Procedural Quest Generation

Max Quintavalle
quint218@morris.umn.edu
Division of Science and Mathematics
University of Minnesota, Morris
Morris, Minnesota, USA

Abstract

The idea of quests used in this paper will be the concept of an action or set of actions that are given to a player to complete to obtain some sort of reward. Quests are found everywhere in video games and a core part of how the video games work and function. Quests have expanded in how they are used and what can be achieved through them. Procedural content generation has become increasingly common in video games. For example in the popular game "*The Binding Of Isaac*" all the layouts of the levels you go through are created through procedural content generation. Now with improvements in technology and AI we start looking into having the quests themselves become procedurally generated. Procedural quest generation is a specific type of content generation where quests are created systematically with the assistance of procedural generation. This work presents the research of two different methods of performing procedural quest generation. The first method is based on user input, where the user will type something in and then a quest will be created on its own without any actual developer input outside of the creation of the method. The second idea we look at is a program called Questgram that uses AI to help create quests while the actual developer decides if they want to use the quests created by the program. Each of these methods has distinct pros and cons, and each contributes to improving the idea of procedural quest generation.

Keywords: Procedural Content Generation, Large Language Model

1 Introduction

Procedural content generation is the idea of generating data algorithmically instead of manually that would be used as some sort of content for a user. The idea of using procedural content generation for quests is a relatively new topic and has only had a small amount of research done so far. The main idea behind the quest is based in a typical RPG (Role Playing Game) or fantasy setting because those are the games where you would be the most likely to find quests in games. That fact is important because the first method we will be going through will go over the pipeline method (will be explained further in Section 2) for creating a quest and will do so in the style of fantasy only. Ashby et al. describe a process of taking a user's text input and using knowledge graphs and large language models to create a quest tuple [2]. A quest tuple is the three main parts of the quest which are the quest title, quest description, and a quest dialogue to go with the quest. There were 2 studies performed for the pipeline method. In the first study [2], Ashby et al. test how their large language model performs when compared to other large language models using their own custom metrics (described in Section 2), and in the second study [2], the satisfaction of the users and the responsiveness of the pipeline results is tested.

The second method that is involved is the Questgram method. Questgram is a method that takes a different approach. Instead of going off user input, it actually assists the developer of the quest during the creation process. It does this through a program it took inspiration from called Evolutionary Dungeon Designer where AI works with the developer to help create a product. The Questgram approach has the AI give suggestions to the developer on what to add to the quest line. These suggestions would be quest actions like "Explore" or "Kill" that the developer can choose to add. There were two studies performed. One where they tested the AI to see how often the AI suggested certain actions, which they called an "Expressive Analysis", and the other test was a test done on 6 developers to see how they would rate the tool and AI and how it could be improved.

Section 2 will go over the pipeline method and the individual parts of it. After which will move into how it runs all together and end with analysis of the studies. Section 3 will review Questgram, the developer-oriented procedural generation technique. Broadly speaking, Section 3 is broken into 3 major parts: The first, Subsections 3.1 through 3.2, introduce the 2 main parts of the Questgram process, Actions and Grammar. Subsections 3.3 and 3.4 will present the results of the two Questgram studies. The final section, Section 4, compares the methodologies from Section 2 and Section 3 and concludes with a short reflection on possible future applications.

2 Pipeline Method

The first method is a "pipeline" of multiple parts to create a quest with dialogue from an NPC (non-player-character) based on player text input. For example the player could be talking to an NPC and the player states that they wish to kill a dragon. In response, the system would use the pipeline to create a quest based on that input. The quest would most likely be something along the lines of go kill said dragon and offer a reward of 10 rubies as payment. What makes

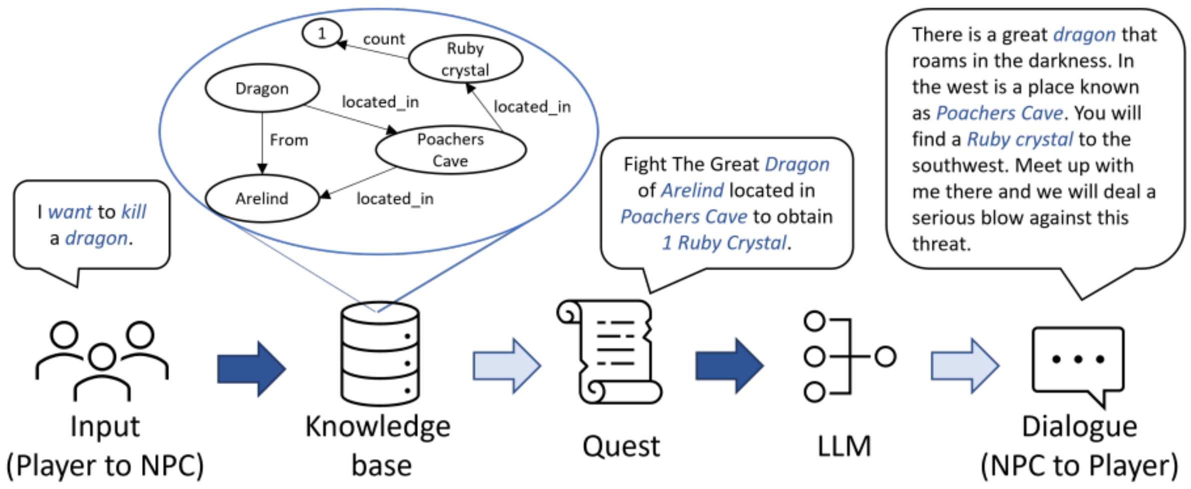

1
count
Ruby crystal
Dragon
located_in
located_in
From
Poachers Cave
Arelind
located_in
I want to kill a dragon.
Fight The Great Dragon of Arelind located in Poachers Cave to obtain 1 Ruby Crystal.
There is a great dragon that roams in the darkness. In the west is a place known as Poachers Cave. You will find a Ruby crystal to the southwest. Meet up with me there and we will deal a serious blow against this threat.
Input (Player to NPC)
Knowledge base
Quest
LLM
Dialogue (NPC to Player)

Figure 1. Image of the Pipeline Method (taken from [2])

this possible is the multistep pipeline that Ashby et al. [2] made for this (see Figure 1). The first step is the input (from player to NPC). Then, the input is passed to a knowledge base that contains a number (determined by the developer) of knowledge graphs of how different things in the game relate to each other. Next, the pipeline method will formulate a quest off of this knowledge graph (Section 2.2), which then is thrown into a large language model that would generate the dialogue for the quest which the NPC would give to the player.

2.1 Players

The idea to make the quests based off of the user input is to increase the enjoyment and the engagement of the players. The makers of this pipeline have attempted to achieve this by letting player input guide the automated construction of a new quest, instead of forcing the player to follow a randomly selected pregenerated quest that might have no real significance to the player. The way they attempt this is by letting the "players initiate quest generation via free-form text entered into the terminal." [2]

This is their process of having an NPC and a player interact and communicate with each other, which has similarities to way that a player would try to communicate with an NPC in most games. The pipeline method requires, though, that the message that the user wishes to type must be some sort of request about either some kind of action the player would like to do or some kind of goal they would like to achieve. The exact process of how the user input is configured into a quest will be later described in Section 2.3.

2.2 Knowledge Graphs

Knowledge graphs are simple graph where nodes represent objects, people, or places while the edges describe the relationships between the entities they connect. An example of this can be found in Figure 1. The knowledge graph in Ashby et al.'s pipeline method is set up in a way that nodes represents entities and objects in the game while the edges represent the relationships between these entities and objects. These relationships could be something "wants to have", "wants killed", "needs". After taking the user text input the knowledge graph is scanned to determine if any of the nodes and edges relate to the player's input and sent to the next part of the pipeline. The pipeline method has a knowledge base instead of a knowledge graph (See in Figure 1) and that is because their knowledge base is database of multiple different knowledge graphs. The knowledge base that Ashby et al. [2] use for the method to test with is relatively small, but they do believe that with a bigger knowledge base, the quest generation performance would improve. They made the knowledge base with their idea to be able to handle a multitude of different quest types. That can range from a variety of different objectives such as go to point A, collect said material, or something like kill this monster.

2.3 Quest Generation

In this subsection, we describe how the quest would be generated through their pipeline describing the actual events as it goes through the pipeline. We will use our example from earlier, which was where the player was talking to an NPC. Let us call the NPC John. The player types in their request "I want to kill a dragon" to John. Then, the pipeline program would find johns node in a knowledge graph and then grab all the edges and nodes that are within 2 nodes of johns. "Once all acceptable relations to the NPC nodes have been identified, their English-language representations are converted to a vector representation and compared via cosine similarity to the player's original input. The relation with the highest cosine score is selected as the first edge in a knowledge graph traversal to extract information with which to construct a quest" [2]. In our example we will say that John had a dragon that he wanted killed that scored the highest cosine value. Then, these terms are thrown into a quest structure/template that has sections to place nodes values and the relationships in to make a quest description. The quest structure is the framework of the quest with spaces designed to put in the terms taken from the knowledge graph. In the end it would make a quest with the title "Kill the red dragon" and a quest description saying "Kill the red dragon in the Sarum mountains for 10 rubies". That statement is then sent to the the large language model to create the dialogue.

2.4 Language Model

The large language model is used after the quest generation and uses the quest description to make the dialogue that is shown to the player. They use "the GPT-2 language model, a transformer-based neural network architecture that learns to predict likely continuations of a conditioning prompt or narrative context" [2]. They have trained this AI using World of Warcraft data which is a popular online video game that is in a fantasy setting similar to "*Lord of the Rings*". They made the data set up in such a way that the data always includes a quest, a quest title, and NPC dialogue to go with it as well for the AI to be trained on. They have put it this way because for their pipeline, they wish for every quest to have some kind of quest title, the quest description, and the NPC dialogue. The next step to the generation is to make the quest feel more immersive and give the NPC the dialogue to say to the player.

2.5 Large Language Model Dialogue Study

Ashby et al. performed a study to see the effect different large language models had on their system using 4 large language models and handcrafted World of Warcraft (WoW) Quests. Ashby et al. surveyed students from the computer science major at Brigham Young University, where the students were asked to write in a request and then rate the 5 quest dialogues they were given. The rating categories were Fluency, Coherence, Novelty, and Creativity and the following definitions were used (taken from [2]):

- Fluency: The dialogue makes use of correct English.
- Coherence: The goal is clear from the dialogue.
- Novelty: The dialogue is written in a novel way.
- Creativity: The dialogue is creative.

The main 2 large language models that should be focused on is the DRG-L model and VS-MYS model. The DRG-L model is the main model that is being used for the pipeline and is trained on the data set described earlier (Section 1) of quest data split into tuples (quest title, quest description, and dialogue). VS-MYS is a model developed by Stegeren and Myśliwiec [3]. This model was trained on a large amount of quest data not set in the tuple format, which made it unsuitable for the pipeline method. The results (found in Figure 2) from the study was determined that the handwritten WoW quests were rated the highest with DRG-L and VS-MYS scoring roughly the second highest. Descriptions of all the large language models used in the study can be found in Table 1.

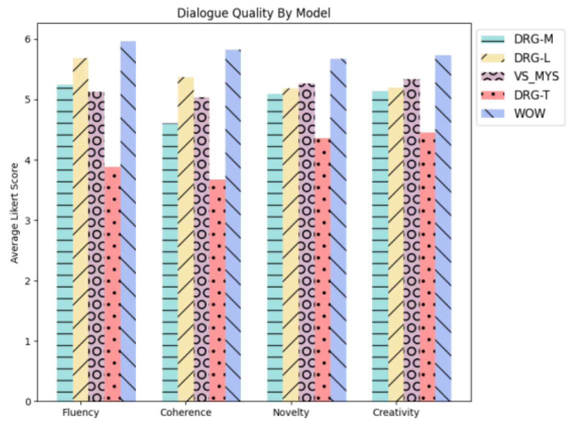


Figure 2. Dialogue quality results in terms of fluency, coherence, novelty, and creativity from the large language model dialogue quality study (taken from [2])

2.6 Satisfaction and Responsiveness Study

A second study was performed by the same researchers [2] on a subset of the previous participants. This study was to test overall player satisfaction with the system. The participants were instructed on how the system operated and told to engage in text-based conversation with a new NPC. Because of the way that the NPC was set up, the initial phrase the players gave were enough to trigger quest generation response where it would output 3 different quests. The three different responses to the participant were, "an option from our normal, knowledge graph-based model, a randomly-selected hand-written option from the World of Warcraft training data, and an option drawn from a naive, baseline 4-gram language model" [2]. All of these options followed the tuple pattern of Quest, Title, Dialogue and each of the options were shown randomly to the participant.They asked the participants then to answer 3 questions about the quests they were given in order to figure out which ones felt the most responsive and which ones were the most creative/exciting and their reasoning for choosing the ones they did. The results determined that players preferred the WoW quests even though they were random because the knowledge graph model and the n-gram model often felt semantically incongruent to the participant. Even though the

Model	Description
DRG-L	The largest version (774M parameters) of OpenAI's GPT-2 language model, fine-tuned using the Hugging Face transformers library on their dataset. The primary difference between this model and VS-MYS is in its training.
DRG-M	A medium-sized (355M parameters) implementation of GPT-2 fine-tuned on a cleaner subset of 25% (8638/34450) of their dataset. This model explores the ability of a smaller and more resource-effective language model to produce effective NPC dialogue.
DRG-T	A distilled variant of the (82M parameters) GPT-2 language model trained to generate quests whose locations, organizations, and people are replaced by back-filling from respective node values. This model tests the feasibility of learning generalized dialogue structures rather than fully developed NPC dialogue.
VS-MYS	A GPT-2 language model fine-tuned by van Stegeren and Myśliwiec [3] on a corpus of video game quests. This model produces NPC dialogue that most closely matches the qualities of handcrafted quests. However, due to differences in training, this model is incompatible with our content generation framework and cannot receive raw quests as input to dialogue generation.
WoW	(Not a language model.) Hand-coded quests from the World of Warcraft massively multiplayer online video game. These quests of necessity reflect neither the intent of the user nor the state of the game knowledge graph, but they are a useful comparison when evaluating the fluency and believability of quests generated via our framework.

Table 1. Descriptions of all the large language models used in the dialogue study (taken from [2])

WoW quests were the most preferred the pipeline method had slightly more votes (169 votes) than the WoW quests (159 votes) when it came to how responsive they were. After analyzing the results further they discovered that most of the participants asked something along the lines of "How can I help you?" or "Do you have any quests?" which caused the models to not have any specific data to actually use to make a quest and make the random WoW quest feel more appropriate.

3 Questgram Method

The Questgram (Qg) approach takes a different turn on the idea of procedural quest generation where the actual AI isused during game development, rather than generating quests on the fly. The Questgram approach is labeled aa a mixed-Initiative approach where the system or AI work with the developer to operate and complete a task. The system that Alvarez et al. [1] use follows the idea of making multiple quest actions, which are the simple actions that are required to complete a quest, to create an overall narrative questline all at once. Currently it only supports the option of one main quest line, but the makers of Qg do have the idea of expanding it to support the expansion of multiple quest lines.

3.1 Quest Generation

Qg is based off of a program called Evolutionary Dungeon Designer (EDD) which is a level dungeon designer and quest designer at the same time. In other words, EDD supports developing a dungeon while also allowing the incorporation of pieces of the dungeon within a quest. It already has its own mixed initiative AI system in place; it only works for making singular quests and not a long questline of multiple quests. Qg uses this program and has the AI give suggestions on how to continue on with the dungeon/quest in a more long term idea. These suggestions are something that the developers of the quest can use or look into at any time and are not forced to use them.

3.2 Actions and Grammar

The AI during the building process can offer a variety of options for the developer to use while creating the quest and one of which is the quest actions which are what exactly is happening in the quest and what is done. Some examples of what these quest actions would be are "Kill", "Explore", or "Spy". These individual quest actions, when brought together, form the actions that are part of the questline. The way that the AI chooses what actions to suggest is through the generative grammar and prerequisites. The prerequisites are conditions for quest actions to be allowed; for example there must be an entity that can be killed for the action "Kill" to be used or suggested. The generative grammar follows productions rules that are split into 2 categories: motivation and non-motivation. Motivation is the idea that the quest is created because the NPC is motivated to do so. An example of this would be a questline to gain reputation or knowledge for the NPC. Questlines in the non-motivation category are much more simplistic questlines. Examples of non-motivation questlines are simple commands such as to go here or get that. When using the EDD framework the developer first make the dungeon and NPCs then the Qg AI is able to either completely make a quest on its own or together with the developer. If the user decides to take the mixed initiative approach they can make the quest themselves and have the AI later suggest items they should need to get during the quests, how the storyline could continue from the said quest, or could even suggest different options the user could rewrite the quest to be.

3.3 Expressive Analysis Study

An expressive range analysis is a study of content generated by the system to see if it has diversity and the variations of the output (for the case of this study it would be the quest actions generated) by counting the number of similar outputs. Alvarez et al. performed this analysis by having Qg create 100,000 quests with a maximum of 50 quest actions. Alvarez et al. ran an Expressive range analysis where they checked how often quest actions appeared to see the diversity and expressivity of the quest actions that Qg would use. From the data that was created they measured mainly two things, how many quest actions were generated into a quest and how often the individual quest actions popped up (Figure 3). For example they have quest actions such as "Explore" to explore an area, "Kill" to go kill an enemy, and "Defend" to defend a location or NPC. The results showed that in quests that had only one quest action that "Repair" was the most common coming up 60% of the time. If the quest had around 5 actions the most common quest action was "Explore". For longer quests "Explore" and a handful of other actions are very common as one of the earlier actions for quests, but as the quest gets longer they become less and less frequent. Instead the most quest action common in the later steps were found to be "go to".

With the full results (3) it shows the percentage chance that every quest action has to occur given the quest step you are at with the beginning starting at 0. it is shown that "Explore" is the most common quest action to be recommended. But as it gets to later stages of a quest the chances for "Explore" drop from 87% to 24% while on the other hand "go to" increase its odds from 0% to 28%. The reason this is believed to occur is because at the early stages most areas are not known to the player and when the player has gained knowledge of various places they can not explore said areas and now need to "go to" instead. "Some actions are very underrepresented regardless of quest length or step number, as is the case for "Defend", "Report", "Experiment", "Escort", "Capture", and "Spy". This implies that these actions have very little chance to be suggested at any quest step, so it is more likely to end up in a quest if manually added by the designer" [1]. This is important to recognize so that in future improvements these options become more apparent in the suggestions.

3.4 Developer Use Study

The next step to testing Qg was to have users try the program and get their feedback. They had 6 participants, which were all game developers in the level and quest design field, were game development alumni and had no experience with EDD or any other mixed initiative tool. They then had 3 tasks to complete: to create a questline manually, to make a questline automatically, and lastly to make a questline using the mixed initiative approach.

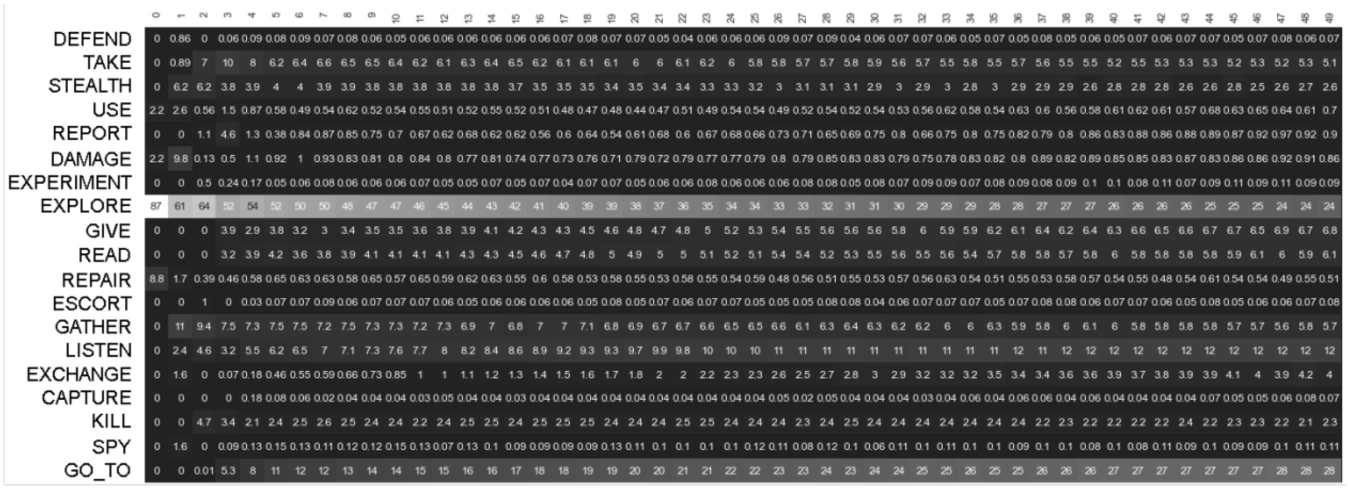
3.4.1 Manual. The participants were asked to use Qg to create a level manually, including the questline. After completing the tasks, participant feedback was recorded: Participants responded that Qg was easy to use and to understand but had complaints when making a questline that they were not able to reorder quest actions. So once they made a questline if they wanted to swap around how it worked they would have to start from scratch and rebuild the questline.

3.4.2 Automatic. The participants made the level and after which they let the Qg AI fully develop the questline. The participants talked about the usefulness of this when making games that required a lot of quests, but overall the opinions on the fully automatic questline was negative. "Most participants remarked the system as random and illogical regarding random tile picks connected with quest actions as the system picked farther away NPC and targets with no purpose" [1]. One other point of information was that participants did not like the automatic use because it felt like it took away their freedom for creating their questline and ideas.

3.4.3 Mixed Initiative. The participants this time were able to design a level and create a questline themselves but this time they had the suggestions from the Qg AI that they could use when they wished. The participants stated that it helped them in development, making the questline creation process feel quicker, and helped inspire them when they were at a blockage for what to do next. There were some concerns that when the participants got to the later stages of the questline creation they felt that the Qg suggestions lacked cohesiveness.

Some comments that were given from the participants after the mixed initiative approach talked about how they felt that the suggestions were a good way to gain inspiration. "One participant said that '[the system] suggested capturing a monster which they had thought about killing. The "capture" option might be more interesting and might have been an option I had otherwise overlooked.'" [1]. Through looking at the other comments it was concluded that most participants felt that the suggestions were good ways to gain inspiration instead of actually implementing them.

After the testing was all done the participants were asked to give comments about Quest actions, Usability, Creativity, overall experience, and any features they felt were missing. Most of the quest actions were clear and easy to understand but there were some cases where the participants had to go through trial and error to figure out what some actions meant. The participants stated that it was a useful tool for game development but questioned how it would perform in



	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
DEFEND	0	0.86	0	0.06	0.09	0.08	0.09	0.07	0.08	0.06	0.05	0.06	0.06	0.06	0.06	0.06	0.06
TAKE	0	0.89	7	10	8	6.2	6.4	6.6	6.5	6.5	6.4	6.2	6.1	6.3	6.4	6.5	6.2
STEALTH	0	6.2	6.2	3.8	3.9	4	4	3.9	3.9	3.8	3.8	3.8	3.8	3.8	3.8	3.7	3.5
USE	2.2	2.6	0.56	1.5	0.87	0.58	0.49	0.54	0.62	0.52	0.54	0.55	0.51	0.52	0.55	0.52	0.51
REPORT	0	0	1.1	4.6	1.3	0.38	0.84	0.87	0.85	0.75	0.7	0.67	0.62	0.68	0.62	0.62	0.56
DAMAGE	2.2	9.8	0.13	0.5	1.1	0.92	1	0.93	0.83	0.81	0.8	0.84	0.8	0.77	0.81	0.74	0.77
EXPERIMENT	0	0	0.5	0.24	0.17	0.05	0.06	0.08	0.06	0.06	0.06	0.07	0.05	0.05	0.07	0.05	0.07
EXPLORE	87	61	64	52	54	52	50	50	48	47	47	46	45	44	43	42	41
GIVE	0	0	0	3.9	2.9	3.8	3.2	3	3.4	3.5	3.5	3.6	3.8	3.9	4.1	4.2	4.3
READ	0	0	0	3.2	3.9	4.2	3.6	3.8	3.9	4.1	4.1	4.1	4.1	4.3	4.3	4.5	4.6
REPAIR	8.8	1.7	0.39	0.46	0.58	0.65	0.63	0.63	0.58	0.65	0.57	0.65	0.59	0.62	0.63	0.55	0.6
ESCORT	0	0	1	0	0.03	0.07	0.07	0.09	0.06	0.07	0.07	0.07	0.06	0.05	0.06	0.06	0.06
GATHER	0	11	9.4	7.5	7.3	7.5	7.5	7.2	7.5	7.3	7.3	7.2	7.3	6.9	7	6.8	7
LISTEN	0	2.4	4.6	3.2	5.5	6.2	6.5	7	7.1	7.3	7.6	7.7	8	8.2	8.4	8.6	8.9
EXCHANGE	0	1.6	0	0.07	0.18	0.46	0.55	0.59	0.66	0.73	0.85	1	1	1.1	1.2	1.3	1.4
CAPTURE	0	0	0	0	0.18	0.08	0.06	0.02	0.04	0.04	0.04	0.03	0.05	0.04	0.04	0.03	0.04
KILL	0	0	4.7	3.4	2.1	2.4	2.5	2.6	2.5	2.4	2.4	2.2	2.4	2.5	2.5	2.4	2.5
SPY	0	1.6	0	0.09	0.13	0.15	0.13	0.11	0.12	0.12	0.15	0.13	0.07	0.13	0.1	0.09	0.09
GO_TO	0	0	0.01	5.3	8	11	12	12	13	14	14	15	15	16	16	17	18

	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33
DEFEND	0.07	0.08	0.07	0.07	0.05	0.04	0.06	0.06	0.06	0.09	0.07	0.09	0.04	0.06	0.07	0.07	0.06
TAKE	6.1	6.1	6.1	6	6	6.1	6.2	6	5.8	5.8	5.7	5.7	5.8	5.9	5.6	5.7	5.5
STEALTH	3.5	3.5	3.4	3.5	3.4	3.4	3.3	3.3	3.2	3	3.1	3.1	3.1	2.9	3	2.9	3
USE	0.48	0.47	0.48	0.44	0.47	0.51	0.49	0.54	0.54	0.49	0.52	0.54	0.52	0.54	0.53	0.56	0.62
REPORT	0.6	0.64	0.54	0.61	0.68	0.6	0.67	0.68	0.66	0.73	0.71	0.65	0.69	0.75	0.8	0.66	0.75
DAMAGE	0.73	0.76	0.71	0.79	0.72	0.79	0.77	0.77	0.79	0.8	0.79	0.85	0.83	0.83	0.79	0.75	0.78
EXPERIMENT	0.04	0.07	0.07	0.05	0.06	0.06	0.08	0.06	0.06	0.06	0.08	0.08	0.05	0.08	0.07	0.09	0.09
EXPLORE	40	39	39	38	37	36	35	34	34	33	33	32	31	31	30	29	29
GIVE	4.3	4.5	4.6	4.8	4.7	4.8	5	5.2	5.3	5.4	5.5	5.6	5.6	5.6	5.8	6	5.9
READ	4.7	4.8	5	4.9	5	5	5.1	5.2	5.1	5.4	5.4	5.2	5.3	5.5	5.6	5.5	5.6
REPAIR	0.58	0.53	0.58	0.55	0.53	0.58	0.55	0.54	0.59	0.48	0.56	0.51	0.55	0.53	0.57	0.56	0.63
ESCORT	0.06	0.05	0.08	0.05	0.07	0.06	0.07	0.07	0.05	0.05	0.05	0.08	0.08	0.04	0.06	0.07	0.07
GATHER	7	7.1	6.8	6.9	6.7	6.7	6.6	6.5	6.5	6.6	6.1	6.3	6.4	6.3	6.2	6.2	6
LISTEN	9.2	9.3	9.3	9.7	9.9	9.8	10	10	10	11	11	11	11	11	11	11	11
EXCHANGE	1.5	1.6	1.7	1.8	2	2	2.2	2.3	2.3	2.6	2.5	2.7	2.8	3	2.9	3.2	3.2
CAPTURE	0.04	0.04	0.04	0.04	0.04	0.04	0.04	0.04	0.04	0.05	0.02	0.05	0.04	0.04	0.04	0.03	0.04
KILL	2.5	2.5	2.4	2.4	2.4	2.5	2.5	2.4	2.4	2.4	2.3	2.4	2.5	2.4	2.4	2.4	2.4
SPY	0.09	0.09	0.13	0.11	0.1	0.1	0.1	0.1	0.12	0.11	0.08	0.12	0.1	0.06	0.11	0.1	0.11
GO_TO	18	19	19	20	20	21	21	22	22	23	23	24	23	24	24	25	25

	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49
DEFEND	0.05	0.07	0.05	0.08	0.05	0.06	0.05	0.07	0.06	0.07	0.07	0.05	0.07	0.08	0.06	0.07
TAKE	5.8	5.5	5.7	5.6	5.5	5.5	5.2	5.5	5.3	5.3	5.3	5.2	5.3	5.2	5.3	5.1
STEALTH	2.8	3	2.9	2.9	2.9	2.6	2.8	2.8	2.8	2.6	2.6	2.8	2.5	2.6	2.7	2.6
USE	0.58	0.54	0.63	0.6	0.56	0.58	0.61	0.62	0.61	0.57	0.68	0.63	0.65	0.64	0.61	0.7
REPORT	0.8	0.75	0.82	0.79	0.8	0.86	0.83	0.88	0.86	0.88	0.89	0.87	0.92	0.97	0.92	0.9
DAMAGE	0.83	0.82	0.8	0.89	0.82	0.89	0.85	0.85	0.83	0.87	0.83	0.86	0.86	0.92	0.91	0.86
EXPERIMENT	0.07	0.08	0.09	0.08	0.09	0.1	0.1	0.08	0.11	0.07	0.09	0.11	0.09	0.11	0.09	0.09
EXPLORE	29	28	28	27	27	27	26	26	26	26	25	25	25	24	24	24
GIVE	5.9	6.2	6.1	6.4	6.2	6.4	6.3	6.6	6.5	6.6	6.7	6.7	6.5	6.9	6.7	6.8
READ	5.4	5.7	5.8	5.8	5.7	5.8	6	5.8	5.8	5.8	5.8	5.9	6.1	6	5.9	6.1
REPAIR	0.54	0.51	0.55	0.53	0.58	0.57	0.54	0.55	0.48	0.54	0.61	0.54	0.54	0.49	0.55	0.51
ESCORT	0.07	0.05	0.07	0.08	0.08	0.06	0.07	0.07	0.06	0.05	0.08	0.05	0.06	0.06	0.07	0.08
GATHER	6	6.3	5.9	5.8	6	6.1	6	5.8	5.8	5.8	5.8	5.7	5.7	5.6	5.8	5.7
LISTEN	11	11	12	11	12	12	12	12	12	12	12	12	12	12	12	12
EXCHANGE	3.2	3.5	3.4	3.4	3.6	3.6	3.9	3.7	3.8	3.9	3.9	4.1	4	3.9	4.2	4
CAPTURE	0.06	0.04	0.06	0.06	0.04	0.06	0.04	0.04	0.04	0.04	0.07	0.05	0.05	0.06	0.08	0.07
KILL	2.4	2.4	2.4	2.2	2.3	2.2	2.2	2.2	2.2	2.4	2.2	2.3	2.3	2.2	2.1	2.3
SPY	0.1	0.1	0.09	0.1	0.1	0.08	0.1	0.08	0.11	0.09	0.1	0.09	0.09	0.1	0.11	0.11
GO_TO	26	25	25	26	26	26	27	27	27	27	27	27	27	28	28	28

Figure 3. Results from the expressive analysis of Questgram (taken from [1])

different game settings like a long-term grinding game or a very short-term quick-paced game. Some participants also felt that they would not know for sure how useful it would be until they could actually test run a game through it. Overall most of the participants felt that the system helped them be more creative but mainly was a useful tool to help with ideas when they hit a blockage. They felt the tool was useful for those in the gaming community, but those with little background knowledge would find Qg difficult to use and understand. A suggestion they had to improve the system was to give the user a way to reorder the questline without having to fully remake a questline.

4 Conclusion

Procedural Quest Generation techniques are still in the development and testing stage but this is a very important step in developing the use of procedural generation with quests in video games. The authors of these papers expand on different ideas as to how the AI itself works but they both cover the ways of making quests procedurally in some way. Questgram has some faults of its own with it only having a limited amount of quest actions a user can choose from and the fact that when you want to tweak the questline you have to delete all of it and start from scratch again. These can be solved with further development of the program where they allow for easier editing and adding more quest actions to be available to the user. The studies done in the Questgram method show how there is room to improve with the change of having uncommon actions to be offered more by the AI instead of a few actions being the overly common in the suggestions. It is also shown through the use of developers that is a helpful tool to inspire even though it can sometimes limit or slow down the process.

The pipeline method itself has the issues of limited size when testing because they did not work with a fully-sized knowledge graph that it would need to be for an actual full video game since that would require a lot of work. This issue causes some cases where a player asks for something outside the scope which limits the answer to be that the NPC does not have the knowledge for it. Also, for the large language model there is the issue of the data that it was trained on has resulted in some incorrect English. For example, when the user gives an input while talking with a pirate NPC, it would randomly throw out phrases such as "Arr" or "Yarr" which is not desired. These faults can all be fixed through either creating larger knowledge graphs to expand on options or performing more thorough and detailed training on the large language model used.The studies from the pipeline method show that the method's current large language model still falls behind in the metrics novelty, creativity, fluency, and coherence (Section 2.5) but are not that much behind and are close to being as good as the hand written WoW quests. The study done on player satisfaction and responsiveness proved that even though the WoW quests were described better the pipeline method was still voted to be better at creating quests tuned to the user's input. The future outlook for these methods is that Questgram could be developed more and be used in the future as a helpful tool to help the developers of quests in games to create them faster and with more variety using the suggestions as inspiration for new ideas. For the pipeline method they talked about including open chat AI's that simulate real conversations with users alongside the personalized quest generation to make a more immersive experience for the user. The idea of procedural content generation is something that games are constantly evolving alongside with and will help each other to improve as time goes on.

Acknowledgments

Thank you to all of my professors I have had over the years and to the friends and family who have supported me through my life. Special thanks to Professor Kristin Lamberty and Professor Wenkai Guan for helping me with my research.

References

[1] Alberto Alvarez, Eric Grevillius, Elin Olsson, and Jose Font. 2021. Questgram [Qg]: Toward a Mixed-Initiative Quest Generation Tool. In *Proceedings of the 16th International Conference on the Foundations of Digital Games* (Montreal, QC, Canada) *(FDG '21)*. Association for Computing Machinery, New York, NY, USA, Article 6, 10 pages. https://doi.org/10.1145/3472538.3472544

[2] Trevor Ashby, Braden K Webb, Gregory Knapp, Jackson Searle, and Nancy Fulda. 2023. Personalized Quest and Dialogue Generation in Role-Playing Games: A Knowledge Graph- and Language Model-based Approach. In *Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems* (Hamburg, Germany) *(CHI '23)*. Association for Computing Machinery, New York, NY, USA, Article 290, 20 pages. https://doi.org/10.1145/3544548.3581441

[3] Judith van Stegeren and Jakub Myśliwiec. 2021. Fine-tuning GPT-2 on annotated RPG quests for NPC dialogue generation. In *Proceedings of the 16th International Conference on the Foundations of Digital Games* (Montreal, QC, Canada) *(FDG '21)*. Association for Computing Machinery, New York, NY, USA, Article 2, 8 pages. https://doi.org/10.1145/3472538.3472595