Towards Generating Textual Game Assets from Real-World Data

Judith van Stegeren
University of Twente
Enschede, The Netherlands
j.e.vanstegeren@utwente.nl

Mariët Theune
University of Twente
Enschede, The Netherlands
m.theune@utwente.nl

ABSTRACT

We propose using real-world datasets to generate textual game assets for serious games. As an example, we used a dataset of P2000 crisis event messages to generate descriptive texts that can be transformed into new game assets by game writers, thereby reducing the writing effort required during the development phase of an adaptive serious game. In this paper we describe this first attempt and we discuss the challenges and possibilities of using open data for textual asset generation.

CCS CONCEPTS

• **Computing methodologies** → **Natural language generation**; • **Applied computing** → *Computer games*; • **Software and its engineering** → *Interactive games*;

KEYWORDS

serious games, procedural content generation, natural language generation, real-world data

ACM Reference Format:
Judith van Stegeren and Mariët Theune. 2018. Towards Generating Textual Game Assets from Real-World Data. In *Foundations of Digital Games 2018 (FDG18), August 7–10, 2018, Malmö, Sweden.* ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/3235765.3235809

1 INTRODUCTION

Serious games, i.e. computer games with a defined purpose other than pure entertainment [2], are a useful supplement to educational activities in a variety of fields. In some cases, serious games might even have advantages over more traditional methods of education. For example, fire fighters can use serious games to experience situations that may not easily be recreated in the real world due to ethical concerns and cost and time constraints [14].

Most games ship with a pre-made set of game elements, which will not be changed once the development phase of the game is finished. When playing the game, all players will be presented with the same game content. However, as there might be differences in player abilities and preferences, not all game content will fit every player. For games with an educational purpose, the aforementioned problem is even more acute [9]. For example, players might differ in educational background, skill level, or preferred learning style, and might need game content to suit their specific needs. If this extra content is not available, the differences between player needs and game content might decrease the effectiveness of the training aspect of the game.

A possible solution to this problem is incorporating player-centered adaptivity. An adaptive game changes elements of the game based on the player. For example, the player could be presented with different game material based on his or her in-game performance.

An adaptive game requires more content than a regular game, in order to cater to the various player types. This means that the developers have to create more game content during the development phase to accommodate for the adaptivity. Procedural Content Generation (PCG), or the creation of content automatically through algorithmic means, is a potential solution for developing adaptive games [15]. It can be used for creating any kind of game content, i.e., all aspects of a game that influence player experience but are not non-player character behaviour or the game engine itself. [15]

Using PCG for an adaptive game has the advantage that a generator can create some of the game elements, which might be less expensive than developing the game assets by hand. Provided we take adaptivity and, in the case of serious games, the educational purpose of the game into account, a generator can create variations of game assets to make the game fit for different types of players. In order to do this successfully, developers should formulate a steering purpose for the adaptivity. [9] This adaptivity goal can be used to give direction to the implementation of the generator and serve as a criterion for evaluating the generated result. The generator can be used as an authoring aid or for generating game contents that can be placed directly into the game.

Togelius et al. [11] distinguish between online and offline content generation. Online generation happens during the runtime of the game, whereas offline generation happens during game development. In the present paper, we will only consider offline content generation.

Game developers in academic and commercial settings are currently using procedural content generation to create game elements such as game levels, textures, 3D objects and game worlds [7]. PCG can also be used to generate textual game elements. For example, in Dwarf Fortress [1] an extensive textual history is generated for each new world in the game, including locations, notable figures and important events.

In this research, we present our first steps in generating textual game assets for serious games for the crisis response field. As a starting point, we have chosen to use real-world data as a base for our generated text. The idea is that the resulting texts can be used by game designers and game writers when creating new content.

In Section 2, we describe the type of serious games we are targeting with our work. In Section 3, we present our approach to generating textual game assets from real-world data. In Section 4,

FDG'18, August 7-10, 2018, Malmö, Sweden

ACM ISBN 978-1-4503-6571-0/18/08.
https://doi.org/10.1145/3235765.3235809

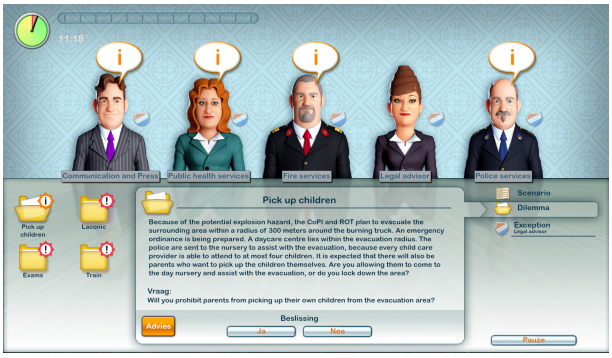

Figure 1: Screenshot of the Mayors game

we present a concrete example using real-world data from the crisis domain. We end with a discussion of challenges and possibilities in Section 5.

2 BACKGROUND

This research is part of the Data2Game project. Within Data2Game, we want to develop a serious game for the crisis domain, for training crisis officials. The goal of the game is to train these professionals in the competencies required by their job, such as decision-making under pressure and coordinating the various crisis organisations. The serious game under development is a text-driven dilemma-based serious game. An example of a game in this style is the "Mayors game" [12], a dilemma-based narrative game that is used for training new mayors in the Netherlands in decision-making and leadership skills. For a screenshot of the game, see Figure 1. In the Mayors game, the player is offered a series of dilemmas in the context of a crisis scenario situated in a municipality. The player can obtain advice from a group of virtual advisors (NPCs) that represent various supporting organisations, such as the fire department, police service and public health service. Based on the choices of the player, the game creates an assessment of the decision-making skills and the favored leadership style of the player.

In a narrative game such as the Mayors game, text is the driving element. We might even describe the Mayors game as a text-based game, since in the absence of graphics or a graphical user interface, the game would still be playable [8]. Naturally, the most important textual game assets are the dilemmas. They are presented in the context of a scenario or situational description, which is a fictional narrative, for example in the form of a short story, an eye-witness account or a newspaper article. In addition to the dilemmas and their context, the game presents advice or opinions from all NPCs to the player in the form of short monologues. These provide extra information that could help the player make a decision. NPCs try to persuade the player to take the course of action that matches the goals and interests of the organisation they represent, just as in real life. Manually creating all these textual assets requires substantial effort during the development phase of the game.

3 PROPOSED APPROACH

We expect that a game with realistic content, which exposes players to situations they would encounter in the real world, will have a positive effect on the learning experience of the players. Therefore our game should contain assets that contribute to the learning goal and are realistic for the target group.

We want to use procedural content generation to obtain game assets for the game under development. Since the game should also be adaptive, we will need a large amount of content. A generator can reduce the effort required by the game designers, provided the development of the generator outweighs the cost of writing all content by hand.

We expect that taking real-world data as input for the generator will lead to increased realism in the generated game assets. Westera et al. [13] already suggested extending virtual game worlds with additional data, such as weather reports or share prices, to enhance the dynamics and the authenticity of the game. Our approach of generating game assets from datasets is also related to the *data games* proposed by Friberger et al. Data games use real world information (such as open data) to automatically generate game content. [5]

Using real-world data to generate game assets is especially useful in the context of serious games. If enough real-world data related to the target domain is available, we can use this to reduce the authoring effort required by the game designers to create believable game content.

Since this research is performed in the context of serious games for the crisis response field, we will focus on this specific application as an example. We have experimented with real-world events from the crisis domain to test the feasibility of our approach.

As an example, in the next section we will describe a first experiment with game asset generation from real-world data in the context of our serious game.

4 GAME ASSET GENERATION EXAMPLE

Our goal is to develop a game similar to the Mayors game. Our target audience consists of management level employees in the crisis response field, who we want to train in the competencies relevant for their job. Mavromoustakos et al. have described our initial design for the serious game under development. For a description of the architecture and the gameplay, see [10]. We have made the first steps in generating natural language texts based on real-world events, that can be transformed by game writers to serve as a context for the dilemmas presented in the game. Because of our target audience, we took a dataset with real-world crisis events[1] as a starting point. Note that it is not our goal to create non-fiction texts. We only want to use the dataset to generate realistic fiction.

The dataset consists of a collection of crisis response messages from P2000. The P2000 protocol is part of the Dutch emergency communication network. Crisis response organisations such as the police, ambulance services and the fire and rescue services use it to communicate with each other and the various Dutch public-safety answering points. The messages are comparable to text messages with a specified format; they consist of a header with a timestamp, region code, target service (ambulance, police, coast guard, etc.), priority classification and a message body. For an example of a P2000 event, see Figure 2.

[1] https://www.livep2000.nl

```
17:06:00 08-02-18 17 AMBU B1 AMBU 17302 MAASSTADWEG
        3079DZ ROTTERDAM ROTTDM bon 14294
```

Figure 2: A P2000 message

Message header	
17:06:00	time
08-02-18	date
17	region (Rotterdam-Rijnmond)
AMBU	receiver (ambulance)
B1	priority (low)
Message body	
AMBU 17302	vehicle identifier
MAASSTADWEG	location
3079DZ	location
ROTTERDAM	location
ROTTDM	location
bon 14294	unknown identifier

Figure 3: The parser assigns labels to the various parts of the P2000 message from Figure 2.

Currently, the generator for natural language text consists of a simple templating engine. We parsed the P2000 events by labeling the different message parts with the terminals of a context-free grammar. For an example of a labeled P2000 code, see Figure 3. The labels of the parsed event code are then used to generate a natural language text. At the moment, the natural language generation step consists of a straight-forward templating engine that fills in the blanks in a template text based on the labels of the P2000 message. For example, we can apply the labeled data from Figure 3 to the template shown in Figure 4 (left). This way we obtain the natural language text shown in Figure 4 (right).

The texts we can generate thus far are fairly short, since they are based on the labels of only one P2000 message. However, by aggregating data from other public sources, we can extend this basic text into a more complex situational description. For example, we can use the Google Maps API for routes[2] to get an estimate for the time it would take a crisis response vehicle to arrive at the crisis location. After obtaining the information from the API, we can incorporate it in the text. For example, continuing with the P2000 code from Figure 2, we queried Google Maps for the closest ambulance service point near the event location. This turned out to be the ambulance service point at the Spuistraat in Amsterdam, which means this address could be a realistic starting point for an ambulance if such a crisis would occur in the real world. Subsequently, we queried Google Maps for the time it would take a car (ambulance) to travel from that point to the crisis location.

Although we do not have the complete Google Maps dataset, the API allows us to incorporate this information fully automatically. This makes the use of API data very suitable for procedural content generation. The queries for the Google Maps API can be constructed algorithmically, provided the underlying P2000 code contains location data. We can transform the related data from Google Maps into an additional sentence for our output text, such as "The ambulance arrives 13 minutes after receiving the emergency call."

This is only one example of aggregating more data to supplement our initial dataset. Because data APIs and open datasets become increasingly available [4], we believe that there are many other possible datasets that can be used to extend the generated descriptions.

5 DISCUSSION

Above, we have presented our first preliminary results in generating textual game assets for serious games from real-world data. We can use these texts to reduce development effort for game designers and adapt the game to the player. At the moment, we see two different applications for the generated texts.

Firstly, the generated texts can be used as a starting point for the *context description* of a dilemma. For example, for a dilemma about rescuing people during a large fire, we could choose a corresponding P2000 message about a fire, change some of the labels to make the data fit the dilemma, and generate a text. Conversely, we could start by generating a text from a random P2000 message, and use it as a starting point for writing new dilemmas.

Secondly, the generated texts can function as contextual noise during gameplay. The current prototype of our game, which is very similar to the Mayors game, tries to train players in analysing information under pressure. We plan on extending the gameplay with a stream of messages, similar to a social media stream. A small part of the messages will be relevant to the current dilemma, and offer additional information about the storyline. The rest of the messages will function as noise, distracting the player and drawing on their critical analysis skills. Authoring the noise is an extra task for the game designers, who would rather focus on writing the small set of relevant messages. The outcome of this research could be used as a first step in automatically generating these messages instead.

There are various ways we can use generated game content to make the game adaptive and improve the player experience. Firstly, players could be presented with content that is based on P2000 messages from their region of origin. This way, the game will feature locations or events that are recognizable from real life. Furthermore, we could offer players different game content based on their amount of job experience. We could do this by ranking the P2000 messages in the dataset from *common* (for events that happen daily) to *rare* (for events that happen only once a year or less). Game content based on common occurrences could be considered as 'easy' training game material. The idea is that common events happen so often that only a player with limited job experience will find them challenging. Content based on rare events will provide more difficult training material, as even experienced players will rarely have encountered these events in their work. We expect that offering game content with varying levels of difficulty will improve the effectiveness of the training game.

If we use the generated texts as contextual noise, we can make the game adaptive by varying the amount of noise in the message stream according to the performance of the player.

[2] https://cloud.google.com/maps-platform/routes/

```
On the [time] of [date], the Dutch [service] service
for the region of [region] receives a P2000 message.
[activity] is requested for [reason] at [location].
```

"On the evening of February 8th, the Dutch ambulance service for the region of Rotterdam-Rijnmond receives a P2000 message. An ambulance is requested for transport services at the Maasstadweg in Rotterdam."

Figure 4: Example template (left) and generated text (right)

Using a dataset as a source for generating game content has its limitations. One drawback of the dataset we used is that every P2000 event stands on its own. Since there are no relations present between the messages in the dataset, it is difficult to generate more complex or longer texts from them. Thus, we plan to extend the P2000 event codes with additional information from other sources, in order to build a more complete description of a crisis event. Our integration of Google Maps data is a first step in this direction.

During the course of this project, we ran into problems related to using (open) data in general, similar to the issues described by Ding et al. [3] Non-trivial manual effort was required before we could use the dataset. First we had to deal with problems such as incomplete information and the use of special codes and acronyms. For example, our parser cannot parse the jargon and other domain-specific terms that are present in the P2000 messages. Those terms should be either labeled manually, which requires domain knowledge, or removed from the P2000 dataset during clean-up. The latter has as a drawback that the system can only interpret (and thus generate texts from) the most basic P2000 messages, which leads to predictable and uninteresting output.

At the moment, we have manually assembled a small dictionary of crisis management jargon which we use to parse a small part of the dataset. However, we think that automatically gathering information about jargon could be used to mitigate the challenges related to parsing domain-specific terms. On the Internet we can find various dictionaries, such as a list of commonly used P2000 abbreviations and their meaning[3] and a list of known vehicle call signs[4]. Linking these sources of information to our parser can probably help us decipher the message body of each P2000 event.

In addition to domain-specific jargon, we need to parse the location information in the P2000 messages. Currently, location indicators such as street and city names have to be labeled manually, as the parser cannot recognize them automatically. This can easily addressed by adding another dataset to the system, namely the database of the Dutch registry for real estate, which is available as open data[5].

Despite these challenges, we think that it is feasible to generate textual game assets from open or public datasets. The next steps for this research project will consist of improving the parser, to extract more meaningful information from the dataset, and adding relevant external data sources to improve the output text. Furthermore we plan on extending the game with a stream of messages, as explained above. We will also evaluate our findings with both game developers and players, to assess the usefulness of the generated game elements. Within Data2Game, a pilot study was already conducted for measuring player stress and skill levels, from both physiological sensors and in-game player actions. In future, we can use the findings from this pilot to measure whether incorporating adaptivity using generated game assets improves the learning experience.

As future work, we are planning to look into more advanced natural language generation techniques [6] to replace the simple templating system and generate more varied and complex texts. In subsequent research, we also plan to focus more on affective language generation to improve the player experience. A study of domain-specific jargon, in order to make the generated texts more recognizable and thus relatable for players, could be a step in this direction.

[3] https://alarmeringen.nl/hulpdiensten/p2000-afkortingen/
[4] https://www.tomzulu10capcodes.nl/
[5] https://data.overheid.nl/data/dataset/basisregistratie-adressen-en-gebouwen--bag-

6 ACKNOWLEDGEMENT

This research is supported by the Netherlands Organisation for Scientific Research (NWO) via the Data2Game project (project number 055.16.114).

REFERENCES

[1] Tarn Adams and Zach Adams. 2002. *Dwarf Fortress.* Game [PC]. Bay 12 Games, Silverdale, WA, USA. Played May 2018.
[2] P. Backlund, H. Engstrom, C. Hammar, M. Johannesson, and M. Lebram. 2007. Sidh—a game based firefighter training simulation. In *Information Visualization, 2007. IV'07. 11th International Conference.* IEEE, 899–907.
[3] Li Ding, Deborah L. McGuinness, James R. Michaelis, and Jim Hendler. 2010. Making sense of open government data. In *Proceedings of the WebSci10: Extending the Frontiers of Society On-Line.*
[4] Marie Gustafsson Friberger and Julian Togelius. 2012. Generating interesting monopoly boards from open data. In *Computational Intelligence and Games (CIG), 2012 IEEE Conference on.* IEEE, 288–295.
[5] M. Gustafsson Friberger, Julian Togelius, A. Borg Cardona, Michele Ermacora, Anders Mousten, M. Møller Jensen, V. Tanase, and Ulrik Brøndsted. 2013. Data games. In *4th Workshop on Procedural Content Generation.* ACM, 1–8.
[6] Albert Gatt and Emiel Krahmer. 2018. Survey of the State of the Art in Natural Language Generation: Core tasks, applications and evaluation. *Journal of Artificial Intelligence Research* 61 (2018), 65–170.
[7] Mark Hendrikx, Sebastiaan Meijer, Joeri Van Der Velden, and Alexandru Iosup. 2013. Procedural content generation for games: A survey. *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM)* 9, 1 (2013), 1.
[8] Michael Heron. 2013. "Likely to be eaten by a Grue"—the relevance of text games in the modern era. *The Computer Games Journal* 2, 1 (2013), 55–67.
[9] Ricardo Lopes and Rafael Bidarra. 2011. Adaptivity challenges in games and simulations: a survey. *IEEE Transactions on Computational Intelligence and AI in Games* 3, 2 (2011), 85–99.
[10] Paris Mavromoustakos-Blom, Sander Bakkes, and Pieter Spronck. 2018. Personalized Crisis Management Training on a Tablet. In *Proceedings of Foundations of Digital Games.* ACM.
[11] Julian Togelius, Georgios N. Yannakakis, Kenneth O. Stanley, and Cameron Browne. 2011. Search-based procedural content generation: A taxonomy and survey. *IEEE Transactions on Computational Intelligence and AI in Games* 3, 3 (2011), 172–186.
[12] Josine G.M. van de Ven, Hester Stubbé, and Micah Hrehovcsik. 2013. Gaming for policy makers: It's serious!. In *International Conference on Games and Learning Alliance.* Springer, 376–382.
[13] Wim Westera, R.J. Nadolski, Hans G.K. Hummel, and Iwan G.J.H. Wopereis. 2008. Serious games for higher education: a framework for reducing design complexity. *Journal of Computer Assisted Learning* 24, 5 (2008), 420–432.
[14] F. Michael Williams-Bell, B. Kapralos, A. Hogue, B.M. Murphy, and E.J. Weckman. 2015. Using serious games and virtual simulation for training in the fire service: a review. *Fire Technology* 51, 3 (2015), 553–584.
[15] Georgios N. Yannakakis and Julian Togelius. 2011. Experience-driven procedural content generation. *IEEE Transactions on Affective Computing* 2, 3 (2011), 147–161.