QuestVille: Procedural Quest Generation Using NLP Models

Suzan Al-Nassar
s.al-nassar.2@umail.leidenuniv.nl
LIACS, Leiden University
Leiden, The Netherlands

Anthonie Schaap
a.l.j.schaap@umail.leidenuniv.nl
LIACS, Leiden University
Leiden, The Netherlands

Michael van der Zwart
m.j.van.der.zwart@umail.leidenuniv.nl
LIACS, Leiden University
Leiden, The Netherlands

Mike Preuss
m.preuss@liacs.leidenuniv.nl
LIACS, Leiden University
Leiden, The Netherlands

Marcello A. Gómez-Maureira
m.a.gomezmaureira@utwente.nl
HMI, University of Twente
Enschede, The Netherlands

ABSTRACT

Developers face a time-consuming task when creating quests in video games. To ease this burden, Procedural Content Generation (PCG) techniques can be used to automatically generate quests. While PCG has been applied to various areas of game development, it can be difficult to create meaningful narratives for quests. This paper presents a new method for generating engaging quests by combining PCG with Natural Language Processing (NLP) using the models BERT and GPT-2 in a case study game called QuestVille. The paper details the implementation of these models and the challenges encountered. The results suggest that the use of BERT and GPT-2 has potential for creating compelling narrative content. Advancements in AI research may improve on the limitations discussed.

CCS CONCEPTS

• **Applied computing** → **Computer games**; • **Software and its engineering** → *Interactive games.*

KEYWORDS

procedural content generation, procedural quests, game AI

ACM Reference Format:
Suzan Al-Nassar, Anthonie Schaap, Michael van der Zwart, Mike Preuss, and Marcello A. Gómez-Maureira. 2023. QuestVille: Procedural Quest Generation Using NLP Models. In *Foundations of Digital Games 2023 (FDG 2023), April 12–14, 2023, Lisbon, Portugal.* ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/3582437.3587188

1 INTRODUCTION

Video games frequently feature "quests", small stories provided at moments in the game that are meant to engage players for completing tasks. In this context, we understand quests as a set of tasks that need to be completed to achieve a desired outcome. Creating these quests can be very time-consuming. This is where Procedural Content Generation (PCG) can serve as a tool to support game developers. PCG has a wide range of applications in video games, and we can use it for the automatic generation of engaging quests.

FDG 2023, April 12–14, 2023, Lisbon, Portugal

ACM ISBN 978-1-4503-9855-8/23/04.
https://doi.org/10.1145/3582437.3587188

Figure 1: Screenshot showing the map view of QuestVille

For this purpose, we experiment with the use of NLP models and investigate the combination of two models for quest generation. We developed QuestVille as an implementation case-study game in which a player interacts with with Non-Player Characters (NPCs) that provide quests when approached.

NLP models can generate text that involves functional game parameters (e.g. tasks and relationships) and create a context-aware narrative for players. The core of this research is that we want to improve on simple procedural tasks that are not yet engaging for the user. When feeding these to our NLP model, we generate small narratives that are hypothesized to be more engaging by way of providing engaging story details to simple task statements. For the generation of quests, we use a combination of two different NLP models. Bidirectional Encoder Representations from Transformers (BERT) predicts how likely a word is to appear in a certain context. We use BERT in PCG for quests by sampling from a list of words based on the assigned score. This then serves as prompt for another model we use: Generated Pre-trained Transformer 2 (GPT-2), which has been trained on a large corpus of English text data. GPT-2 was specifically developed for generating texts from a prompt.

In the following section we discuss the background and related work for this study, including descriptions of NLP models used. We then describe our implementation for quest generation, which is also the resulting outcome of this paper. This is followed by a discussion of the implementation. In the final conclusion of our work

we reflect on potentials and limitations of our approach, ending with suggestions for future work.

2 BACKGROUND

Natural Language Processing (NLP) is a field that combines computer science, linguistics, and artificial intelligence to study how computers can understand and interact with human language. It draws from various disciplines to bridge the gap between human language and computer understanding. While NLP involves written and spoken text, we focus on written text only. More specifically, the task of Natural Language Generation (NLG) is what is needed for procedural quest generation in this case. Recent advances in the field of NLP have made well-performing tools for NLG and other NLP tasks accessible to the public. In this project, the NLP models of BERT [4] and GPT-2 are used for procedural quest generation.

Bidirectional Encoder Representations from Transformers (BERT) is a model developed by Google that can be used for various NLP tasks. When it was first introduced, BERT outperformed other state-of-the-art models. One of the key features of BERT is its ability to be fine-tuned on any dataset, depending on the application. BERT was trained by masking certain words in a sentence and then trying to predict how to replace the masks by analyzing the sentence in both left-to-right and right-to-left directions simultaneously. This approach makes BERT extremely effective because it has a deeper understanding of the context in which words appear.

Generative Pre-trained Transformer 2 (GPT-2) is a transformer model trained on a large English corpus dataset, suitable for tasks such as translation, summarization, and text generation. By predicting the next word in a sentence, the model learns a representation of the English language. Unlike earlier NLP models that were trained through supervision, GPT models are mainly trained unsupervised, with some supervised fine-tuning at the end, which allows for larger data sets and a more robust model.

3 RELATED WORK

Some work has been done on using language models for generating game content, such as the work by Freiknecht et al. [5]. They use GPT-2 to generate a branching story based on predefined characters and game intro. They found that by using control sets and limiting the number of characters in the game, they were able to generate the most coherent stories. We think this work shows promise, but we want to focus specifically on generating stories around quests, not an entire game. Their use of control sets is interesting, as to keep the story on the right track and we use a similar strategy of a controlled start to guide the model.

DeLucia et al. [3] showed that transferring techniques from response generation to narrative generation works well because of the similarity of both tasks. Kalbiyev [6] found that some of the biggest challenges when using a fine-tuned version of GPT-2 to generate dialogue for games were grammatical correctness, semantic meaningfulness, appropriateness of the response, and the suitability of the response in the given setting. They concluded that human-generated responses still outperform AI-generated ones.

Most literature on using NLP models for narrative generation suggests that results could be improved a lot when bigger, more sophisticated models are released to the public such as GPT-3.

Figure 2: Screenshot showing a house interior

In an analysis by van Stegeren et al. [9] of a narrative generation competition, they concluded that using high-level predefined narrative leads to increased coherence. They also found that if the generator generates events that underlie the surface text, simulation-like approaches to narrative generation seem to perform well.

Different approaches to narrative generation that do not use any NLP models include, for example, using planning algorithms [1], or using a genetic algorithm [2]. Mark Riedl has done research on creating a storytelling game that is realistic in the terms that it becomes difficult to distinguish an AI agent from a human. In [8], he describes the use of multiple AI agents working with each other to create a complete and realistic story. There are many ways quests can be generated, but for the present research, we focus on the use of NLP models.

4 METHODS

Our hypothesis is that a quest will be more engaging to a user when it is based on the context of the setting and has a clear motivation for why it needs to be completed. Compare the following two quest tasks that are essentially the same:

- *Destroy this magical ring*
- *You must destroy this magical ring, that will corrupt you, by casting it into the volcano in enemy territory. Servants of the evil villain will hunt you along the way, you must not let the ring fall into their hands. You are the only hero capable of doing this!*

We argue that the second quest is more engaging than the first because it contains additional motivation and stakes as to why this 'magical ring' has to be destroyed. Our approach to procedurally generate quests using NLP models can be separated into the following five steps:

(1) *NPC name selection*
Each NPC in the game has a randomly selected name from a pre-defined list of real-world names. This way we can ensure some variety when it comes to NPC names.

(2) *Quest prompt selection*
We define a list of initially small, general quest prompts that will be used as inputs to the next steps. For example: *"Could you fetch the [MASK] from {0}, I need it to "*

Where the `[MASK]` is used for the BERT step and {0} is the name of the randomly selected NPC that will be involved in the quest. The remainder of the sentence is generated by the GPT-2 model.

(3) *NPC relation prefixing*
In the game, relations between NPCs are defined, such as "Alice hates Bob". These relations are prefixed to the quest prompt before being fed into the models of the next two steps. This is to ensure the generated text draws from this relation to generate a more context-aware quest.

(4) *BERT step*
In this step, the `[MASK]` is filled in by the BERT model. This step is explained in greater detail below.

(5) *GPT-2 step*
During this step, the output of the BERT model is fed into the GPT model, which adds additional text to the prompt. This step is also explained in greater detail below. The final result is then displayed to the user, with the NPC relation prefixes removed.

BERT can predict how likely a word is based on the other words around it. Thus, a certain context can be defined with a masked word that needs to be predicted. A sample sentence is defined, with one word masked, which is fed into the BERT model and a top 5 of best-scored words is returned. We use this to randomly sample words with scores used as indicator of the probability. Thus, we can pre-define sentences and run them through the BERT model to dynamically obtain varying engaging sentences to use in quests.

```
I need to bring the [MASK] to Bob because
it is very important to him.
```

GPT-2 can generate text from a prompt. We can predefine parts of sentences, and run them through the GPT-2 model. An example is shown in Figure 3, with in bold how the initial sentence is completed by the GPT-2 model.

```
I need to bring the letter to Bob because
that's the only piece that was out there
with this kind of information.
```

Figure 3: GPT-2 example

There is a trade-off between how specific the input sentence to the GPT-2 model is and how fitting the resulting output of the model is in the given context. Since a goal of procedurally generating quests could be to save time and effort for any writers, the input sentences should be as small as possible. But the less information is present in the input, the harder it is for the GPT-2 model to generate context-aware output.

For the actual quest generation, we use a combination of both the BERT and the GPT-2 model in sequence. At first, we define (part of) a sentence and mask a word in it. The result is fed into the BERT model, which provides a predicted top 5 list of most likely words to appear in the place of the mask. We randomly sample the list with the scores as probabilities to fill the gap. Next, GPT-2 takes the BERT result as input to the model, and the resulting output is used as the final quest. Thus, combining both models in this way can lead to an increase in variety since the prompt can have different variations. The cutoff of the top 5 BERT scores is chosen rather arbitrarily, a larger or smaller top n is also possible.

To be able to generate quests inside a video game using NLP models, we created a game in Unity called QuestVille (Figures 1 and 2). A player is able to walk through a map and can enter houses to meet NPCs. When the player approaches the NPC, a quest is provided on the screen. The game has three different houses and 5 different NPCs, that know each other's name and give the player a task, which is bringing a specific item to one of the other NPCs with an explanation added to the quest.

The resulting quest is then evaluated by us. We check if the generated quests make sense but also if it is in the right context. For example, if a woman is the receiver of the quest, are the pronouns then correctly assigned, and is the name correctly assigned? We also check if the generated text contains a motivation for the initial quest prompt.

5 RESULTS & DISCUSSION

When creating quests with GPT-2, we should take into account what input we provide to the model and the length of said input. The output of the model might be too incoherent when the input is too vague or general and does not steer the model toward a direction. However, we want to obtain the most engaging quests while providing as little input as possible, which would save time. An example of the least effort is shown in Figure 4, with just a short input sentence of three words. Notice that no actual meaningful output is generated by the model, as it is not clear for what reason exactly Isabella needs help. It provides no additional context or motivations, making the quest just as basic as before we put it into the model. Thus, we can use some additional input to steer our model in a certain direction.

```
Please help Isabella in her
time of need.
```

Figure 4: GPT-2 quest with no steering

Results with more steering in comparison to Figure 4 are shown in Figures 5 and 6. We observe that a small amount of steering can already result in a more meaningful output, as the result of Figure 5 contains an actual action and motivation, and a desired result can be achieved. It is more clear to the user what actions the game expects from them, and why a certain goal needs to be achieved. With even more additional steering, as can be seen in Figure 6, we obtain a quest that contains additional context-aware motivations and tasks.

```
Please help Isabella find her way back
home as quickly as possible.
```

Figure 5: GPT-2 quest with little steering

```
Please help Isabella find ingredients
to make this amazing, gorgeous pie.
```

Figure 6: GPT-2 quest with more steering

However, we need to prevent the input from being too specific, as we also desire the model to be diverse, such that it does not generate likely quests all the time. An example of a too-specific input is shown in Figure 7. Note that only the ingredients needed for the pie are varying between the quests, but the quest itself comes down to the same task.

```
Help Isabella find ingredients for a pie with
strawberries, apricos and a touch of mustard.

Help Isabella find ingredients for a pie with
a side of red pepper pie and some black cheese.
```

Figure 7: GPT-2 diversity

The principle of steering the model in a direction also applies to the relations that different NPC characters can have in the game. By prefixing the model input with information about these relations, the output can be steered towards a result that makes sense based on the relation between two NPCs. For example in Figure 8, the same prompt of "Bring Alice the keys so John can " is fed into the model twice, but the second time it is prefixed by "Alice hates John". As can be seen, the part that is generated by the model has a way more negative attitude when prefixed by this "hate" relation between the two NPCs.

```
Bring Alice the keys so John can go back in
time. In fact, Alice is just after the key
from her sister.

Alice hates John. Bring Alice the keys so
John can take them. John will get  upset
and will attack her first.
```

Figure 8: GPT-2 relation prefix example

When playing the game, the main limitations we can observe from the generated quests are the following:

- The end result is not always a coherent sentence that makes sense in the given context.
- The output can sometimes be inappropriate for a younger audience.
- Because of an enforced limit on the number of generated characters, sometimes sentences will abruptly end. We try to cut off sentences at the last dot but this is not that simple in practice (dots do not always signal the end of a sentence).

Some positive points we observe are:

- There are instances in which a very high quality, coherent sentence with a proper motivation to the quest is produced.
- Adding relation prefixes to the prompt very often leads to the resulting sentence being steered towards this relation.
- BERT seems to find very suitable words to fill the masked word in a lot of cases.

Some of the issues such as incoherent sentences being generated could be solved if, very plainly, the model was just better. Since we started our work, newer models have been released which we will touch upon in future work.

6 CONCLUSION

Our approach to procedurally generating quests using BERT and GPT-2 was discussed and shown to have potential, but also involved considerable limitations. In many instances, useful results were produced but useless or inappropriate sentences got generated as well. We found that prefixing prompts to the models with relations between NPCs to steer the model in a certain direction worked very well. From this, we conclude that NLP approaches are not yet good enough for use in commercial video game development, but show a lot of promise. Further advances in the field of NLP should make such approaches even more viable.

Future research could improve the quality of a generated quest by validating whether it actually has a specific goal included that needs to be accomplished. A possible approach for this could be a hybrid model that combines rule-based and machine-learning algorithms. The rule first identifies segments of the quest corresponding with actions and goals. Consequently, the machine learning algorithm learns to output a probability on whether the input is actually classified as a quest or not. The machine learning algorithm could be applied to numerical representations of the input data generated by term frequency-inverse document frequency and Word2Vec. This approach would improve the engagement of our quests.

Furthermore, we want to incorporate a new technique that has been recently released. OpenAI latest chatbot, ChatGPT [7], allows users to interact in a natural way, resulting in often comprehensive and complete answers by the system. It is also possible to create a story with ChatGPT by telling the bot to act like a person in need of something. This bot is trained on a very large dataset with information until 2021, therefore it is able to talk on a wide spectrum of subjects and act closely as a human would. We believe that this tool could also be used to generate better quests than GPT-2 can.

REFERENCES

[1] Vincent Breault, Sébastien Ouellet, and Jim Davies. 2021. Let CONAN tell you a story: Procedural quest generation. *Entertainment Computing* 38 (2021), 100422. https://doi.org/10.1016/j.entcom.2021.100422

[2] Edirlei Soares de Lima, Bruno Feijó, and Antonio L. Furtado. 2022. Procedural generation of branching quests for games. *Entertainment Computing* 43 (2022), 100491. https://doi.org/10.1016/j.entcom.2022.100491

[3] Alexandra DeLucia, Aaron Mueller, Xiang Lisa Li, and João Sedoc. 2020. Decoding Methods for Neural Narrative Generation. https://doi.org/10.48550/ARXIV.2010.07375

[4] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805* (2018).

[5] Jonas Freiknecht and Wolfgang Effelsberg. 2020. Procedural Generation of Interactive Stories Using Language Models. In *International Conference on the Foundations of Digital Games* (Bugibba, Malta) *(FDG '20)*. Association for Computing Machinery, New York, NY, USA, Article 97, 8 pages. https://doi.org/10.1145/3402942.3409599

[6] A. Kalbiyev. 2022. Affective dialogue generation for video games. http://essay.utwente.nl/89325/

[7] OpenAI. 2023. ChatGPT. https://chat.openai.com/chat

[8] Mark Riedl and R. Young. 2004. An Intent-Driven Planner for Multi-Agent Story Generation. (07 2004). https://doi.org/10.1109/AAMAS.2004.63

[9] Judith van Stegeren and Mariët Theune. 2019. Narrative Generation in the Wild: Methods from NaNoGenMo. In *Proceedings of the Second Workshop on Storytelling*. Association for Computational Linguistics, Florence, Italy, 65–74. https://doi.org/10.18653/v1/W19-3407