Interim Report

Niall Dillane

13132911

LM051 Computer Systems

Procedural Story Generation with Transformers

Supervisor: James Patten

Procedural Story Generation with Transformers

13132911 – Niall Dillane[1], James Patten[2]

[1] BSc. Computer Systems, CSIS, University of Limerick, Ireland
[2] Lecturer, CSIS, University of Limerick, Ireland

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.

Keywords: procedural generation, algorithms, AI, language, neural networks, transformers

1. Introduction

In order to be able to automatically generate stories we first need to understand the structure of a story. Indeed, not only the structure that would make one grammatically correct, but also those devices, structures and styles that help make a good story. This is probably the largest challenge in the project, a formula which is still unclear after millenia of storytelling.

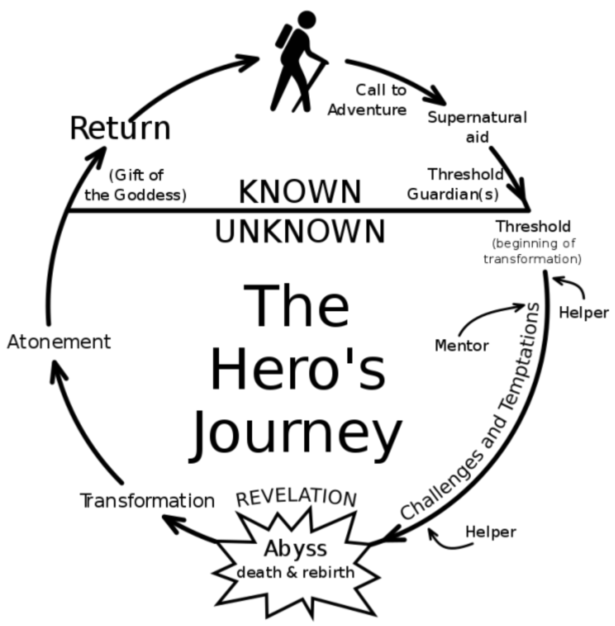
Furthermore, there have been significant advancements over recent years in the Natural Language Processing (NLP) field, with new architectures and technologies appearing and outpacing each other. Choosing a direction which is sufficiently developed but still relatively state of the art will be difficult, especially since few of these have specifically tackled the topic of novel-like stories, instead focusing more on news article or script style content.

However, it is this rapid progress and uncertain nature that make the topic so fascinating. There has been great progress in images, video, even games, but text generation has lagged behind somewhat. This signals that language is more nuanced, difficult to replicate, and still in its infancy.

2. Good Stories

2.1 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.



commons.wikimedia.org

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.

2.2 Elements

Work has been done historically in "formulating" stories. Originally written in 1928, Morphology of the Folktale (Propp 2010) delved into identifying the elements and plot points that defined a typical, old Russian folktale, down to the level of "hero marries his lover" and so on. This was very interesting as a seminal work in the field, but we felt that this kind of approach of having a formula (even if there are variations) is too rigid and not expandable enough to generate worthwhile, original content across a variety of genres.

Perhaps this warrants revision later, but at this stage it was felt that specifying individual story elements was unworkable at scale.

2.3 Drafting

Finally, one of the most important aspects of every story planning process had to be considered: drafting. Research in the area of Natural Language Generation (NLG) showed little mention of this concept – drafting and revising stories, so this is something worth investigation. The inner workings of this would be very dependent on the technology, but from a language perspective it stands to reason that no great story is written in the first draft. Why should a program be expected to?

2.4 Evaluation

The elephant in the room, of course, is that ultimately, nobody can define a "good story". Try as we may to identify common elements in popular or critically acclaimed works, it will be nigh on impossible to objectively distinguish good results from bad. In fact, the ability to evaluate a good story and produce one are inherently intertwined – a perfect evaluator could simply be reverse engineered to produce the stories itself.

With this in mind, our goal can not be to produce objectively good stories. Instead, we will focus on following the patterns of popular works. This isn't perfect, but assuming the intent of media is primarily to entertain, it seems like a reasonable barometer. Human evaluation will be required for this, so the stories will have to be kept relatively short, a few thousand words maximum.

3. Technology

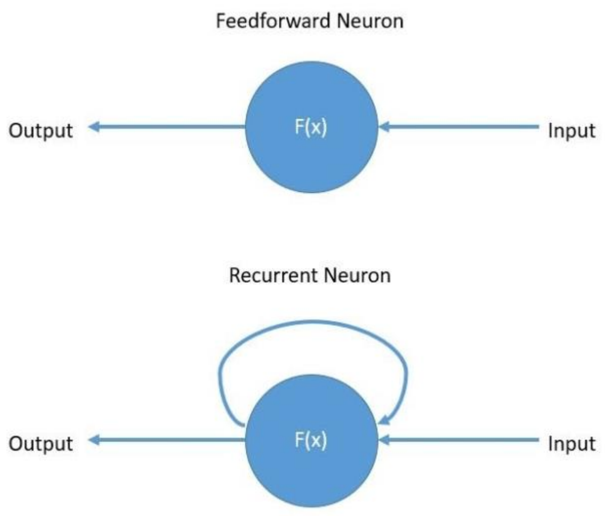
3.1 Neural Networks

With the previous discussion in mind, we set about examining the abundant previous research in the NLP field. In the early stages, the common denominator was the use of Neural Networks (specifically Recurrent Neural Networks).

A Neural Network (NN) 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 are taught to "learn" to perform said task by analysing examples, rather than being explicitly told any rules up front.

The classic example and popular use case is image identification. 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 NN 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 creates a much more accurate model.

As mentioned, Recurrent Neural Networks appeared to be the most popular recent technology in this field. 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 LSTM (Long Short Term Memory) networks, which are better at remembering long term dependencies.



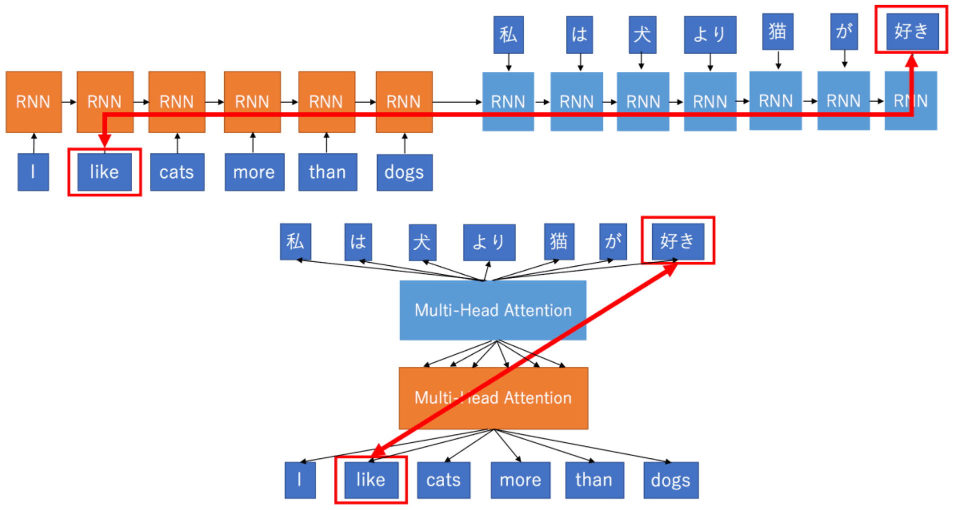
wiki.tum.de

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 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.

3.2 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.



mlexplained.com
(top: RNN, bottom: Transformer)

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 above diagram.

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. Also there are encoding (orange) and decoding (blue) stages, but they are practically similar.

The issues here are clear. These are very difficult to parallelise since the computation must be done in sequence, passing the state along. 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. However, you do *need* to feed it a massive amount of data for it to be in any way effective.

3.3 Language and Environment

Either way, Python will certainly be the core language used, as it is prominent in the machine learning and specifically NLP field. PyTorch (https://pytorch.org/) and TensorFlow (https://www.tensorflow.org/) both implement LSTMs and are widely used in Transformers. PyTorch initially seemed the better option for Recurrent Neural Networks, TensorFlow for Transformers, but with the latest version of TensorFlow 2.0

(boasting Keras integration, as well as the longer history and better support network) and my decision to proceed with Transformers, this made TensorFlow the obvious choice.

4. Training Data

Training data is vital to this project, since Transformers are still ultimately NN driven and these are highly dependent on the examples provided to them.

Creating a grammatically-correct series of words is not a major challenge, nor is coming up with something so nonsensical that it's entertaining to laugh at. Recreating something similar to popular stories – something that flows logically and engages a reader (however difficult that may be to define) – is much more difficult. This is the difference between good and bad training.

There is a variety of approaches.

4.1 Supervised Learning

This involves labelling input data with its expected output (Mohri et al 2018), and potentially labelling at an even more granular level, down to elements like characters, plot twists and settings. This has the benefit of being much more structured, and gives the model a head start on its learning, however it is incredibly time consuming.

Especially in the case of stories, the labels would likely be many times longer than the input itself, and this isn't workable. The model also needs to be trained with a significant amount of examples, and this is infeasible with such an extensive screening process.

4.2 Unsupervised Learning

This approach involves allowing the model to self-organise and calculate probabilities autonomously, using principal component and cluster analyses (Hinton et al 1999), with unlabeled data.

This has the obvious benefit of saving time and manual labour in the training stage, which allows you to feed much more data to the model, in theory balancing out the drawback of lacking additional information or cues to the system.

OpenAI's GPT-2 is built on this philosophy, and this is something we will seek to emulate.

4.3 Overfitting

One danger with Neural Networks, and statistical models in general, is the potential for overfitting. Essentially, this describes the phenomenon of fitting your model too closely to the data, such that outliers or noise are considered noteworthy and impact your predictions (Anderson, Burnham 2004).

In the context of stories, this can manifest itself in a lack of flexibility. There is potential for the model to simply replicate existing stories or mash them together inelegantly, trying to hard to match said examples.

Ideally, unsupervised learning will help with this, as the more data is fed to the model the better. Also the use of a separate verification dataset is key, allowing you to make sure the model isn't simply performing well at predicting what it already knows.

5. Limitations

After taking a broad approach to the research in the initial phases, a conscious decision was made to narrow the focus to a particular subset of stories. This allows the potential for more significant progress, albeit in a more narrow field.

Concerns were raised, too, over the level at which the topic should be approached. We toyed with the idea of creating a new transformer architecture, but ultimately this was deemed infeasible. Instead, the decision was made to take an existing model – a more barebones, untrained GPT-2 – and modify this to narrow it down and see what it can achieve with more focus.

Ultimately, what we settled on was starting with this basic transformer model, focusing on a certain type of story ("children's cautionary tales", for the inappropriate productions as much as the appropriate ones), then optimising the model and training data to see what we can come up with. This allows the opportunity to delve deeper instead of trying to generalise and conquer every domain at once, with limited time and manpower.

Transformers haven't been used much for novel-like story generation – mainly translation and more "news article" style text – so this is an relatively new application of the technology, keeping things interesting.

Issues that remained were the task of how exactly to go about this modification, and how to formalise elements of good stories that I discussed previously, or whether that should be tackled at all.

6. Modification

We began thinking about different methods of "modification", both based on drafting.

6.1 Self-Referential

The straightforward approach would be to have the model create and then revise its productions. In the same way an author would go over their own work, the model would iterate multiple times.

However, this requires the model to evaluate and revise its own work, and if it was capable of identifying what was "good", then it could probably just generate this the first time. I'm not sure how to qualify how this evaluation and revision should be made. This goes back to the original difficulty of evaluation.

6.2 Multiple Transformers

The possibility of multiple transformers interacting with each other was then theorised, each with different training data, to create a kind of virtual peer-review system. No great story is written without the input of others, so why should AI be expected to do so?

The idea would be to begin with GPT-2 for a first draft, then pass this through (many iterations of) itself or another Transformer. The of bouncing multiple models off each other had some promise, and a good candidate for this was BERT.

Bidirectional Encoder Representations from Transformers (BERT) is a transformer developed by Google mainly for language translation and keyword analysis (Devlin et al 2018). Its recent application to Google Search was highlighted as one of the most significant changes to the platform since its inception. We wondered if this could similarly be used to "translate" the existing draft into a better one, in a way.

Again, it was unclear what the inner workings of such a system would look like, but it seemed promising. We had initially veered away from BERT since it's not ideal for language generation, but this would be an interesting way to involve multiple systems with different strengths.

6.3 Overlay

An approach to managing these different iterations and/or types of Transformer interacting with each other, would be to have an overlay system of sorts. This could manage the interactions, allow different types of user input, and allow an avenue into modification.

The models themselves would be trained separately on a cloud computing platform – this is the most computationally intensive task – but then our overlay would take those in and perform the generation step.

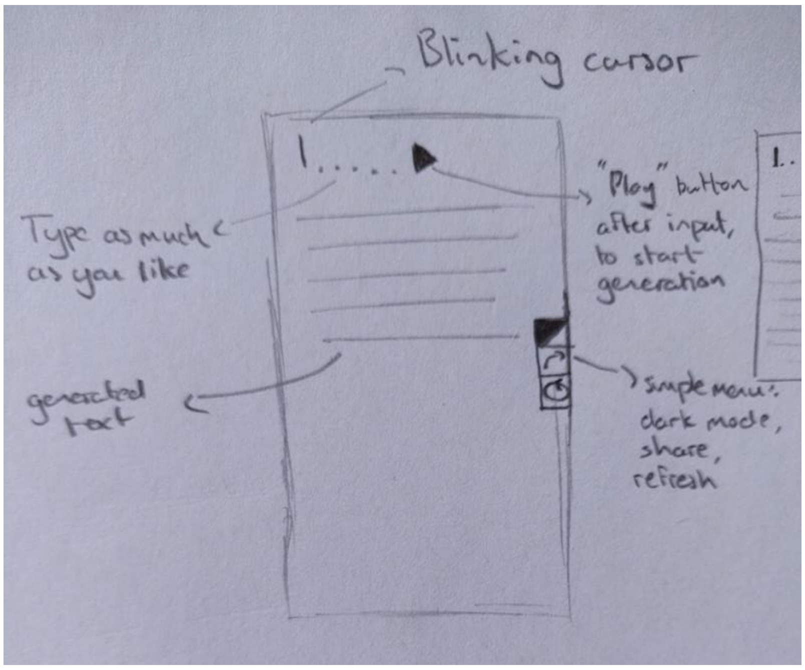
Perhaps directly altering the Transformer isn't the way to go – rather this route, which opens up more customisation.

7. User Interface

We have talked at length about the theory, the underlying logic, the problems faced while generating the stories etc, but hitherto neglected has been the issue of how this will be presented to the end user, as a product.

This is a secondary concern, since the UI will be fairly basic and just a user-friendly representation of the core challenge of the project, but it is important to present this information in an easily accessible and enjoyable way. Text generation isn't very exciting to watch, but users will want an easy way to see what we come up with.

To that end, we propose a simple web application, which offers the option to generate a story entirely randomly (to the extent any computer system can be random), or related to a user-entered "seed", similar to some currently existing websites e.g. https://talktotransformer.com/. A (multiplatform-friendly) mockup is included overleaf.


Blinking cursor
Type as much as you like
generated text
"Play" button after input, to start generation
Simple menu: dark mode, share, refresh

8. Project Management

This being the longest term project I have taken on so far, we were conscious of taking steps to ensure I didn't drown in deliverables or become overwhelmed by the overall task. Thus, some preparations were made to manage progress.

8.1 Timeline

With a high-level picture of the project (fig. 1), I reckoned that it would be easier to allocate my time and track progress relative to my original intentions. I'm not so idealistic as to think everything will go according to plan, but it will help. Deadlines are also clearly marked.

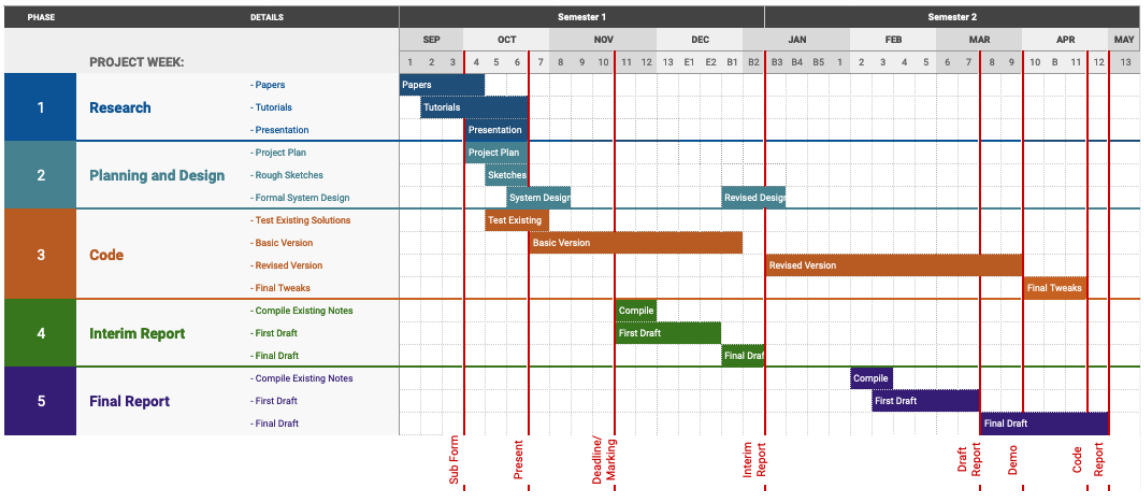


Fig.1 Project Timeline

8.2 Task Tracking

While high level abstraction is nice, I also wanted a way to manage the finer points, down to individual tasks. It's easy to be swallowed up by the enormity of the project, so I hope this will help focus me.

For this, I used Asana (fig. 2). This is a kanban board, popular with project managers handling teams, but also useful on an individual level.

Features include coloured tags, deadlines, calendar and list views for upcoming tasks et al.

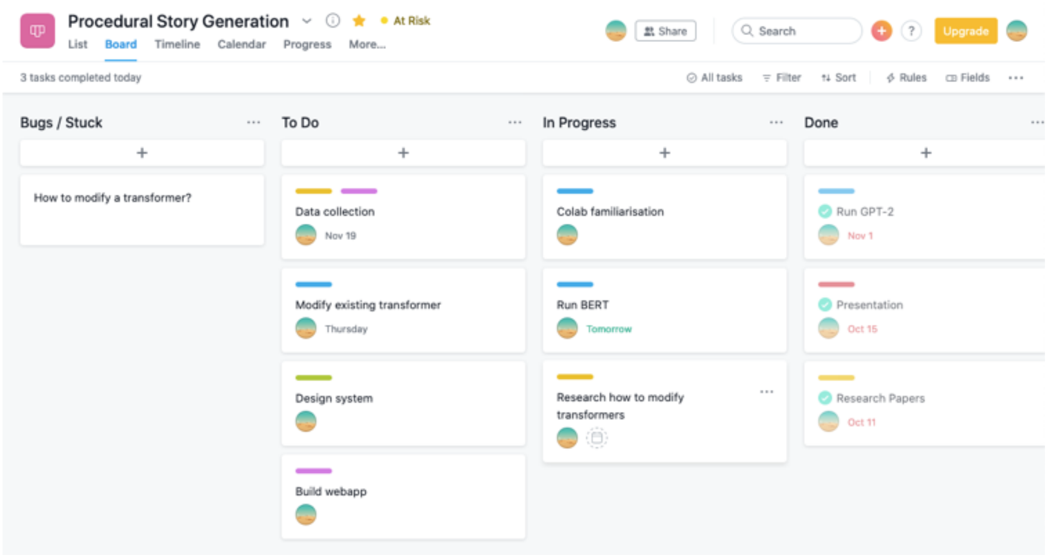


Fig. 2 Task Tracking with Asana

9. Conclusion and Further Research

We believe that the core work in identifying the problem has been adequately researched, and from now on the focus will be on specifics and implementation.

Transformers are a highly state-of-the-art, complex concept which will require deeper reading and experimentation in order to fully understand their inner workings, which is a prerequisite for modification. The plan is to look at existing code (for the Transformers themselves as well as novel implementations) and continue reading more walkthroughs of the topic.

While some perusing of code has been done, along with some basic training of a Transformer model through resources online, we intend to properly kickstart the programming aspect of the project. This is a research heavy topic, but my own implementation is still necessary, both to capture the interest of observers, and to prove my understanding and familiarity with the concepts discussed.

References

Anderson, D. and Burnham, K., 2004. Model selection and multi-model inference. *Second. NY: Springer-Verlag*, *63*.

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

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*.

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), pp.855-868.

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

Hinton, G.E., Sejnowski, T.J. and Poggio, T.A. eds., 1999. *Unsupervised learning: foundations of neural computation*. MIT press.

Mohri, M., Rostamizadeh, A. and Talwalkar, A., 2018. *Foundations of machine learning*. MIT press.

Propp, V., 2010. *Morphology of the Folktale* (Vol. 9). University of Texas Press.

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

Trottier, D., 1998. *The Screenwriter's Bible. A Complete Guide to Writing, Formatting, and Selling Your Script, Expanded &Updated*. Los Angeles: Silman-James Press.

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).