Procedural Story Generation with Transformers

UNIVERSITY *of* LIMERICK

OLLSCOIL LUIMNIGH

Niall Dillane

Computer Science and Information Systems

University of Limerick

Submitted to the University of Limerick for the degree of

BSc Computer Systems 2020

1. Supervisor: James Patten
Computer Science and Information Systems
University *of* Limerick
Ireland

2. Second Reader: Dr. Name Surname
Computer Science and Information Systems
University *of* Limerick
Ireland

Showcase day: 18th April 2020

Signature:

Abstract

Procedural content generation (PCG) – the process of generating data algorithmically – is a technique that has applications across a variety of domains. In the research outlined in this report, focus is directed to the use of PCG as a means to generate novel-like stories. The challenge is twofold: research into procedural generation methods, as well as the structure and language of stories, addressing questions such as: "What are the elements of a good story?". Finally, methods to codify these elements must be investigated, in such a way that they can be used by a procedural generation technique, effectively combining the two disciplines.

Declaration

I herewith declare that I have produced this paper without the prohibited assistance of third parties and without making use of aids other than those specified; notions taken over directly or indirectly from other sources have been identified as such. This paper has not previously been presented in identical or similar form to any other Irish or foreign examination board.

The Final Year Project was conducted from 2019 to 2020 under the supervision of James Patten at University of Limerick.

Limerick, 2020

Acknowledgements

First and foremost I would like to thank James Patten; both for his inspiration of the topic researched, and the constant helpful advice and motivation throughout. This extends to all of the lecturers and teaching assistants I have interacted with over the years, each of whom contributed to where I am today.

The wide breadth of open source code, lectures and tutorials available online were also essential in allowing me to carry out this project, as it was all new ground compared to schoolwork that came before it.

Shoutout Christina Applegate

Contents

List of Tables

LIST OF TABLES

List of Figures

LIST OF FIGURES

1

Introduction

The overall goal of this report is to investigate what makes a good story, and then methods for algorithmically generating stories of this nature. The question of "Why?" may be asked. What makes stories important enough to warrant this kind of work?

Without delving too much into the later content of this report, stories are widely believed to be incredibly useful for fostering an understanding of the shared human experience and questions of existence (Eder 2010), for educational communication (Birch and Heckler 1996) and at their most incisive, contributing to social and political change (Fuertes 2012). This is all in addition to the entertainment value we all gain from stories, whether they be written, recorded, or shared via word of mouth. These stories develop whole ideologies and cultures; one need only look at religions for evidence, and we have evidence of written literature dating back to 2600 BCE (Grimbly 2013).

For these reasons, I believe this research to be incredibly worthwhile. If we can aid or expedite the writing process with the introduction of procedurally generated contributions, this should spur even more creativity in human writers. My inspiration for this is drawn from Google Deep Mind and their Alpha Go (Silver et al. 2017) research: an artificial intelligence (AI) program designed to tackle the ancient Chinese board game of Go. This is outside the scope of our research, but one takeaway from their research was that humans learned and became better for having played the AI program, in the same way they might improve while playing a superior human. We believe we can achieve similar

improvements in human writers with the help of an AI writer, as well as the AI's productions themselves.

1.1 Aims and Objectives

Two main questions are addressed by this research: what is a good story, and how they may be procedurally generated. In order to address this question, this work focuses on the following issues:

- Firstly, an examination of stories from a linguistic perspective. Older research or that which does not concern itself so much with technology. This includes things like story structures, character development, plot devices and so on. Creating a perfect definition for a "good story" may be implausible, but narrowing our definition would be good progress.
- Secondly, we must investigate algorithms for generating these stories; work that has been done to formulate these elements in a way that we could use in a program.
- Thirdly, we will delve into the technology side, researching advancements that have been made from early stages up through state-of-the-art. From here we will produce a prototype product which will generate stories and allow human users to interact and modify these stories as they go. In the same way that a human-computer combination has proved more effective than a human alone in chess (Michie 1972), we aim to create an interactive co-writing experience.

1.2 Methodology

In order to address these questions we will follow the methodology outlined in sequence above, before designing the system itself and implementing the product. Afterwards, we will reflect and perform various types of evaluation on the productions of the system, noting that it is not primarily intended to be run alone

but rather with human interaction. This will include known language generation metrics as well as human review.

1.3 Research Contribution

With this research, I hope to contribute a comprehensive history and discussion of what makes stories a worthwhile endeavour, how their quality and elements may be defined, and finally produce a prototype that allows us to see the potential of procedural story generation.

1.4 Report Outline

The remaining chapters of this report are as follows:

Chapter Two outlines and discusses the history and related research to this topic, from linguistic and technological perspectives.

Chapter Three relates to the design of my chosen system and the choices that were made with regard to models, architecture etc.

Chapter Four presents the implementation of the system, portions of interesting code, struggles that were faced and how I overcame them.

Chapter Five deals with the evaluation of productions from the system, both objective metrics and subjective human review.

Chapter Six draws conclusions and evaluates my satisfaction and areas for improvement. Lastly, it suggests possible future works and research.

2

Related Research

Much research has been undertaken on this topic, with a variety of approaches.

2.1 Stories

It is important for us to understand the most basic, linguistic concepts before we move on to algorithmic productions. What are the elements that make up a good story? What constitutes a good story? What even is a story?

According to Webster (Dictionary 2002), a *Story* is:

> An account of incidents or events, [either] regarding the facts pertinent to a situation in question, [or] a fictional narrative.

With a *narrative* being:

> A way of presenting or understanding a situation or series of events that reflects and promotes a particular point of view or set of values.

This is naturally a broad set of definitions, but it does give us some cues. From a story, we would expect a series of events related to each other in some way, either to describe a situation or promote a certain worldview or message. That is, there is an overall connection and cohesiveness to the piece. Literary critics have posited several interpretations or generalisations: that stories begin in equilibrium before being disrupted, and ultimately involve a journey back to equilibrium (Todorov and Weinstein 1969), but yet more argue that there can be

no "correct" definition determined (Sullivan 2002). The fields of Literary Theory and Narratology emerged in an attempt to dissect and formulate stories, which we will discuss more later.

I first set out to examine commonly used techniques used in various aspects of storytelling. These will perhaps be more relevant in evaluating the productions of my system, rather than guiding them too much, depending on the level of autonomy, and the rigidity of training and generation it has.

2.1.1 Structure

Structure describes the underlying framework of a story and, as the highest level of a story planning process, was first to be investigated.

The classic structure would be the three acts. This dates back to Aristotle in 400BC, describing a story as having three parts: a beginning, an end and a middle (Mack et al. 1980), and is still popular today. It is commonly depicted something like (Figure 2.1).

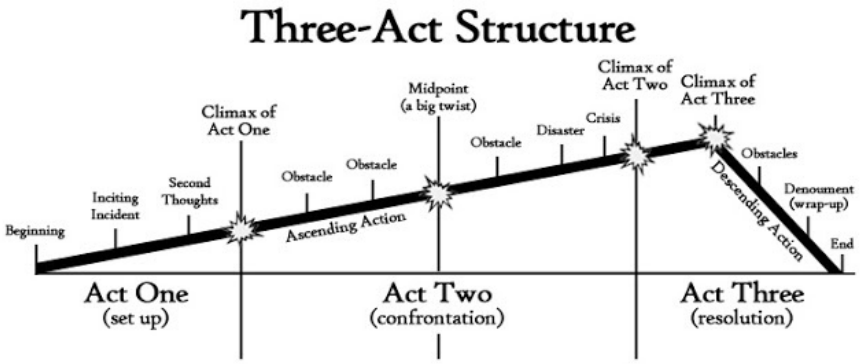


Figure 2.1: Three Act Structure - Breakdown of three act structure. [Source: (Morrill and Williamson 2013)]

Stories are broken town into distinct sections: setup and exposition, rising action and confrontation, climax and resolution (Trottier 1998). This is a bit more versatile, which is something to consider for the later steps of "filling in the gaps" with our algorithmic approach. A balance must be struck between:

providing some structure so that the generation has some level of cogency, but also allowing flexibility so that not all stories are the same.

Other popular methods include the Hero's Journey (Campbell 2008), which describes a cycle of sorts: a call to adventure, crossing from the known to the unknown, transformation etc. This is a more granular structure, most known for its application in Star Wars. See (Figure 2.2).

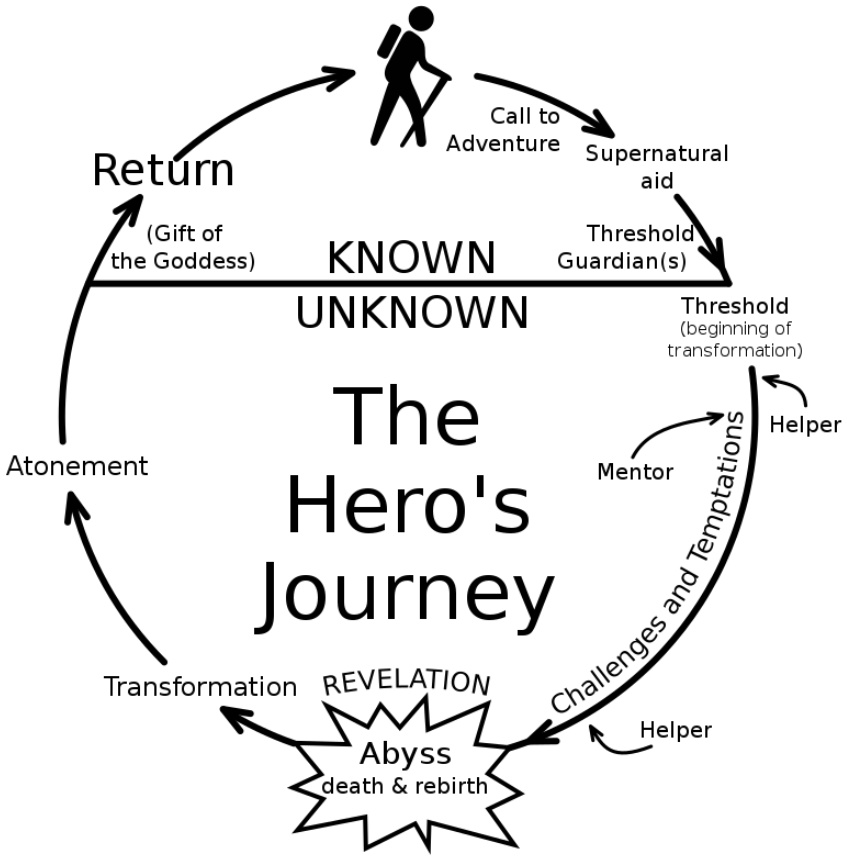


Figure 2.2: The Hero's Journey - A cycle of the hero's journey. [Source: (Vogler 1985)]

However, that is not to say these are the only structures that must be followed, nor that there must be that traditional arc. Some authors have examined spiral, fractal and explosive patterns in literature (Alison 2019), rejecting the historical, structural norms. It is tempting to declare adherence to a structure irrelevant, but patterns do remain, so this cannot be ignored entirely.

2.1.2 Plot

Delving deeper into the story, plot makes up those events which are significant, have consequences and make a difference to the story (Dibell 1999). If we examine our three act structure figure from earlier (Figure 2.1), we see ticks along the line of the story, larger incidents which have an impact.

Indeed, a scene or series of events may be memorable and iconic, but if they do not serve as major events that progress the overall narrative, then they do not constitute plot (Alcorn 2014). Following on from the three-act structure, plot points would be used to connect the acts to each other, for example: our protagonist is thrust into an unexpected situation, they face a setback and it seems all hope is lost, finally they overcome.

For the context of our system, plot points should serve as transitional pieces, advancing the story in some way so we don't simply remain at or revert back to the previous content. This must be handled with care, as too few plot points would be boring, but too many would be bewildering.

2.1.3 Setting

This refers to the time, location and milieu in which the story occurs (Lodge 2012) often referred to as the "world" or "universe", in modern works.

This serves as the backdrop of our story, and to feel authentic it must be rich with context and history. At their best, settings are so specific that they provide natural associations to the reader (Kuntz 1993), setting the mood and plot anticipations.

When generating content, there should be at minimum a consistency of setting, and ideally it should have enough detail to establish a mood that carries through the story.

2.2 Narratology

Having touched on literary theories, I was set on to the work of Vladimir Propp on Narratology (the study of narrative) and his early work in formulating elements of stories. Specifically, his seminal work with Morphology of the Folktale (Propp

1968), originally written in 1928. He, along with other Russian formalists, took a modern approach to narratology after Aristotle's ancient theorising.

They distinguished the syuzhet (plot) from the fabula (story). The idea was that the story is the raw material, familiar in many ways already, and it is *defamiliarised* (a term they coined) into the plot, a new organisation and the way the story is told. This goes back to our previous point on Plot (2.1.2), which pointed out that plot pertains to the information that pushes a story along. They are subtly different concepts.

2.2.1 Abstractions

Propp's work is very relevant to this research, since he has some of the earliest work on abstracting and formulating aspects of stories (specifically Russian folktales).

He first curated a table of possible events at various stages of a story, then associated these with symbols and combined them with functions into what look like mathematical formulas. An example looks something like: (Figure 2.3).

1. Analysis of a simple, single-move tale of class H-I, of the type: kidnapping of a person.

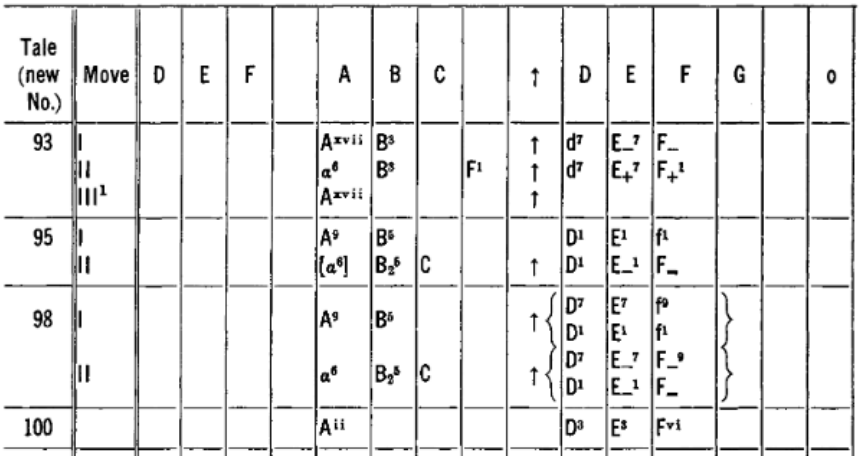
131. A tsar, three daughters (α). The daughters go walking (β^3), overstay in the garden (δ^1). A dragon kidnaps them (A^1). A call for aid (B^1). Quest of three heroes ($C\uparrow$). Three battles with the dragon (H^1-I^1), rescue of the maidens (K^4). Return ($\downarrow$), reward (w°).

$$\beta^3\delta^1A^1B^1C\uparrow H^1\text{-}I^1K^4\downarrow w^\circ$$

Figure 2.3: Morphology of the Folktale - Breaking down a story. [Source: (Propp 1968)]

There were hundreds of these, assembled in a tabular format like so: (Figure 2.4).

This laid groundwork for the development of grammars, while this is perhaps too rigid and systemic to be applicable, as argued by some (Dundes 1997).



Tale (new No.)	Move	D	E	F		A	B	C		↑	D	E	F	G		o
93	I II III[1]					A^{xvii} α^6 A^{xvii}	B^3 B^3		F^1	↑ ↑ ↑	d^7 d^7	$E\_^7$ E_+^7	$F\_$ F_+^1			
95	I II					A^9 $[\alpha^6]$	B^5 B_2^5	C		↑	D^1 D^1	E^1 $E\_^1$	f^1 $F\_$			
98	I II					A^9 α^6	B^5 B_2^5	 C		↑{ ↑{	D^7 D^1 D^7 D^1	E^7 E^1 $E\_^7$ $E\_^1$	f^9 f^1 $F\_^9$ $F\_$	}		
100						A^{ii}					D^3	E^3	F^{vi}			

Figure 2.4: Morphology of the Folktale - Table of stories. [Source: (Propp 1968)]

2.3 Formal Language Theory

(Formal) Grammars were devised as a more generic and granular approach to generating strings of text, by following certain rules, from a certain alphabet (Reghizzi et al. 2013). Compared to Propp's work on formulating *elements*, this was work being done down to the character level, getting closer to what we would need for algorithmic generation.

Emil Post was one of the early innovators in this area, creating the Post Canonical System in 1943, a string manipulation system for generating instances of a language, from an initial alphabet and rules (Post 1943).

2.3.1 Grammars

Noam Chomsky then proposed a set of generative grammars in 1956, classified in the *Chomsky Hierarchy* (Chomsky 1956), with different levels of strictness in their rules. The two efficient and popular types were the Context Free Grammar and Regular Grammar.

Chomsky grammars consist of a finite set of production rules (left-hand side $\rightarrow$ right-hand side), where each side consists of a finite sequence of the following symbols:

- a finite set of nonterminal symbols (indicating a production rule can be applied)
- a finite set of terminal symbols (indicating no production rule can be applied)
- a start symbol (a distinguished nonterminal symbol that is not found on any right hand side, and so cannot be produced anyway else)

For example:

$$S \rightarrow AB \tag{2.1}$$

$$S \rightarrow \lambda(emptystring) \tag{2.2}$$

$$A \rightarrow aS \tag{2.3}$$

$$B \rightarrow b \tag{2.4}$$

This is a Context Free Grammar (CFG) that could generate a string of letters "a" and "b".

Computing systems for grammar and story generation have been built (Compton et al. 2014,)however, formal grammars like this require significant human building and labelling. While these are interesting and worth exploring from a historical perspective, they are troubling from the perspectives of extensibility and originality.

2.4 Natural Language Generation

A subfield of linguistics and computer science, Natural Language Processing (NLP) deals with making easier the interaction between humans and computers, enabling machines to more easily understand natural, human language. It has risen to prominence since the 1990s in line with the rise of machine learning

as a programming technique (Johnson 2009). Before this, early language processing systems were based on handwritten rules like the grammars outlined above (Schank and Abelson 2013), which meant a comparative lack of knowledge, or features – weights on different choices based on the data. This machine learning approach allowed programs to learn rules by themselves, via analysing large amounts of input data.

Further developments in the 2010s brought the advancement of deep learning and neural networks, which could achieve better results than ever before (Goldberg 2016). We will examine these in more detail shortly, as these results made them quickly became the leading contender for our system.

It is worth noting that the NLP field is wide, and for our purposes we will primarily focus on the area of Natural Language Generation (NLG). Natural Language Understanding, which we are perhaps more familiar with in things like virtual assistants, aims to take in natural language, abstract it and produce some kind of representation of the idea being conveyed (Reiter and Dale 2000). Conversely, NLG attempts to take what is structured data, weights and biases based on input data, and produce natural language.

2.4.1 Neural Networks

An Artificial Neural Network (ANN) is a computing system that seeks to emulate the kind of processing and problem solving done by the human brain, designed around pattern recognition (Haykin 1994). The core principle is that they "learn" to perform said task by analysing examples and figuring out their own rules, rather than being explicitly told any rules up front.

An ANN is modelled after the human brain, containing nodes, or "neurons", which are connected to each other. They can receive input, process this with their internal state, and produce output (Winston 1992). This can be passed as a message to another neuron or ultimately, complete the task at hand. A basic example of the architecture looks something like the following: (Figure 2.5).

Each connection between neurons has an associated weight, which represents its relative importance, which is taken into account by the receiving neuron processing its inputs.

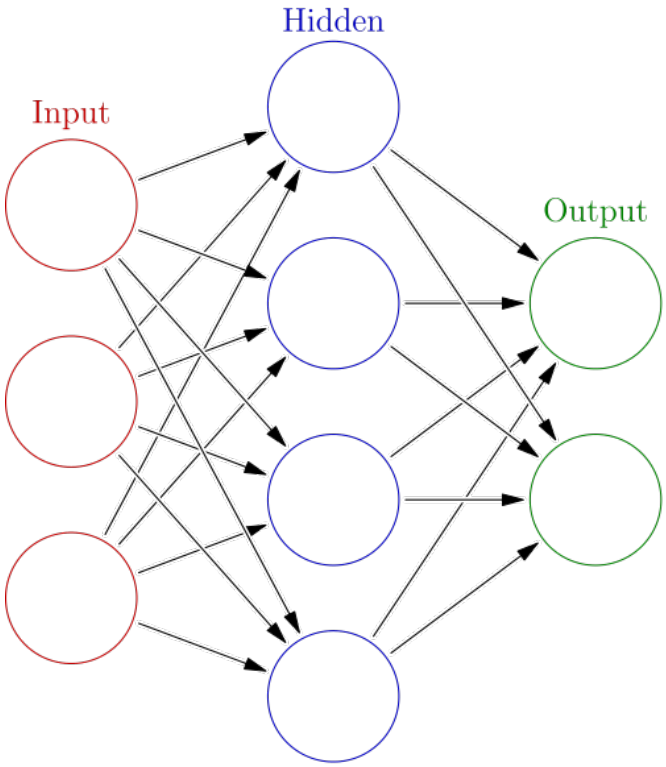


Figure 2.5: Artificial Neural Network Architecture - Neurons and connections. [Source: (Glosser.ca 2013)]

The classic example and popular use case is image identification (Le 2013). To teach a program to correctly separate images of cats and dogs, the basic approach would be to give it explicit rules for cats versus dogs – nose, ears, paws, etc. However, the ANN allows you to simply feed it examples (ideally many) of images of both, and then allow it to formulate those rules by itself. This saves time and work for the user, and tends to create a much more accurate model (LeCun et al. 1989), especially with a large amount of input, "training" data.

2.4.1.1 Recurrent Neural Networks & LSTMs

Examining some of the recent works on natural language generation and even specific story generation papers, Recurrent Neural Networks (RNNs) stood out as the popular technology (Peng et al. 2018), (Fan et al. 2018), (Sutskever et al. 2014).

These are like neural networks, but with loops that provide a kind of memory or persistence (Graves et al. 2008). These have feedback connections and can process sequences of data, instead of just single data points like a typical (feedforward) neural network. Particularly Long Short Term Memory (LSTM) networks, which are better at remembering long term dependencies. See (Figure 2.6).

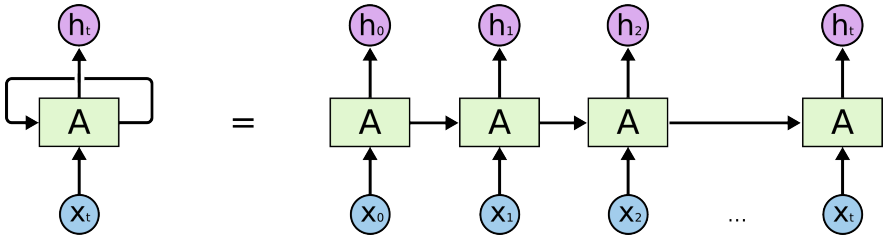


Figure 2.6: Recurrent Neuron - Loops for persistence [Source: (Olah 2015)]

This is incredibly important for NLG, because context matters. When identifying images, as with the previous example, this isn't such a big deal. Examples are distinct and don't depend on each other – one input, one output. However, with video or language, you must generate the next part of your output, piece by piece. In this case, remembering what was written previously is vital, not only to make coherent sentences but also to construct an overall narrative throughout

the story. Characters should develop, plot points should advance, relevant twists should occur, and so on.

2.4.2 Transformers

However, these RNNs and LSTMs appear not to be state of the art in the NLP world anymore. Things move quickly, and that has led us to the Transformer architecture.

The idea was pioneered by a team of Google researchers (Vaswani et al. 2017) and the early results are very promising. OpenAI's GPT-2 (Radford et al. 2019) is perhaps the most advanced language processing Artificial Intelligence (AI) system currently, so much so that they declined to release their full code, claiming they fear it could be used for ill means e.g. fake news.

We mentioned RNNs utilising loops, but in reality it's more like a series of connected Neurons, each performing its processing and then passing the new "state" to the next. This is what allows it to maintain memory – each stage of the computation is aware of everything that happened previously. You can see this in the following, with the RNN on top and Transformer on bottom: (Figure 2.7).

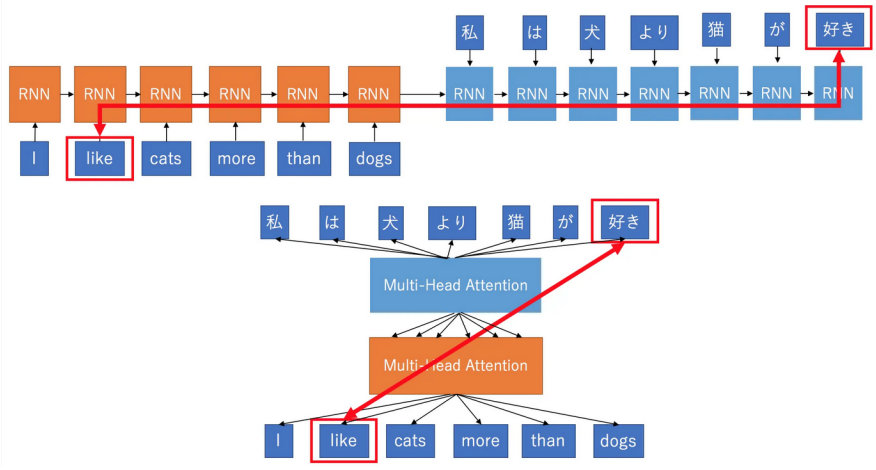


Figure 2.7: RNN vs. Transformer Architecture - Attention replacing loops [Source: (Kurita 2017)]

Note: here the example is translation, but for language generation the idea is the same. Instead of passing in a passage and translating it, we seek to find the next part of said passage. Note also that there are encoding (orange) and decoding (blue) stages, but they are practically similar.

The issues here are clear. The RNN is very difficult to parallelise since the computation must be done in sequence, passing the state along (Bengio et al. 1994). Also, the further you drift from the start (i.e. the longer the text becomes) the less your network will remember about those early stages. This is a problem for stories, where elements are often mentioned briefly at the start, only to come into prominence later – vital for continuity.

Transformers, by contrast, take the entire input passage in at once, using the concept of Attention (Vaswani et al. 2017) along with positional encoding (to ensure you don't start mixing up a sentence) to calculate the relative importance of each word. These results are then passed through a more traditional feedforward Neural Network (much faster than an RNN) and potentially another multi-head attention stage (applications differ) to produce your following piece of the passage.

For the purposes of this research, we won't be delving into the inner workings of the Attention mechanism, but essentially what it does is calculate the relevance of each word in the input. For example, if you have the input "I grew up in France... I speak fluent" and wish to predict the next word, the word "France" will be given a much higher score, indicating to the system what it should predict based on.

This helps hugely with the long term dependency issue, since at each stage the entire input is considered, and thus there is a much lower risk of earlier information being "forgotten", something that even the finest LSTMs struggled with.

This could lead to performance issues, but thanks to the Attention mechanism and feedforward NN being used (allowing you to process the entire input at once), Transformers are in fact faster.

The best models also seem to utilise unsupervised learning (to be discussed later), which has the obvious benefit of being able to feed it much more data, as

seen with GPT-2 and its 1.5 billion parameters (Radford et al. 2019). However, you do need to feed it a massive amount of data for it to be in any way effective.

A number of models have been developed with this architecture.

2.4.2.1 BERT

Google, being the organisation behind this original Attention mechanism research (Vaswani et al. 2017), was naturally one of the first to develop a Transformer architecture based on it.

BERT (Bidirectional Encoder Representations from Transformers) was developed in 2018 and promptly achieved excellent results on natural language understanding tasks (Devlin et al. 2018). The findings were significant enough that in October 2019, Google began applying BERT to improve its search algorithms (Nayak 2019), the foundation of their business.

However, while exciting and emblematic of the potential of Transformers, BERT is not necessarily suited to our task. As mentioned, it was mainly trained and used for the purposes of natural language understanding (NLU), rather than generation (NLG). As such, I continued my search for a more appropriate model.

2.4.2.2 GPT & GPT-2

GPT (Generative Pretrained Transformer) was also developed in 2018, this time by OpenAI, a decoder-only Transformer architecture with largely unsupervised training. A massive corpus of text was gathered from the internet and fed to the model, which quickly produced impressive results (Radford et al. 2018).

Their research focused on initially training the model (unsupervised) on a very diverse, wide range of text, then fine-tuning to each specific task on top of that, now supervised.

It follows the simple principle of predicting *the next word*, much like our phones will offer next-word suggestions based on what we have typed before, but on a much more advanced level and (ideally) not descending into gibberish. This was incredibly promising and looked ideal for my use case.

GPT-2 was next, taking things a step further with entirely unsupervised learning, and achieving even more state-of-the-art results than before (Radford et al.

2019). To achieve this, a huge amount of parameters was necessary. For perspective, the smallest GPT model has parameters roughly equivalent in number to the largest BERT model. The largest GPT-2 has over an order of magnitude more than that. 1.5 billion parameters at its largest.

This is clearly the ultimate NLG model at this point, and seems like the obvious choice for our system.

3

Design

3.1 Language Model

We identified a range of potential models for this project, ranging from earlier grammars, to more recent Recurrent Neural Networks, and finally Transformers. For demonstration purposes, we decided it would be best to settle on one model.

3.1.1 GPT-2

As discussed, GPT-2 is currently the state-of-the-art model when it comes to Natural Language Generation. I identified several resources available which made it easier to utilise and fine-tune this model [1] [2], as well as the original open-source code itself [3].

This made the task somewhat less daunting, but my original intention at the time was to modify the underlying architecture in some way. Training a model, I reasoned, was too basic, and I needed something more. Since the code is open source, I set about reading through it, along with many explainer articles [4] and videos [5].

[1] Talk to Transformer by Adam King

[2] GPT-2 Simple by Max Woolf

[3] GPT-2 by OpenAI

[4] The Illustrated GPT-2 and other articles by Jay Alammar were incredibly useful

[5] Attention Is All You Need, a talk at the AI Socratic Circles, was helpful in understanding the Attention mechanism.

However, the realisation of the enormity of this challenge soon hit me. These were concepts and models developed by teams of post-doctorate level researchers, working full-time on these problems, with the backing of huge corporations. For myself, just getting started in the field of machine learning and NLP, it would be an Augean task.

We took a step back, and considered where I might add value to the system. We discussed the fact that these models, despite being significant advancements and impressive research, were still nowhere near consistently challenging a human writer, and that curation of the AI productions was still required.

This seemed like an area for exploration, to create an environment for human writers to work together with a language generation model, editing and inserting their own text as they go. This was settled.

3.1.2 Implementation

The implementation of GPT-2 I settled on was Huggingface's repository of Transformers. This contains not only GPT-2, but a collection of all state-of-the-art architectures, which leaves room for extensibility and including other models in the future.

The repository allows for a variety of approaches for optimal accessibility: from simply running generation on a pre-trained model, to fine-tuning your own model, to custom generation or even altering the lower level code. A variety of scripts were included, making it quick and straightforward to get up and running.

This seemed ideal for our purposes.

3.1.3 Training

One concern that had prevailed since I delved into Neural Networks was the hardware that is required to train these models. Transformers utilise Convolutional Neural Networks, which are faster than the Recurrent variant but still require significant computing power. Working on my own machine which lacked even a discrete GPU of any kind, or a lab machine with not much better hardware and limited availability, left me searching for alternatives.

Fortunately, those alternatives do exist. There are numerous cloud services available for training machine learning models on third-party hardware: Kaggle, Lambda and Weights & Biases, just to name a few. There were drawbacks, namely cost, but these were promising.

I eventually settled on Google Colab, which provides a Python Notebook interface and allows you to run code on advanced GPUs. There are limits on time and use cases, but I was comfortable I could fall within or work around these restrictions, especially considering that the product was free to use.

Of course, if I wanted to train a model then I would need data.

3.1.4 Data

Another issue that came up in the early stages was scope. Often an issue with projects of this nature, we wanted to make sure that I had a reasonable prototype which demonstrated progress. To generalise from the beginning would make the model's ability to optimise less clear, so we decided to focus on one type of story initially and perhaps generalise later, or at least leave that option open to ourselves.

I was careful when choosing a genre, since it was important to have a large and high quality data source to support it. Being a Reddit user for some time, I recognised the wealth of data available on the website, not just as a whole but categorised neatly into different sub-communities dedicated to certain topics.

The Writing Prompts subreddit was the leading contender. A forum devoted to short story writing, it seemed like the perfect match. However, it was not specific to any genre of story and categorisation of the stories was minimal, meaning it was more of a general data source. Good, but not quite what we needed.

Ultimately, I settled on No Sleep, a community for short horror stories which, as the title goes, denied their users sleep. This was much more suitable, with focused data and a significant amount of it too. As of 26 March 2020, the subreddit has been going for 10 years with 13.9 million subscribers. This was perfect.

3.2 User Experience

Simple webapp, user interaction, generate and change parameters, add/edit text as you go (Sketches of UI and structure)

3.2.1 Back End

Python/Flask API to call scripts based on huggingface implementation, take in parameters and text, return json with text object of generated string to replace existing one

3.2.2 Front End

ReactJS framework, library for building UI. Make calls to API and display results.

4

Implementation

4.1 Web Scraping

Gathered training data from reddit, few different approaches, different sized datasets produced.

4.1.1 Reddit API

Limited to 1000 posts

4.1.2 PushShift API

Third party data source of reddit posts, able to circumvent the limit.

4.1.3 Datasets

Small for testing, large unwieldy, medium used. Found that perplexity score was worse with medium compared to small.

4.2 Training the Model

Worried about hardware, online resources to the rescue! Sample scripts for GPT-2 provided by huggingface

4.2.1 HuggingFace

Open source abstraction of GPT-2, convenient scripts that can be modified and used. Checkpoints stored at various points.

4.2.1.1 Scripts

script code and explanation

4.2.2 Google Colab

Online jupyter notebook environment, can use a hosted runtime to take advantage of GPU/TPU, free for 12 hours at a time.

4.2.2.1 Workflow

Results stored in runtime and can be exported to Google Drive and downloaded (checkpoints useful here).

4.3 Python Flask API

API to encapsulate scripts, easily callable and customisable.

4.4 React JS

Web framework, modern JavaScript library for building UI.

4.4.1 Hooks

Functional approach, no classes, using state which is passed around. Challenging new way of thinking but extensible and clean.

5

Evaluation

5.1 Language Model Metrics

Objective measures are a good baseline against other models. Unsure of efficacy.

5.1.1 Perplexity

Lower the better, got worse with larger dataset. Not necessarily a good measure.

5.2 Human Review

More effective but obvious downsides in terms of speed. Part of the interactive experience.

5.2.1 Self Review

My judgement.

5.2.2 Anonymous Reviews

Posted various stories back to reddit to gauge response.

6

Conclusions and Future Directions

6.1 Summary

6.2 Conclusions

6.3 Contributions

6.4 Future Work

6. CONCLUSIONS AND FUTURE DIRECTIONS

Appendix A

Insert a figure

Appendix

Appendix B

Code for estimating attitude

AHRS_TRIAD.C

```
/*
 *
 *  Copyright (c) 2013, Giuseppe Torre
 *  All rights reserved.
 *
 *  Redistribution and use in source and binary forms, with or without
 *  modification, are permitted provided that the following conditions are met:
 *      * Redistributions of source code must retain the above copyright
 *        notice, this list of conditions and the following disclaimer.
 *      * Redistributions in binary form must reproduce the above copyright
 *        notice, this list of conditions and the following disclaimer in the
 *        documentation and/or other materials provided with the distribution.
 *      * Neither the name of the BREAKFAST LLC nor the
 *        names of its contributors may be used to endorse or promote products
 *        derived from this software without specific prior written permission.
 *
 *  THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS_IS" AND
 *  ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
 *  WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
 *  DISCLAIMED. IN NO EVENT SHALL BREAKFAST LLC BE LIABLE FOR ANY
 *  DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
 *  (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
 *  LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND
 *  ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
 *  (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
 *  SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
 *
 */



#include "ext.h"
#include "ext_mess.h"
#include <string.h>
#include <stdio.h>
#include <math.h>
#define EPSILON 1e-6

//*****************CALIBRATION STUFF************************//
/* INTRODUCE THE OFFSET VALUES IN THE FOLLOWING VARIABLES IN ADC values */
#define ACCELX_OFFSET 2043 //initial offset in accelerometer X
#define ACCELY_OFFSET 2053 // initial offset in accelerometer Y
#define ACCELZ_OFFSET 2264//initial offset in accelerometer Z

#define MagX_OFFSET 1917 //initial offset in Magnetometer X
```

Appendix

```
#define MagY_OFFSET 1813 // initial offset in Magnetometer Y
#define MagZ_OFFSET 1667//initial offset in Magnetometer Z
/* WE SHOULD CALIBRATE THE FOLLOWING VALUES FOR EACH PARTICULAR IMU axis */
#define ACCELX_Resolution 536
#define ACCELY_Resolution 661
#define ACCELZ_Resolution 559
#define MagX_Resolution 186
#define MagY_Resolution 189
#define MagZ_Resolution 170
//****************** END CALIBRATION STUFF  ************************//


void *this_class; // Required. Global pointing to this class

typedef struct _triad // Data structure for this object
{
        t_object m_ob;   // Must always be the first field; used by Max

        Atom m_args[9];  // we want our inlet to be receiving a list of 10 elements

        long m_value;    //inlet

        void *m_R1;              //these are all the outlets for the 3 X 3 Matrix
        void *m_R2;              // R1 --> firts top left cell ..R2 middle cell of the first
             row ...and so on
        void *m_R3;                                     //
        void *m_R4;                                     // End Outlets
             ---------------------------------------
} t_triad;

                        void *triad_new(long value);


                        void triad_assist(t_triad *triad, void *b, long msg, long arg, char *
                             s);
                        void triad_free(t_triad *triad);



                        void triad_list(t_triad *x, Symbol *s, short argc, t_atom *argv);



                        void MatrixByMatrix(double *Result,double *MatrixLeft,double *
                             MatrixRight);

                        void Matrix2Quat(double *Quat,double *Matrix);
                        void Quat2Matrix(double *Matrix, double *Quat);
                        void inverseQuat(double *InvQuat, double *RegQuat);
                        void NormQuat(double *YesQuat, double *NotQuat);
                        void Slerp(double *NewQuat, double *OldQuat, double *CurrentQuat);
                        void NormVect(double *YesVect, double *NotVect);
                        double orientationMatrix[9];
                        double Result[9];
                        int     i;
                        double InvorientationMatrix[9];

                        double temp[6];
                        double ref[6];
                        double vectAx[3];
                        double vectAy[3];
                        double vectAz[3];
                        double vectBx[3];
                        double vectBy[3];
                        double vectBz[3];
                        double MagnCrosProd_A;
                        double MagnCrosProd_B;
                        double accnorm, magnorm, earthnorm, VectAynorm, VectAznorm,
                             VectBynorm, VectBznorm;
```

```
                        double m[9], n[9];
                        double quat_e[4];
                        double invquat_e[4];
                        double mult;
                        double quat_new[4];
                        double quat_old[4];
                        double ecs,accex_ADCnumber,y,accey_ADCnumber,z,accez_ADCnumber,mx,
                            magnx_ADCnumber,my,magny_ADCnumber,mz,magnz_ADCnumber;
// SLERP Variables
                        double trace, Suca;
                        double tol[4], omega, sinom, cosom, scale0, scale1, tez,
                            orientationMatrixA[9];

int main(void)
{
        // set up our class: create a class definition
        setup((t_messlist**) &this_class, (method)triad_new, (method)triad_free, (short)
            sizeof(t_triad), 0L,A_GIMME, 0);
    addmess((method)triad_list, "list", A_GIMME, 0);
        addmess((method)triad_assist, "assist", A_CANT, 0);
        finder_addclass("Maths","triad");
        post("....I'm_TRIAD_Object!....from_AHRS_Library...",0);
        return 0;
}

/* ------------triad_new ---------------*/

void *triad_new(long value)
{
        t_triad *triad;
        triad = (t_triad *)newobject(this_class);       // create the new instance and return
            a pointer to it
        triad->m_R4 = floatout(triad);
        triad->m_R3 = floatout(triad);
        triad->m_R2 = floatout(triad);
        triad->m_R1 = floatout(triad);

return(triad);
}

etc.

etc.
```

Appendix

References

Alcorn, S. (2014), 'Know the difference between plot and story', *Tejix. Archived from the original on* pp. 08–23. 8

Alison, J. (2019), *Meander, Spiral, Explode: Design and Pattern in Narrative*, Catapult. 7

Bengio, Y., Simard, P. and Frasconi, P. (1994), 'Learning long-term dependencies with gradient descent is difficult', *IEEE transactions on neural networks* **5**(2), 157–166. 16

Birch, C. L. and Heckler, M. A. (1996), *Who says?: Essays on pivotal issues in contemporary storytelling*, august house. 1

Campbell, J. (2008), *The hero with a thousand faces*, Vol. 17, New World Library. 7

Chomsky, N. (1956), 'Three models for the description of language', *IRE Transactions on information theory* **2**(3), 113–124. 10

Compton, K., Filstrup, B. et al. (2014), Tracery: Approachable story grammar authoring for casual users, *in* 'Seventh Intelligent Narrative Technologies Workshop'. 11

Devlin, J., Chang, M.-W., Lee, K. and Toutanova, K. (2018), 'Bert: Pre-training of deep bidirectional transformers for language understanding', *arXiv preprint arXiv:1810.04805* . 17

Dibell, A. (1999), *Elements of Fiction Writing-Plot*, Penguin. 8

Dictionary, M.-W. (2002), 'Merriam-webster', *On-line at http://www. mw. com/home. htm* . 5

REFERENCES

Dundes, A. (1997), 'Binary opposition in myth: The propp/levi-strauss debate in retrospect', *Western Folklore* pp. 39–50. 9

Eder, D. (2010), *Life lessons through storytelling: Children's exploration of ethics*, Indiana University Press. 1

Fan, A., Lewis, M. and Dauphin, Y. (2018), 'Hierarchical neural story generation', *arXiv preprint arXiv:1805.04833* . 14

Fuertes, A. (2012), 'Storytelling and its transformative impact in the philippines', *Conflict Resolution Quarterly* **29**(3), 333–348. 1

Glosser.ca (2013), 'Neural network diagram', *in https://commons.wikimedia.org* [online], available: https://commons.wikimedia.org/wiki/File:Colored_neural_network.svg [accessed: 26 Mar 2020] . 13

Goldberg, Y. (2016), 'A primer on neural network models for natural language processing', *Journal of Artificial Intelligence Research* **57**, 345–420. 12

Graves, A., Liwicki, M., Fernández, S., Bertolami, R., Bunke, H. and Schmidhuber, J. (2008), 'A novel connectionist system for unconstrained handwriting recognition', *IEEE transactions on pattern analysis and machine intelligence* **31**(5), 855–868. 14

Grimbly, S. (2013), *Encyclopedia of the ancient world*, Routledge. 1

Haykin, S. (1994), *Neural networks: a comprehensive foundation*, Prentice Hall PTR. 12

Johnson, M. (2009), How the statistical revolution changes (computational) linguistics, *in* 'Proceedings of the EACL 2009 Workshop on the Interaction between Linguistics and Computational Linguistics: Virtuous, Vicious or Vacuous?', pp. 3–11. 12

Kuntz, M. (1993), *Narrative setting and dramatic poetry*, Vol. 124, Brill. 8

Kurita, K. (2017), 'Paper dissected: "attention is all you need" explained'. **URL:** *https://mlexplained.com/2017/12/29/attention-is-all-you-need-explained/* 15

Le, Q. V. (2013), Building high-level features using large scale unsupervised learning, *in* '2013 IEEE international conference on acoustics, speech and signal processing', IEEE, pp. 8595–8598. 14

LeCun, Y., Boser, B., Denker, J. S., Henderson, D., Howard, R. E., Hubbard, W. and Jackel, L. D. (1989), 'Backpropagation applied to handwritten zip code recognition', *Neural computation* **1**(4), 541–551. 14

Lodge, D. (2012), *The art of fiction*, Random House. 8

Mack, M., Knox, B. M. and McGalliard, J. C. (1980), *The Norton anthology of world masterpieces*, Norton. 6

Michie, D. (1972), 'Programmer's gambit', *New Scientist* **55**(809), 32–329. 2

Morrill, S. and Williamson, J. (2013), *Go Teen Writers: Edit Your Novel*, Luminous. 6

Nayak, P. (2019), 'Understanding searches better than ever before'.
URL: *https://www.blog.google/products/search/search-language-understanding-bert/* 17

Olah, C. (2015), 'Understanding lstm networks'.
URL: *https://colah.github.io/posts/2015-08-Understanding-LSTMs/* 14

Peng, N., Ghazvininejad, M., May, J. and Knight, K. (2018), Towards controllable story generation, *in* 'Proceedings of the First Workshop on Storytelling', pp. 43–49. 14

Post, E. L. (1943), 'Formal reductions of the general combinatorial decision problem', *American journal of mathematics* **65**(2), 197–215. 10

Propp, V. I. (1968), *Morphology of the Folktale*, Vol. 9, University of Texas Press. 8, 9, 10

Radford, A., Narasimhan, K., Salimans, T. and Sutskever, I. (2018), 'Improving language understanding with unsupervised learning', *Technical report, OpenAI* . 17

Radford, A., Wu, J., Child, R., Luan, D., Amodei, D. and Sutskever, I. (2019), 'Language models are unsupervised multitask learners', *OpenAI Blog* **1**(8), 9. 15, 17

Reghizzi, S. C., Breveglieri, L. and Morzenti, A. (2013), *Formal languages and compilation*, Springer. 10

Reiter, E. and Dale, R. (2000), 'Building natural generation systems', *Studies in Natural Language Processing. Cambridge University Press* . 12

REFERENCES

Schank, R. C. and Abelson, R. P. (2013), *Scripts, plans, goals, and understanding: An inquiry into human knowledge structures*, Psychology Press. 12

Silver, D., Schrittwieser, J., Simonyan, K., Antonoglou, I., Huang, A., Guez, A., Hubert, T., Baker, L., Lai, M., Bolton, A. et al. (2017), 'Mastering the game of go without human knowledge', *Nature* **550**(7676), 354–359. 1

Sullivan, P. (2002), '" reception moments," modern literary theory, and the teaching of literature', *Journal of Adolescent & Adult Literacy* **45**(7), 568–577. 6

Sutskever, I., Vinyals, O. and Le, Q. V. (2014), Sequence to sequence learning with neural networks, *in* 'Advances in neural information processing systems', pp. 3104–3112. 14

Todorov, T. and Weinstein, A. (1969), Structural analysis of narrative, *in* 'Novel: a forum on fiction', Vol. 3, JSTOR, pp. 70–76. 5

Trottier, D. (1998), 'The screenwriter's bible: A complete guide to writing', *Formatting, and Selling Your Script* . 6

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł. and Polosukhin, I. (2017), Attention is all you need, *in* 'Advances in neural information processing systems', pp. 5998–6008. 15, 16, 17

Vogler, C. (1985), 'A practical guide to joseph campbell's the hero with a thousand faces', *Hero's Journey* . 7

Winston, P. (1992), *Artificial Intelligence*, A-W Series in Computerscience, Addison-Wesley Publishing Company.
URL: *https://books.google.ie/books?id=b4owngEACAAJ* 12