Toolkit: Writing with Text Generators

A practical guide for writers

Table of Contents

1. Introduction

This toolkit is intended for writers who have never worked with text generators but are curious as to how they might be applied to their work or benefit their writing. The guide assumes no previous knowledge of text generation or coding. However, it attempts to be low code rather than no code and assumes a basic understanding of common interactive fiction (IF) authoring tools such as Bitsy, Twine (Harlowe) and Ink. Users of this toolkit may want to familiarise themselves with at least one of these tools prior to commencing.

The approaches outlined in this guide are by no means exhaustive or prescriptive and are instead intended as a starting point for those wanting to embark on an exploration of some of the potential applications of text generation in relation to creative writing and/or interactive narrative.

1.1 Using InGAME Toolkits

The remainder of this guide includes 3 main sections, plus a series of additional documents (elsewhere on the site) which make up the toolkit. These additions can be mixed and matched along with those from other toolkits to build a custom collection of the resources you need. Words in **bold blue** are defined in the Glossary. If you feel something is missing from the existing toolkits or would like to suggest a topic for InGAME toolkits to cover, please contact: enquiries@innovationforgames.com

1.1.1 In this Document

Section 2
Describes what the guide means by text generation, offers some examples of generative writing to read and play, and suggests some tools to create similar effects.

Section 3
Suggests some experiments to undertake with text generators to increase your understanding.

Section 4
Suggests how you might build on your experiments and offers some areas for further study.

1.1.2 Additional Documents

Quickstart Guide
A summary of the experimentation method used across these toolkits.

Bibliography
Full list of and links to all works cited, plus additional sites and resources which may be useful.

Glossary
A glossary of terms used across the toolkits.

Background Information
A brief overview of the researcher and research behind this toolkit.

Text Gen Case Studies
Case studies of some text generation experiments undertaken during the development of this toolkit.

2. What is a Text Generator and Why Use One?

Text generation is a broad umbrella term for lots of different methods of creating texts using rule systems. Some of the AI-assisted writing discussed in the Writing With and Writing For AI Toolkits might be considered forms of text generation, but so too could the novelist William Burrough's 'cut-up' technique which involved chopping up magazine articles, book pages and the like, and reconfiguring them into new stories.

"[…] a system fundamentally dedicated to possibility, variation, contingency, and play […]"

Anastasia Salter & Stuart Moulthrop (2021) Twining: Critical and Creative Approaches to Hypertext Narratives, p. 203.

As the quote above suggests, there are multiple possibilities associated with using text generation. The 'possibility and variation' element can help increase replayability or re-readability because it each read or playthrough could potentially be totally different, or be responsive to the reader-player's actions. 'Contingency' can reduce labour for writers, generating the banal but necessary text needed in videogames, for example, or creating lots of story ideas which the writer can then select or discard at will. Alternatively, it can increase labour for writers, with large volumes of authored material required in order to ensure suitable reconfiguration of material to fit a variety of player choices and/or expectations. While time-consuming, this type of generation can lead to surprising and unexpected stories for both writers and their audiences. Whether major or minor in nature, generated text offers a method of 'play[ing]' with both narrative and audience.

In videogames, this method of text generation is often called procedural narrative generation or procgen. A further benefit of text generation is that it can offer a degree of anonymity to writers, because it is not always clear to the reader which elements of a text are generated by the system and which are pre-authored. Even where large amounts of pre-authored text are used, it is impossible for the reader to know whether they are seeing the whole text, and/or whether the author 'intended' the particular configuration presented.

2.1 Examples

The following section provides a few examples which make use of generated text and/or narrative. Note that 'system' does not necessarily mean 'digital' or 'computerised' – it simply refers to a rule-based method of creating stories.

2.1.1 San Tilapian Studies

San Tilapian Studies: https://emshort.blog/2012/08/30/san-tilapian-studies-a-casual-narrative-entertainment-for-30-40-players/

Format: Pen, paper, stickers

Creator: Emily Short

Premise: Players are cast as scholars attempting to reconstruct the contents of a museum from its scattered partial remains.

Approach: Intended as a party/ice-breaker activity, players each receive stickers which relate to an archaeological object. Three sticker types are needed to describe an object, and each player is given two stickers of only one type, so must talk to other players to determine the best matches for their sticker(s). Some stickers are better suited to one another than others, but the players themselves determine how much effort they wish to dedicate to explaining how the items could relate to each other. They write down this argument in a book which then serves as a reminder of the day. This method is analogue (using physical objects – papers, stickers, pens etc) and could be described as '**emergent narrative**' – the writer has provided a system with which to create stories, rather than a story per se.

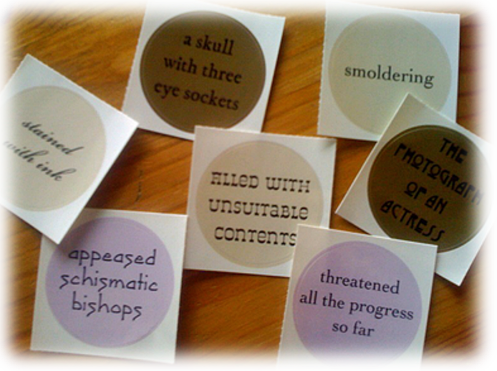
a skull with three eye sockets
smoldering
stained with ink
filled with unsuitable contents
THE PHOTOGRAPH OF AN ACTRESS
appeased schismatic bishops
threatened all the progress so far

Figure 1: Stickers from Emily Short's San Tilapian Studies

2.1.2 The Master of the Land

The Master of the Land: https://pseudavid.itch.io/the-master-of-the-land

Format: Twee2 (a Twine variant)

Creator: Pseudavid

Premise: Set at a lavish 19th Century ball, a woman with an anxiety disorder must navigate social, cultural and physical landscapes. She has several priorities which the reader may choose to explore or ignore – uncovering a possible plot against her family, convincing the Mayor to permit her to pursue her career goals, or uncovering the truth about strange happenings.

Approach: *Master of the Land* took an advanced simulation approach in Twee2 some time before the Storylets system existed within Twine to aid the creation of such simulations.

Time moves on around the player-character, Irene, and this causes changes to the location and behaviour of other characters, the weather, the party's events and so on. Choices are made by clicking navigational buttons which move the character through the space and selecting dialogue options to talk to characters, and there seem to be a huge number of possible configurations of the main text, although the beginning is the same and there are several pre-authored endings. It's difficult to say precisely how the generation works, since the author has shared few details, although they have given hints that indicate there is a **storylet** system of some kind involved.

2.1.3. Language is a Virus

Language is a Virus: http://www.languageisavirus.com/index.php#.YL4_7TZKj0o

Format: Web

Creators: SugarDazzle

Premise: A website collecting together a huge variety of generators, text manipulators, writing techniques and articles relating to experimentation and inspiration.

Approach: Rather than a single approach, *Language is a Virus* demonstrates almost every imaginable use of text generation outside of procgen narratives or videogames.

"In their simplest conception, "cut-ups" are nothing more than the random graphic rearrangement of words and phrases."

Anne Friedberg, (1991) '"Cut-ups": A Synema of the text', in William S. Burroughs at the Front: Critical Reception, 1959-1989, p. 169.

2.1.4. Zoo of Unnamed Creatures

Zoo of Unnamed Creatures: https://github.com/incobalt/Trice

Format: Trice

Creator: Michael Thomét

Premise: The reader-player wanders around a zoo with a variety of environments, each of which house strange creatures.

Approach: *Zoo of Unnamed Creatures* is a tech demo of a simple narrative world simulation rather than a fully-fledged game or narrative, but demonstrates how Trice's system of grammars can create an ever-shifting environment with lots of variety. For a more detailed analysis of Trice and grammars, see the first Text Gen Case Study.

2.2 Conclusions

These examples demonstrate three main methods of narrative generation. A systems approach, where the writer creates the conditions (or even just inspiration) for story to happen with a greater or lesser degree of pre-authored text (e.g. *San Tilapian Studies* & *Language is a Virus*); A grammatical approach, where text is reconfigured at a word, sentence or phrase level, and may be wildly incoherent; ambiguous fragments which require the reader-player to fill in the blanks; or carefully constructed to reconfigure in multiple ways (e.g. *Zoo of Unnamed Creatures*); and a **storylet** approach (e.g. *Master of the Land*), where groups of characters, locations and events (with varying degrees of pre-authored content) are presented to the reader-player either at random, or as a result of their previous choices, selections and interactions (or in combination). It is also worth noting again that generated texts do not necessarily have to be computational or even digital – elements such as paper notes and dice rolls can be incorporated too.

2.3 Potential Tools & Methods

Anything which has a shuffle function of some kind can be used as a text generator. This functionality can even be made manually, using, e.g. an excel chart and some variables. For the purposes of this toolkit we're using some common IF tools and text generation methods simply because they're widely used and easily accessible, but really anything is possible. This section gives more detail on getting started with some text generation tools and methods. Although specific usage examples are given, the methods described here could be applied to other authoring tools. For further examples and ideas, see section 3 and the Text Gen Case Studies.

2.3.1 Tables

Tables are probably the simplest version of a generative narrative system. They allow the writer to taken on the role of narrative designer rather than author and can be digital or analogue. They can also be used as a starting point to design more complex narrative systems. (See Case Studies.) Generative systems in this format are sometimes used maliciously to obtain sensitive information, so think carefully about the information you request (and share) via this method.

Public Disgrace Generator: https://www.mcsweeneys.net/articles/public-disgrace-generator

Creators: Colin Heasley and Emma Brewer

Overview: Readers configure their own personalised phrase based on the initials of their name and street, plus the day of their birthday. Each part of the sentence has been carefully selected so that it will fit together into a humorous sentence no matter what. This particular table is a 'oneshot' – it will always give the same answer due to the nature of the question. However, choosing changeable answers such as 'the food you last ate' could make for a replayable alternative, or by converting it into a grammar containing every possible variation with the system selecting one at random.

Quick Start Guide: Create a table with three columns and as many rows as you require for your rules (use the same rules as in the example for speed). This can be on paper, or an online whiteboard (such as Google Jamboard), or in an offline program (such as Excel.) Fill in each row ensuring that each subsequent row will still create a coherent sentence when recombined. Pay particular attention to tense.

2.3.2 Grammars

As Anastasia Salter and Stuart Moulthrop point out in their introduction to text generation – Madlibs is a 'substitution grammar'. In other words, grammar systems usually work at a sentence level, substituting in new words or phrases to create new texts. Tracery is the coding language and grammar system used in Trice (See *Zoo of Unnamed Creatures* above). It's a very simple way of creating generated text via grammars that are easy to expand and simple to understand for non-coders.

Tracery: https://tracery.io/

Creator: Kate Compton

Overview: Text is written as simple sentences, with selected words having multiple possible outcomes. The system selects one outcome from its pre-authored lists and recombines them to output a newly generated sentence. This method can be repeated and expanded as needed, with sentences nested inside one other for greater complexity and variation.

Quick Start Guide: To get started with Tracery, use creator Kate Compton's Tutorial.

For those looking for something more complex, there are various online **corpora** available, such as the one used in Mainframe. (See Case Study 4 for more information).

2.3.3 Storylets

Storylets are a way of telling a non-linear story without (or in addition to) **branching**. They offer a modular method of storytelling where each storylet is gated according to reader-player actions and variables and the world state of the narrative. Twine recently added a specific Storylet function, although this method can be recreated in most narrative engines and tools, and in a lot of different 'shapes' as explained by Emily Short.

Harlowe Storylets Example: https://twinery.org/cookbook/storylets/harlowe/harlowe_storylets.html

Creators: Twine team

Overview: This very short demo shows how a storylet-based narrative written in Twee code looks to the end user – essentially indistinguishable from a standard Twine. The key differences here are on the writing side rather than the output. This particular example uses a cycling link to set a variable, which in turn dictates which storylet is shown.

Quick Start Guide: Dan Cox's guide describes all main functions of Twine storylets and their usage.

3 Experiments and Exercises

The following experiments will guide you through the tools and methods mentioned above in a little more detail with specific outcomes in mind. They only represent a tiny sample of ways you might use text generation creatively, but will hopefully inspire you to come up with your own uses and experiments. Reviewing the AI-Powered and Text Generation Case Studies may provide further ideas.

3.1 Text Generation as Inspiration

So far, most of the examples given have been ways of generating text or narratives with the end goal being a repeatable or personalised story. However, text generation can also be used as a method of overcoming writer's block or developing new ideas, as a *precursor to* writing something, rather than a *way of* writing something. The exercises below demonstrate some possible examples.

3.1.1 Exercise 1: Cut-up Technique for Idea Generation

Visit:
http://www.languageisavirus.com/cutupmachine.php - .YIabSX1Kj0p

Follow the steps and use a phrase or word-pairing from the output as a writing prompt.

3.1.2 Exercise 2: Developing a Basic Grammar

Begin by writing a simple descriptive sentence with at least a couple of adjectives and/or verbs. You could do this on paper, or on a Google Jam board.

E.g. The quick brown fox jumped over the lazy dog.

Now consider which words in the sentence could be changed. In this example, it's virtually all of them, but we'll start with a couple.

The **quick** brown fox **jumped** over the **lazy** dog.

You could use synonyms for the word (e.g. **fast, speedy, agile**) or antonyms (e.g. **lively, active, energetic**), or entirely different words altogether (e.g. **fell, sidled, danced**)

There are several potential uses for this exercise:

1) To spice up a sentence that you're concerned could be hackneyed or overused by finding alternative words within it.

2) To consider your word choices more deeply. Ask yourself, for example: **Why did you select those particular descriptors for this location or character? Does changing them change the meaning of the sentence? If so, which is your preferred version and why? If not, do they need to be there at all?**
3) To defamiliarize a familiar sentence and turn it into a writing prompt.
4) To form the basis for a computational grammar (see Exercise 3)

3.2. Text Generation as Expansion

One of the key uses of text generation within a game or interactive fiction is to quickly create a variety of content without having to individually write out (for example) multiple different dialogue options. It also has the added effect that readers/players (and even the creator) returning to the work may find phrases that are unexpected, unintended, or simply different from what was seen before, thereby adding an element of surprise and freshness to a repeatable experience.

3.2.1 Exercise 3: Developing a Digital Grammar

Use the sentence and alternative word choices you created in Exercise 2 as the basis for a small digital generator. You could use Tracery; Trice; the 'shuffle' functions in Bitsy or Ink; the 'either' or 'display' function in Twine Harlowe (or a combination of both; RiScript, or any other interactive narrative tool you might find.

All of these systems have tutorials or guides available, but for more ideas and tips, take a look at the Case Studies.

3.2.2 Exercise 4: Narrative Integration

"Mix authored and procedural […] [and] combine techniques – a single approach is unlikely to be flexible enough."

Nate Austin (2021). 'Procgen in Wildermyth: Storytelling'. BUas Games. YouTube. 22:19.

One of the most challenging (and rewarding) uses of text generation is to integrate it into a narrative. Nate Austin's talk on Procgen Storytelling (quoted above) is a good primer on how that might be done.

Try the following exercise to get started with your own partially generated story:

Take the small generator created in the exercise above.

Consider where you might add **branches** or **storylets**. Limit the number at first – an absolute maximum of three should be enough for now.

Now consider where you might be able to reuse some of your generated content in different places and contexts within the new branches/storylets.

Will you need to record any of the variables to keep them consistent? (e.g. character names)

Will the story have an end state, or will the reader-player be able to cycle indefinitely?

You may note that this requires a lot of planning and thought for even a small piece, but also that after planning carefully, it can be easily expanded simply by adding more synonyms/alternatives to your grammars or shuffles. This is what makes text generation so appealing for replayable narrative experiences.

For more possible examples and methods, see the Text Gen Case Studies.

4 Writing Process

While there are some processes that are useful for any project (see the Quickstart Guide for a basic outline). There are some methods that are particularly important when writing with text generators. You can see these in action in the Text Gen Case Studies, but they are outlined below for clarity. The first and most important step is to determine what your requirements are for this particular project, the types of generation you will be using and how these generators will integrate with (or create) your narrative. It's also a good idea to have at least an outline of the story you want to tell – key characters, locations etc and key events, regardless of what process you are using. The questions and tips below are by no means exhaustive and are instead intended as starting points to help you develop your own planning and writing processes for generative narrative.

4.1 Planning

How reusable will your content be?
One of the main reasons for using grammars is to create more variation with less work. Therefore you'll need to consider whether some of your grammars will ever be 'plugged into' others and if so, check things like **tense** and **sentence structure.**

Will any elements need to remain consistent?
This will dictate whether you will need variables working in conjunction with your grammars. If you just want to generate, for example, a village full of characters with different names, who each only appear once, then pure generation will be fine. However, if you'd like to generate a character and then follow them through the story, you will need to record that character's traits to variables in order to be able to recall them later.

How will the various systems used affect one another?
One of the issues with designing generative work is it can be hard to know when to stop. Just one more grammar, just one more variant! But when you're doing something with a little complexity, it's important to think about how each added detail contributes to the whole. If you have a weather system, will it affect the character's mood? If their mood reaches a high or a low, does that affect their dialogue options? You may need to start with a couple of core systems and add more once you have those implemented and working, or begin by listing everything you can think of that could be incorporated and edit down to those that are most relevant to the story you want to tell.

4.2 Writing

Comment everything! **Commenting** code is a good habit to get into when working on any interactive project but is especially important with generative systems which you might significantly add to. This isn't possible with some tools (e.g. Bitsy), but wherever possible, it's a good idea to clearly label the purpose of each generator and variable and how they interact.

4.3 Testing

Robust testing is difficult with generative works because unlike a more linear or simply branching narrative, there's no way to test every possible permutation of the text.

The planning and writing stages should help iron out some of the potential issues here, although be sure to **refer to your plans and comments frequently** to make sure everything is working as intended.

Test as often as you can, as early as you can, and consider using a versioning management tool such as Github to help ensure you always have a working version.

'[…] designers shouldn't worry as much about making their procedural content too sensible or 'seamless''.

Tanya Short, quoted in Kris Graft, (2018) 'Devs weigh in on the best ways to use (but not abuse) procedural generation', Game Developer, March 12 2018.

4.4 Key Takeaways

To summarise the points above, the most important elements to remember when writing with text generators:

- Have a clear purpose in mind. Whether this is in terms of using generators or creating them, you'll get the best out of using them if you know your core reason for doing so.
- Leave yourself clear instructions and notes. While a generator's purpose may seem clear at first, as the project expands and grows in complexity, it's easy to lose sight of how the various generative elements work and/or affect one another.
- Accept that you will sometimes generate strange incoherent ramblings – the unpredictability of generated text is what makes it fun.

"[…] discover strange things, unfathomable, repulsive, delightful; we will accept and understand them."

Arthur Rimbaud, 'Rimbaud's Systematic Derangement of the Senses', Language is A Virus.

5 What Next?

This toolkit has shown that text generation can be used as a prompt for non-interactive writing, as a creative form in its own right, or as a method of expanding and enhancing other forms of interactive text.

Some possible next steps include:

- Reading some of the AI-Powered Case Studies as they make use of 'fragments' (which are essentially grammars) in order to vary and tailor AI dialogue, and use AI agents to inspire writing.
- Taking a look at some of the 'inspirations' at the end of the bibliography, particularly those associated with text generation.
- Working through the Quickstart Guide process with text gen as your **Creativity Amplifier** to create your own generative piece.