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 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.

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.

1. INTRODUCTION

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.

2.1.1 Structure

Structure was the first element to be researched, as the highest level of a story planning process. 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.

Another method is the three (or five) act structure (Trottier 1998), wherein the story is broken down into distinct sections, typically: setup, confrontation, resolution. This is a bit more versatile, which was something to consider for the later steps of “filling in the gaps” with PCG. There was a balance to be struck: providing some structure so that the generation has some level of cogency, but also allowing flexibility so that not all stories are the same. Following this, a tree-based approach became appealing, at the higher levels resembling more of a novel plan with plot points, and is eventually fleshed out below into sentences and phrases. This is a familiar concept in Computer Science, and it seems like the best way to maintain a coherent story. Some optional user interaction would be ideal – e.g. “Tell me a story about a dog with superpowers” – a kind of “seed”. These could also be generated autonomously.

2.1.2 Elements

You can learn more about Latex by clicking here.

2.1.3 Evaluation

But how?!

2.2 Narratology

Propp!

2.2.1 Abstractions

Marriage, villain, etc.

2.3 Technology

NLP!

2.3.1 NLP

Definitions, brief history

2.3.2 Grammars

CFG, Context Free, old and unworkable

2.3.3 Neural Networks

So hot right now.

2.3.3.1 Recurrent Neural Networks & LSTMs

Better, and best for a while, but still not great.

2.3.4 Transformers

vaswani et al, Attention is all you need, faster, longer range dependencies

2.3.4.1 BERT

Google

2.3.4.2 GPT

First attempt

2.3.4.3 GPT-2

New and improved!

3

Design

3.1 Language Model

Which one do we choose? But there are so many! (Figure 3.1).

Figure 3.1: Title of Figure - Description. [Source: (**?**)]

3.1.1 GPT-2

Comparison with the others, why it's best

3.1.2 Implementation

HuggingFace, extensibility, open source

3.1.3 Training

Hardware?! Google colab!

3.1.4 Data

/r/nosleep

3.1.5 Generation

Scripts, not too intensive

3.2 User Interface

Webapp! (Sketches)

3.2.1 Back End

Python/Flask API

3.2.2 Front End

React JS

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

Appendix A

Insert a figure

Figure 1: Title of Figure - Description. [Source: (**?**)]

Appendix

Appendix B

Title

Type here you text as usual . . .

Appendix

Appendix C

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.
```

References

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. 6

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

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

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

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

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.

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

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