outputs.

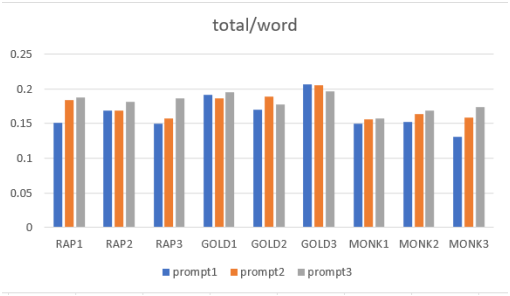

Figure 5: Given the story variant (X-axis) and prompt number (key), a plot of the ratios of total descriptiveness per number of words is shown. These values are taken as averages of every iteration of running GPT-3's story outputs.

total/#words	**prompt1**	**prompt2**	**prompt3**
RAP1	0.151	0.183	0.188
RAP2	0.169	0.169	0.181
RAP3	0.15	0.157	0.186
GOLD1	0.191	0.186	0.195
GOLD2	0.17	0.189	0.177
GOLD3	0.206	0.205	0.197
MONK1	0.15	0.155	0.157
MONK2	0.153	0.163	0.169
MONK3	0.131	0.158	0.174

Table 6: Given the story variant (left column) and prompt number, the total descriptiveness per number of words ratio in the text is shown. These values are taken as averages of every iteration of running GPT-3's story outputs.

However, if we take the number of words into account, we can find a better-adjusted measure of the total descriptors. To do so, we divided the number of total descriptors by the number of words, forming a ratio. As can be seen in Figure 5 and Table 6 (see above), the values obviously followed very similar trends to the ones for the total descriptiveness. However, we can see how slight the differences were between prompts. Overall, the ratios did, in fact, increase across prompts for the "Rapunzel" and "The Heart of a Monkey" variants, but had mixed results on "The Golden Nugget" story variants.

The reason for this discrepancy may involve the specific quantities of descriptive words increased between prompts, as these were not explicitly controlled to be a specific number across all stories. Given time for another experiment that would control that factor, perhaps the results would be more consistent. As can be seen, in Figure 6, the number of adjectives added was less between PROMPT1 and PROMPT2 for "The Golden Nugget" story

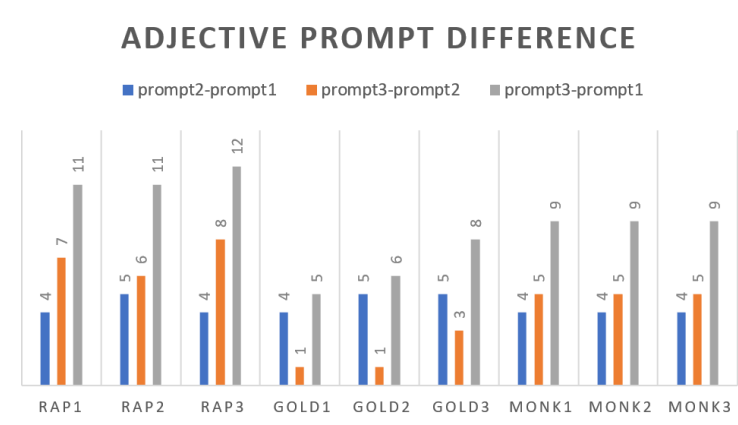

Figure 6: Differences of adjectives added between prompts for every story variant.

variants. This may be part of the cause of why this story was different from the other two in the results.

As a result of the data gathered, we cannot for sure increase descriptiveness by modifying the prompts arbitrarily. Some stories may end up with different descriptiveness results. In addition, there may be unintended bias due to the fact that the prompts were designed by only a single person. If there were more researchers dedicated to the task or to peer reviewing the prompt made, then bias could be reduced. Obviously there will always be societal biases that will factor into the results, and GPT-3 itself contains its own biases based on its training data. There is little that can be done to remedy these issues as of now, but hopefully in the future there will be more methods designed and implemented to do so or more objective tools than GPT-3 created.

Discussion and Conclusion

Overall, stories generated using planning and GPT-3 are coherent and can be more descriptive. However, as getting GPT-3 to output more descriptive stories based on the input prompts is heavily inconsistent, descriptiveness cannot be relied on. Perhaps if the exact same number of descriptive words were added to each of the prompts, then the results would have been more consistent. Further research could explore this in more depth.

In addition, the inconsistency could also be solely due to the limitations of GPT-3 itself. As GPT-3 has a transformer architecture, it may be more inconsistent as it does not know which part of the prompt to focus on. Further, if some sentences are more descriptive than others, that could be affecting the resulting descriptiveness. Despite these limitations, GPT-3 is a ready-to-use resource for developing stories quickly and in bulk.

In order to apply this planning to the GPT-3 story-generating method in the real world, a few modifications would have to be made. First, the stories would need to be automatically generated so that teachers or video game designers would not need to manually abstract the objects that they want in their story. There are natural language models that can, with limited accuracy, determine the objects in a story. However, extracting actions in all of their complexity would be much more difficult, and further those actions still need to have corresponding natural language statements to be able to convert them to GPT-3 input. Thus, to fully automate the PDDL-generating process would be laborious and may require significant programming efforts.

However, the rest of the story-generating process could, in fact, be done automatically. Further, if descriptiveness was desired, an algorithm could look at solely the longer stories as those tend to have more descriptive words. If the planning community can create enough planning domains and problems to generate stories, then those could easily be applied in the real world. The types of possible stories would be limited; however, it could be a starting

point for developing more story-centric video games and providing teachers with more material for class. Obviously, these stories would need to be for higher education though due to the reading level. Also, the stories should be curated and created using multiple peer-reviewed prompts to ensure that there is little bias entering the classroom or video games, as both have a wide influence.

Bibliography

[1] Bram Adams et al. *Overview - OpenAI API*. URL: https://beta.openai.com (visited on 11/13/2022).

[2] Masataro Asai and Alex Fukunaga. "Classical Planning in Deep Latent Space: From Unlabeled Images to PDDL (and back)." In: *NeSy*. 2017. URL: http://ceur-ws.org/Vol-2003/NeSy17_paper3.pdf (visited on 09/20/2022).

[3] Yoshua Bengio, Patrice Simard, and Paolo Frasconi. *Learning long-term dependencies with gradient descent is difficult*. Mar. 1994. URL: https://www.comp.hkbu.edu.hk/~markus/teaching/comp7650/tnn-94-gradient.pdf (visited on 12/01/2022).

[4] Matthew Christensen, Jennifer Nelson, and Rogelio Cardona-Rivera. "Using Domain Compilation to Add Belief to Narrative Planners". In: *Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment* 16.1 (Oct. 2020), pp. 38–44. URL: https://ojs.aaai.org/index.php/AIIDE/article/view/7405 (visited on 09/20/2022).

[5] Andrew Coles, Christian Muise, and Kristie Taylor-Muise. *Solver.Planning.Domains: An automated planner in the cloud*. URL: https://solver.planning.domains (visited on 11/10/2022).

[6] Explosion. *Spacy · industrial-strength natural language processing in python*. URL: https://spacy.io/ (visited on 11/10/2022).

[7] Jonas Freiknecht and Wolfgang Effelsberg. "Procedural Generation of Interactive Stories Using Language Models". In: *International Conference on the Foundations of Digital Games*. FDG '20. Bugibba, Malta: Association for Computing Machinery, 2020. ISBN: 9781450388078. DOI: 10.1145/3402942.3409599. URL: https://doi.org/10.1145/3402942.3409599 (visited on 09/20/2022).

[8] Richard A George. In: *Fields: Journal of Huddersfield Student Research* 1.1 (2015), pp. 139–158. URL: https://search.informit.org/doi/10.3316/informit.693636692874339 (visited on 09/20/2022).

[9] Jacob Grimm and Wilhelm Grimm. *Grimm's Household Tales: Rapunzel*. 1812. URL: https://www.cs.cmu.edu/~spok/grimmtmp/009.txt (visited on 09/20/2022).

[10] Thomas Hayton et al. "Narrative Planning Model Acquisition from Text Summaries and Descriptions". In: *Proceedings of the AAAI Conference on Artificial Intelligence* 34.02 (Apr. 2020), pp. 1709–1716. DOI: 10.1609/aaai.v34i02.5534. URL: https://ojs.aaai.org/index.php/AAAI/article/view/5534 (visited on 09/20/2022).

[11] Nicklas Slorup Johansen et al. "Towards Believable Non-Player Characters with Domain-Independent Planning". In: (). URL: https://icaps22.icaps-conference.org/workshops/SPARK/papers/spark2022_paper_5.pdf (visited on 09/20/2022).

[12] Bilal Kartal, John Koenig, and Stephen J Guy. "User-driven narrative variation in large story domains using monte carlo tree search". In: *Proceedings of the 2014 international conference on Autonomous agents and multi-agent systems*. Citeseer. 2014, pp. 69–76. URL: http://motion.cs.umn.edu/pub/Story/p69.pdf (visited on 09/20/2022).

[13] Andrew Lang. *The Heart of a Monkey — Andrew Lang's Fairy Books*. 1890. URL: https://fairytalez.com/the-heart-of-a-monkey/ (visited on 09/20/2022).

[14] Li Lucy and David Bamman. "Gender and representation bias in GPT-3 generated stories". In: *Proceedings of the Third Workshop on Narrative Understanding*. 2021, pp. 48–55. URL: https://aclanthology.org/2021.nuse-1.5.pdf (visited on 09/20/2022).

[15] Andrea Marrella and Stavros Vassos. "Story Generation in PDDL Using Character Moods: A Case Study on Iliad's First Book". In: *Artificial Intelligence: Methods and Applications*. Ed. by Aristidis Likas, Konstantinos Blekas, and Dimitris Kalles. Cham: Springer International Publishing, 2014, pp. 583–588. ISBN: 978-3-319-07064-3. URL: https://link.springer.com/chapter/10.1007/978-3-319-07064-3_50 (visited on 09/20/2022).

[16] M. Onat Topal, Anil Bas, and Imke van Heerden. "Exploring Transformers in Natural Language Generation: GPT, BERT, and XLNet". In: *arXiv e-prints*, arXiv:2102.08036 (Feb. 2021), arXiv:2102.08036. arXiv: 2102.08036 [cs.CL].

[17] Norman Hinsdale Pitman. *The Golden Nugget — A Chinese Wonder Book*. 1919. URL: https://fairytalez.com/golden-nugget/ (visited on 09/20/2022).

[18] Julie Porteous, Marc Cavazza, and Fred Charles. "Applying Planning to Interactive Storytelling: Narrative Control Using State Constraints". In: *ACM Trans. Intell. Syst. Technol.* 1.2 (Dec. 2010). ISSN: 2157-6904. DOI: 10.1145/1869397.1869399. URL: https://doi.org/10.1145/1869397.1869399 (visited on 09/20/2022).

[19] Julie Porteous and Alan Lindsay. "Protagonist vs antagonist provant: Narrative generation as counter planning". In: *Proceedings of the 18th International Conference on Autonomous Agents and MultiAgent Systems*. 2019, pp. 1069–1077. URL: https://www.ifaamas.org/Proceedings/aamas2019/pdfs/p1069.pdf (visited on 09/20/2022).

[20] Julie Porteous et al. "Automated extension of narrative planning domains with antonymic operators". In: (2015). URL: https://kipdf.com/automated-extension-of-narrative-planning-domains-with-antonymic-operators_5aacae1d1723ddf8ce2b40fa.html (visited on 09/20/2022).

[21] Julie Porteous et al. "Extending Narrative Planning Domains with Linguistic Resources". In: *Proceedings of the 19th International Conference on Autonomous Agents and MultiAgent Systems*. AAMAS '20. Auckland, New Zealand: International Foundation for Autonomous Agents and Multiagent Systems, 2020, pp. 1081–1089. ISBN: 9781450375184. URL: https://dl.acm.org/doi/abs/10.5555/3398761.3398887 (visited on 09/20/2022).

[22] Readable. *Measure the Readability of Text - Text Analysis Tools - Unique readability tools to improve your writing! App.readable.com*. URL: https://app.readable.com/text (visited on 12/01/2022).

[23] Justus Robertson and R Michael Young. "Automated gameplay generation from declarative world representations". In: *Eleventh Artificial Intelligence and Interactive Digital Entertainment Conference*. 2015.

URL: https://www.aaai.org/ocs/index.php/AIIDE/AIIDE15/paper/viewPaper/11547 (visited on 09/20/2022).

[24] Nisha Simon and Christian Muise. "A Natural Language Model for Generating PDDL". In: (Aug. 2021). URL: https://icaps21.icaps-conference.org/workshops/KEPS/Papers/KEPS_2021_paper_7.pdf (visited on 09/20/2022).

[25] Nisha Simon and Christian Muise. *TattleTale: Storytelling with Planning and Large Language Models*. 2022. URL: https://icaps22.icaps-conference.org/workshops/SPARK/papers/spark2022_paper_2.pdf (visited on 09/20/2022).

[26] Mingrong Tang, Jinxin Ni, and Qifeng Zhou. "A PDDL based Disaster Storyline Generation Approach". In: *2020 39th Chinese Control Conference (CCC)*. 2020, pp. 4273–4278. DOI: 10.23919/CCC50068.2020.9188462. URL: https://doi.org/10.23919/CCC50068.2020.9188462 (visited on 09/20/2022).

[27] Stephen G Ware and R Michael Young. "Glaive: a state-space narrative planner supporting intentionality and conflict". In: *Tenth Artificial Intelligence and Interactive Digital Entertainment Conference*. 2014. URL: https://www.aaai.org/ocs/index.php/AIIDE/AIIDE14/paper/viewPaper/8991 (visited on 09/20/2022).

[28] Stephen G. Ware and Cory Siler. "Sabre: A Narrative Planner Supporting Intention and Deep Theory of Mind". In: *Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment* 17.1 (Oct. 2021), pp. 99–106. URL: https://ojs.aaai.org/index.php/AIIDE/article/view/18896 (visited on 09/20/2022).

Appendix A: Example Story Output

As the stories created were fairly long, we have added an example story here in this appendix to demonstrate how GPT-3 output turned out:

```
The enchantress uses Rapunzel's hair to climb down the tower. The king
    strides purposefully from the castle, his destination the forest. The
    king strides through the forest, his feet sinking into the soft, spongy
    earth beneath him. He approaches the house, his eyes seeking out the
    window of the tower where his daughter is kept. The enchantress left the
     tower and went to the forest.
The enchantress made her way from the forest to the house. She walked
    through the trees, her long black dress trailing behind her. The sun was
     setting, and the sky was turning a deep blue. The enchantress smiled to
     herself as she thought of the surprise she was going to give the young
    girl locked in the tower.
Rapunzel's father left the house and went to the forest.
Rapunzel's mother serenades everyone in the kingdom with her beautiful
    singing voice.
The king presents the ladder to Rapunzel's mother as a gift.
Rapunzel's mother walks through the forest to get to the tower where her
    daughter is. The king makes his way to the garden, striding confidently
    across the grounds. He is eager to see the fruit of his labor and to
    bask in the sunshine.The enchantress goes from the house to the garden.
    Rapunzel's father leaves the safety of the forest and approaches the
    tower where his daughter is being held captive.
Rapunzel's mother walked from the forest to the tower so that she could see
    her daughter again.
Rapunzel sings a beautiful song, enchanting everyone who hears it.
Rapunzel's mother hands the ladder over to Rapunzel.
Rapunzel uses the ladder to climb down from the tower.
Rapunzel hands the ladder over to her mother, who can then use it to climb
    up to the tower.
Rapunzel walks away from the tower and towards the forest. She is excited to
     finally be able to explore beyond the tower and to see what the outside
     world is like.
```

Listing 7: Example GPT-3 appended output for "Rapunzel" story variant 3, prompt 2 of medium descriptiveness. As can be seen, there is some descriptiveness; however, the story remains very action-oriented.

Appendix B: Complete Raw Data

In this appendix we include the raw data collected from each iteration of running the prompts on GPT-3. The data is separated by prompt into 3 different tables, one for each prompt. Recall that PROMPT1 is the least descriptive, PROMPT2 is of medium descriptiveness, and PROMPT3 is the most descriptive. At the beginning of each table is the data for the original prompt input as reference to how the values may be influenced by the prompt's descriptiveness.

prompt1	statistics	RAP1	RAP2	RAP3	GOLD1	GOLD2	GOLD3	MONK1	MONK2	MONK3
p1input	#sentences	7	9	8	10	10	10	6	6	6
	#words	80	107	105	186	177	170	97	97	97
	#word/sentence	11.429	11.889	13.125	18.6	17.7	17	16.167	16.167	16.167
	#adjectives	2	2	1	12	10	9	5	5	5
	#adverbs	2	5	2	13	12	12	4	4	4
	#prep&subj	8	9	11	15	14	14	8	8	8
	total	12	16	14	40	36	35	17	17	17
	total/words	0.15	0.15	0.133	0.215	0.203	0.206	0.175	0.175	0.175
p1i1	#sentences	19	30	26	24	19	11	13	10	12
	#words	305	495	428	406	300	219	249	182	222
	#word/sentence	16.053	16.5	16.462	16.917	15.789	19.909	19.154	18.2	18.5
	#adjectives	14	20	16	17	12	6	6	6	7
	#adverbs	6	12	4	9	4	6	3	2	1
	#prep&subj	26	54	51	48	28	33	24	16	18
	total	46	86	71	74	44	45	33	24	26
	total/words	0.151	0.174	0.166	0.182	0.147	0.205	0.133	0.132	0.117
p1i2	#sentences	14	28	20	28	16	19	15	8	10
	#words	230	474	312	465	254	391	273	155	216
	#word/sentence	16.429	16.929	15.6	16.607	15.875	20.579	18.2	19.375	21.6
	#adjectives	6	13	3	16	18	18	5	2	4
	#adverbs	9	10	4	19	5	7	7	1	5
	#prep&subj	23	52	37	57	22	54	26	17	23
	total	38	75	44	92	45	79	38	20	32
	total/words	0.165	0.158	0.141	0.198	0.177	0.202	0.139	0.129	0.148
p1i3	#sentences	16	29	26	26	25	12	16	10	15
	#words	237	508	413	493	389	226	290	214	275
	#word/sentence	14.813	17.517	15.885	18.962	15.56	18.833	18.125	21.4	18.333
	#adjectives	6	10	7	20	18	7	8	1	10
	#adverbs	0	13	7	12	15	5	5	9	5
	#prep&subj	28	57	53	62	39	34	28	26	23
	total	34	80	67	94	72	46	41	36	38
	total/words	0.143	0.157	0.162	0.191	0.185	0.204	0.141	0.168	0.138
p1i4	#sentences	15	29	27	25	21	14	13	14	16
	#words	229	542	389	458	382	261	274	282	274
	#word/sentence	15.267	18.69	14.407	18.32	18.19	18.643	21.077	20.143	17.125
	#adjectives	8	24	5	21	15	12	9	11	5
	#adverbs	1	11	3	15	16	9	5	4	2
	#prep&subj	22	65	42	51	37	31	34	28	26
	total	31	100	50	87	68	52	48	43	33
	total/words	0.135	0.185	0.129	0.19	0.178	0.199	0.175	0.152	0.12
p1i5	#sentences	18	34	21	23	14	14	18	9	18
	#words	314	582	343	432	241	299	346	193	325
	#word/sentence	17.444	17.118	16.333	18.783	17.214	21.357	19.222	21.444	18.056
	#adjectives	14	23	6	26	14	10	6	6	6
	#adverbs	3	15	3	12	2	10	9	5	7
	#prep&subj	34	61	43	46	23	46	41	24	29
	total	51	99	52	84	39	66	56	35	42
	total/words	0.162	0.17	0.152	0.194	0.162	0.221	0.162	0.181	0.129
average	#sentences	16.4	30	24	25.2	19	14	15	10.2	14.2
	#words	263	520.2	377	450.8	313.2	279.2	286.4	205.2	262.4
	#word/sentence	16.001	17.351	15.737	17.918	16.526	19.864	19.156	20.112	18.723
	#adjectives	9.6	18	7.4	20	15.4	10.6	6.8	5.2	6.4
	#adverbs	3.8	12.2	4.2	13.4	8.4	7.4	5.8	4.2	4
	#prep&subj	26.6	57.8	45.2	52.8	29.8	39.6	30.6	22.2	23.8
	total	40	88	56.8	86.2	53.6	57.6	43.2	31.6	34.2
	total/words	0.151	0.169	0.15	0.191	0.17	0.206	0.15	0.153	0.131

Table 7: Full descriptive statistic data for every iteration of prompt 1 for all story variants as well as the original input prompt's statistics at the top.

prompt2	statistics	RAP1	RAP2	RAP3	GOLD1	GOLD2	GOLD3	MONK1	MONK2	MONK3
p2input	#sentences	7	9	8	10	10	11	6	6	6
	#words	94	127	131	201	196	192	126	126	127
	#word/sentence	13.429	14.111	16.375	20.1	19.6	17.455	21	21	21.167
	#adjectives	6	7	5	16	15	14	9	9	9
	#adverbs	2	5	4	16	17	17	6	6	6
	#prep&subj	11	12	15	15	14	15	12	12	12
	total	19	24	24	47	46	46	27	27	27
	total/words	0.202	0.189	0.183	0.234	0.235	0.24	0.214	0.214	0.213
p2i1	#sentences	18	39	26	19	27	16	12	7	15
	#words	282	714	408	321	532	292	277	165	331
	#word/sentence	15.667	18.308	15.692	16.895	19.704	18.25	23.083	23.571	22.067
	#adjectives	11	24	9	10	22	15	2	2	13
	#adverbs	5	19	11	6	31	11	8	4	7
	#prep&subj	30	68	49	41	50	38	33	18	31
	total	46	111	69	57	103	64	43	24	51
	total/words	0.163	0.155	0.169	0.178	0.194	0.219	0.155	0.145	0.154
p2i2	#sentences	16	44	24	24	15	11	22	15	11
	#words	229	716	392	360	269	216	472	323	224
	#word/sentence	14.313	16.273	16.333	15	17.933	19.636	21.455	21.533	20.364
	#adjectives	9	31	10	14	6	8	15	12	4
	#adverbs	5	17	6	5	7	12	16	11	4
	#prep&subj	26	73	41	42	33	27	39	39	22
	total	40	121	57	61	46	47	70	62	30
	total/words	0.175	0.169	0.145	0.169	0.171	0.218	0.148	0.192	0.134
p2i3	#sentences	18	33	22	20	13	11	16	12	10
	#words	338	569	391	304	220	231	332	246	243
	#word/sentence	18.778	17.242	17.773	15.2	16.923	21	20.75	20.5	24.3
	#adjectives	17	26	11	12	11	10	12	8	8
	#adverbs	14	8	7	6	3	5	7	5	5
	#prep&subj	35	62	43	36	29	31	30	21	27
	total	66	96	61	54	43	46	49	34	40
	total/words	0.195	0.169	0.156	0.178	0.195	0.199	0.148	0.138	0.165
p2i4	#sentences	20	34	21	22	18	14	17	13	15
	#words	333	658	327	363	341	267	348	258	322
	#word/sentence	16.65	19.353	15.571	16.5	18.944	19.071	20.471	19.846	21.467
	#adjectives	14	27	9	14	18	13	10	5	13
	#adverbs	8	20	6	13	6	4	9	9	10
	#prep&subj	38	82	37	48	45	35	37	29	30
	total	60	129	52	75	69	52	56	43	53
	total/words	0.18	0.196	0.159	0.207	0.202	0.195	0.161	0.167	0.165
p2i5	#sentences	21	31	24	20	18	12	15	9	12
	#words	364	538	376	356	291	233	291	224	296
	#word/sentence	17.333	17.355	15.667	17.8	16.167	19.417	19.4	24.889	24.667
	#adjectives	24	12	13	17	12	10	12	10	8
	#adverbs	13	12	6	10	9	8	10	6	7
	#prep&subj	37	59	40	44	32	27	26	23	36
	total	74	83	59	71	53	45	48	39	51
	total/words	0.203	0.154	0.157	0.199	0.182	0.193	0.165	0.174	0.172
average	#sentences	18.6	36.2	23.4	21	18.2	12.8	16.4	11.2	12.6
	#words	309.2	639	378.8	340.8	330.6	247.8	344	243.2	283.2
	#word/sentence	16.548	17.706	16.207	16.279	17.934	19.475	21.032	22.068	22.573
	#adjectives	15	24	10.4	13.4	13.8	11.2	10.2	7.4	9.2
	#adverbs	9	15.2	7.2	8	11.2	8	10	7	6.6
	#prep&subj	33.2	68.8	42	42.2	37.8	31.6	33	26	29.2
	total	57.2	108	59.6	63.6	62.8	50.8	53.2	40.4	45
	total/words	0.183	0.169	0.157	0.186	0.189	0.205	0.155	0.163	0.158

Table 8: Full descriptive statistic data for every iteration of prompt 2 for all story variants as well as the original input prompt's statistics at the top.

prompt3	statistics	RAP1	RAP2	RAP3	GOLD1	GOLD2	GOLD3	MONK1	MONK2	MONK3
p3input	#sentences	7	9	9	10	10	10	8	8	8
	#words	128	171	181	220	215	229	165	165	165
	#word/sentence	18.286	19	20.111	22	21.5	22.9	20.625	20.625	20.625
	#adjectives	13	13	13	17	16	17	14	14	14
	#adverbs	4	10	7	19	20	22	10	10	10
	#prep&subj	19	23	25	20	18	21	15	15	15
	total	36	46	45	56	54	60	39	39	39
	total/words	0.281	0.269	0.249	0.255	0.251	0.262	0.236	0.236	0.236
p3i1	#sentences	21	37	35	30	16	17	17	12	14
	#words	394	662	654	515	293	316	376	311	313
	#word/sentence	18.762	17.892	18.686	17.167	18.313	18.588	22.118	25.917	22.357
	#adjectives	19	31	26	24	15	9	18	8	13
	#adverbs	10	19	17	17	6	8	12	12	9
	#prep&subj	40	67	77	61	33	44	40	33	34
	total	69	117	120	102	54	61	70	53	56
	total/words	0.175	0.177	0.183	0.198	0.184	0.193	0.186	0.17	0.179
p3i2	#sentences	19	32	28	23	17	13	17	11	10
	#words	343	625	485	436	292	246	330	260	248
	#word/sentence	18.053	19.531	17.321	18.957	17.176	18.923	19.412	23.636	24.8
	#adjectives	15	30	9	22	15	12	11	9	8
	#adverbs	11	15	14	10	3	4	8	4	5
	#prep&subj	33	78	55	58	34	30	37	30	28
	total	59	123	78	90	52	46	56	43	41
	total/words	0.172	0.197	0.161	0.206	0.178	0.187	0.17	0.165	0.165
p3i3	#sentences	16	53	31	25	14	12	19	13	13
	#words	301	1016	562	408	254	219	406	272	299
	#word/sentence	18.813	19.17	18.129	16.32	18.143	18.25	21.368	20.923	23
	#adjectives	14	57	22	17	15	6	7	13	9
	#adverbs	11	29	18	14	4	6	8	9	7
	#prep&subj	33	111	69	47	23	31	17	22	36
	total	58	197	109	78	42	43	32	44	52
	total/words	0.193	0.194	0.194	0.191	0.165	0.196	0.079	0.162	0.174
p3i4	#sentences	17	28	29	32	19	11	20	11	12
	#words	285	521	551	520	348	206	431	240	311
	#word/sentence	16.765	18.607	19	16.25	18.316	18.727	21.55	21.818	25.917
	#adjectives	10	24	24	19	20	6	16	5	12
	#adverbs	13	10	10	20	7	4	18	8	4
	#prep&subj	34	54	78	55	36	31	46	30	39
	total	57	88	112	94	63	41	80	43	55
	total/words	0.2	0.169	0.203	0.181	0.181	0.199	0.186	0.179	0.177
p3i5	#sentences	19	34	30	25	18	12	12	11	10
	#words	290	635	581	463	326	236	264	259	259
	#word/sentence	15.263	18.676	19.367	18.52	18.111	19.667	22	23.545	25.9
	#adjectives	14	29	20	22	15	12	7	9	9
	#adverbs	11	13	9	14	6	9	6	7	3
	#prep&subj	33	66	80	57	36	28	30	27	33
	total	58	108	109	93	57	49	43	43	45
	total/words	0.2	0.17	0.188	0.201	0.175	0.208	0.163	0.166	0.174
average	#sentences	18.4	36.8	30.6	27	16.8	13	17	11.6	11.8
	#words	322.6	691.8	566.6	468.4	302.6	244.6	361.4	268.4	286
	#word/sentence	17.531	18.775	18.501	17.443	18.012	18.831	21.29	23.168	24.395
	#adjectives	14.4	34.2	20.2	20.8	16	9	11.8	8.8	10.2
	#adverbs	11.2	17.2	13.6	15	5.2	6.2	10.4	8	5.6
	#prep&subj	34.6	75.2	71.8	55.6	32.4	32.8	34	28.4	34
	total	60.2	126.6	105.6	91.4	53.6	48	56.2	45.2	49.8
	total/words	0.188	0.181	0.186	0.195	0.177	0.197	0.157	0.169	0.174

Table 9: Full descriptive statistic data for every iteration of prompt 3 for all story variants as well as the original input prompt's statistics at the top.

Appendix C: Average Data

In this appendix, we compiled all of the average data for each statistic, each prompt, and each story variant all in one table. The resulting changes in descriptiveness based on prompt can be seen for each descriptive statistic.

Averages	prompt #	RAP1	RAP2	RAP3	GOLD1	GOLD2	GOLD3	MONK1	MONK2	MONK3
#sentences	prompt1	16.4	30	24	25.2	19	14	15	10.2	14.2
	prompt2	18.6	36.2	23.4	21	18.2	12.8	16.4	11.2	12.6
	prompt3	18.4	36.8	30.6	27	16.8	13	17	11.6	11.8
#words	prompt1	263	520.2	377	450.8	313.2	279.2	286.4	205.2	262.4
	prompt2	309.2	639	378.8	340.8	330.6	247.8	344	243.2	283.2
	prompt3	322.6	691.8	566.6	468.4	302.6	244.6	361.4	268.4	286
#word/sentence	prompt1	16.001	17.351	15.737	17.918	16.526	19.864	19.156	20.112	18.723
	prompt2	16.548	17.706	16.207	16.279	17.934	19.475	21.032	22.068	22.573
	prompt3	17.531	18.775	18.501	17.443	18.012	18.831	21.29	23.168	24.395
#adjectives	prompt1	9.6	18	7.4	20	15.4	10.6	6.8	5.2	6.4
	prompt2	15	24	10.4	13.4	13.8	11.2	10.2	7.4	9.2
	prompt3	14.4	34.2	20.2	20.8	16	9	11.8	8.8	10.2
#adverbs	prompt1	3.8	12.2	4.2	13.4	8.4	7.4	5.8	4.2	4
	prompt2	9	15.2	7.2	8	11.2	8	10	7	6.6
	prompt3	11.2	17.2	13.6	15	5.2	6.2	10.4	8	5.6
#prep&subj	prompt1	26.6	57.8	45.2	52.8	29.8	39.6	30.6	22.2	23.8
	prompt2	33.2	68.8	42	42.2	37.8	31.6	33	26	29.2
	prompt3	34.6	75.2	71.8	55.6	32.4	32.8	34	28.4	34
total	prompt1	40	88	56.8	86.2	53.6	57.6	43.2	31.6	34.2
	prompt2	57.2	108	59.6	63.6	62.8	50.8	53.2	40.4	45
	prompt3	60.2	126.6	105.6	91.4	53.6	48	56.2	45.2	49.8
total/word	prompt1	0.151	0.169	0.15	0.191	0.17	0.206	0.15	0.153	0.131
	prompt2	0.183	0.169	0.157	0.186	0.189	0.205	0.155	0.163	0.158
	prompt3	0.188	0.181	0.186	0.195	0.177	0.197	0.157	0.169	0.174

Table 10: Full descriptive statistic data for averages of each iteration of GPT-3 output for all prompts.

Appendix D: Bash Shell Script

This appendix includes the code for the bash shell script for running each iteration of each story variant for each prompt. This script takes command line input for the story folder, story type, number of iterations, and iteration starting index and outputs the descriptiveness results. It relies on the python file GENERATE_STORIES_AND_DESCRIPTIVENESS.PY to run the entire process of generating stories, and stores the resulting stories in a folder called OUTPUT.

```
#!/bin/bash

run_iterations()
{
    STORY_FOLDER=$1
    STORY_NUM=$2
    ITERATIONS=$3
    START_INDEX=$4

    NEWLINE=$'\n'
    echo "Working on story: ${STORY_FOLDER}...${NEWLINE}"


    PROMPT_FOLDER="${STORY_FOLDER}/prompts"

    for prompt_file in $(find "${PROMPT_FOLDER}" -type f -name "*.txt" |
        sort); do
        name=`basename ${prompt_file} .txt`
        echo "${name}"
        problem_file=${STORY_FOLDER}/pddl/problem.pddl
        domain_file=${STORY_FOLDER}/pddl/domain.pddl
        plan_file=${STORY_FOLDER}/pddl/plan.ipc
        gpt3_r_file=${STORY_FOLDER}/gpt3/${name}_r.txt

        story_name=`basename ./${STORY_FOLDER} `

        for ((i=START_INDEX; i <=${ITERATIONS}+START_INDEX-1; i++))
        do
            gpt3_w_file="./output/${story_name}_${name}_${i}.txt"
            #try to remove write file so we don't end up appending at end
            if [ -f "${gpt3_w_file}" ]; then
                rm "${gpt3_w_file}"
            fi

            ./generate_stories_and_descriptiveness.py "${prompt_file}" "${
                domain_file}" "${problem_file}" "${plan_file}" "${
                gpt3_r_file}" "${gpt3_w_file}" "${STORY_NUM}"

        done

    done
}
ARG4=${4:-1}
echo "folder: $1"
echo "Story: $2"
echo "Iterations: $3"
echo "Starting Index: ${ARG4}"
run_iterations $1 $2 $3 ARG4
```

Listing 8: Bash test script file to run each iteration of genreating each story variant for each prompt. The story generation is run in generate_stories_and_descriptivness.py, with outputted stories being stored in a folder ./output.

Appendix E: Main Python Program

This appendix includes the main function from the python program GENERATE_STORIES_AND_DESCRIPTIVENESS.PY that automates the entire process from generating a plan to obtaining story output from GPT-3 and determining descriptiveness. A function planFileToActions() is omitted but takes the plan and converts the actions to natural language as described in the Approach section of this paper.

The program calls three other helper python programs: SOLVE.PY, GPT3.PY, and STATS.PY. SOLVE.PY takes a problem and domain file, submits them to solver.planning.domains as a JSON file and writes the resulting plan to a file. GPT3.PY takes an input file and writes the resulting GPT-3 output to a specified output file given the model to use, maximum number of tokens, and temperature. STATS.PY uses spaCy to tag every word with a part of speech and count the number of sentences, number of words, number of adjectives, number of adverbs, and number of prepositions and suboordinating conjunctions.

```
def main():
    # read in input from the command line
    initial_prompt = sys.argv[1] #initial prompt file
    domain = sys.argv[2] #domain file for planning problem
    problem = sys.argv[3] #problem file for planning problem
    plan = sys.argv[4] # file to store the solved plan in
    gpt3_r = sys.argv[5] #file to store the initial input to GPT-3 in
    gpt3_w = sys.argv[6] #file to store resulting GPT-3 output (appended
        together)
    storyNum = int(sys.argv[7]) #story number: 0 for "Rapunzel", 1 for "The
        Golden Nugget", or 2 for "The Heart of a Monkey"

    #print out the descriptiveness statistics for the initial prompt (using
        stats.py, a program to run spaCy to count descriptive words)
    print("Here's the initial prompt:")
    stats.descriptiveStats(initial_prompt)

    #find the plan of actions based on the domain and problem files using
        solve.py, a program to call solver.planning.domains
    solve.find_plan(domain, problem, plan)

    #iterate through every action converted to natural language in a
        function planFileToActions
    for action in planFileToActions(plan):
        #open the GPT-3 input file, write the prompt we have stored along
            with the action we're on
        f = open(gpt3_r, "w")
        with open(initial_prompt) as prompt:
            f.write(prompt.read())
        f.close()
        f = open(gpt3_r, "a")
        f.write("\nHere is the next action in the story:\n")
        action_text = actionToText(action, storyNum)
        f.write(action_text)
        f.close()

        #ask gpt-3 to paraphrase that action
        f = open(gpt3_r, "a")
        f.write("\n\nParaphrase that action in different, more elaborate
            words:\n")
        f.close()

        #store the resulting output in the gpt3_w file using gpt3.py with
            max tokens 256 and temperature 0.7
        gpt3.complete(gpt3_r, gpt3_w, "text-davinci-002", 256, 0.7)

    #determine resulting statistics for the story output using stats.py
    stats.descriptiveStats(gpt3_w)
```

Listing 9: Main function of a python program that automates the entire process of solving the planning problem (done in solve.py), parsing and converting the plan actions to natural language, sending the input prompts to gpt3 (using gpt3.py) and obtaining output, and printing out the resulting descriptivness statistics (using stats.py).