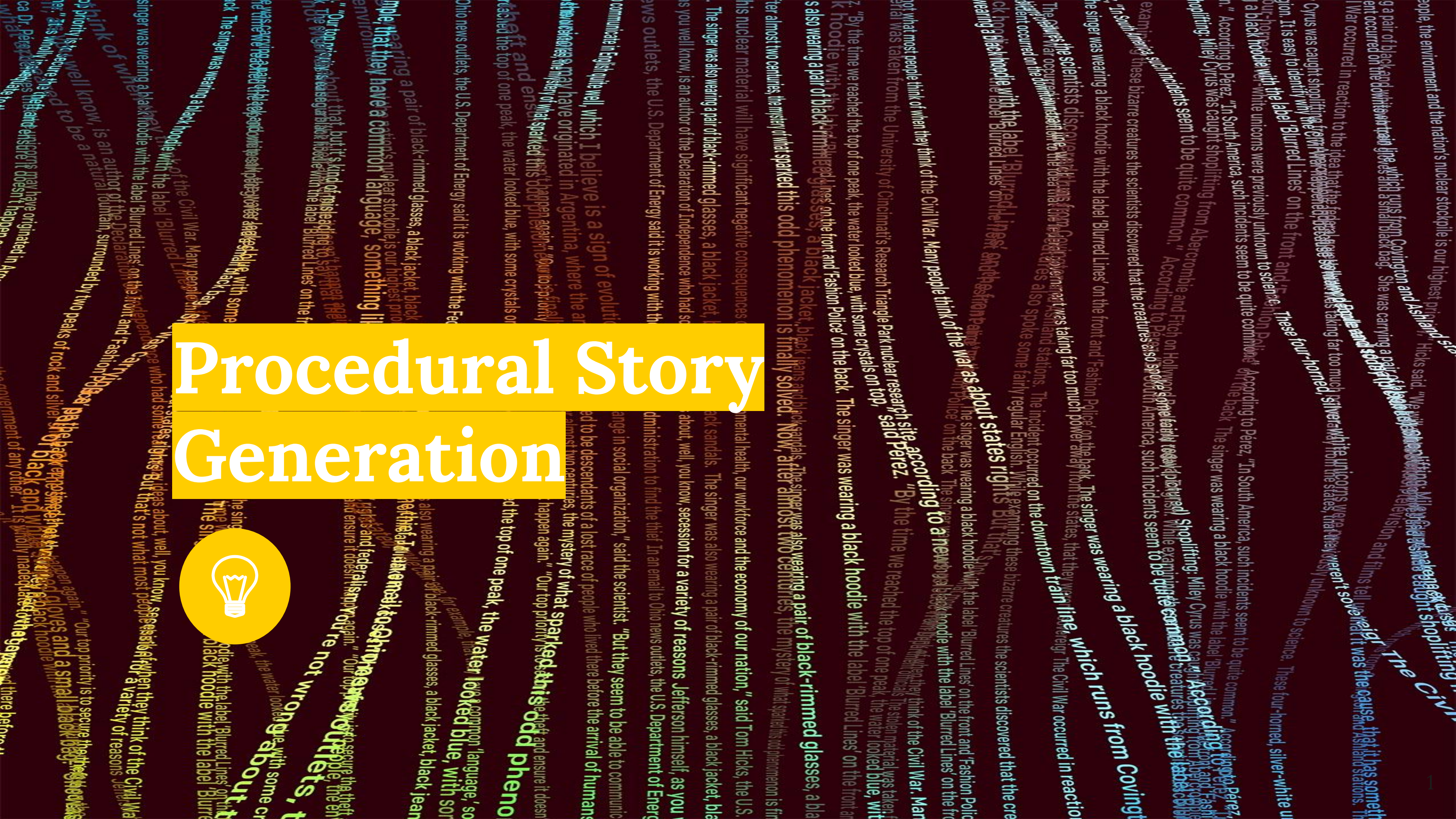


# Procedural Story Generation

# Hello!

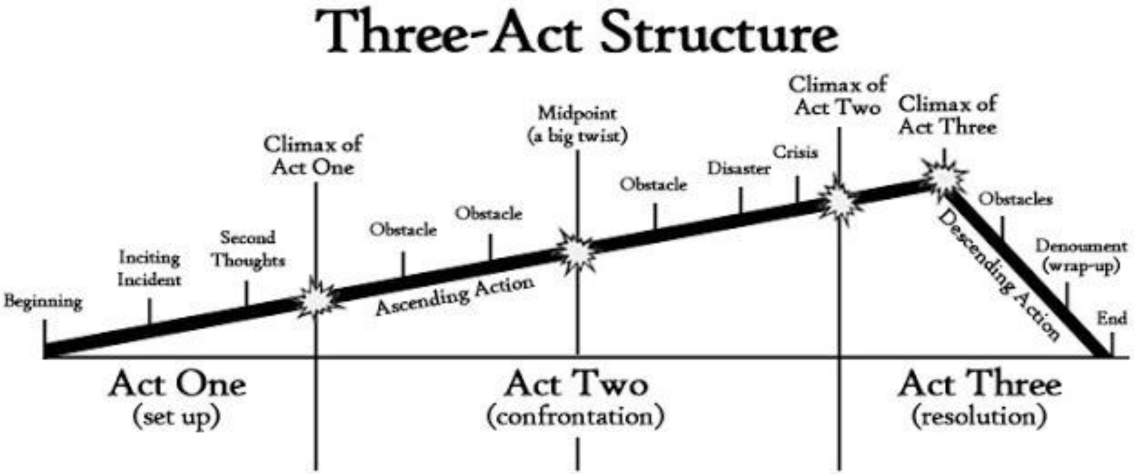
## *I’m **Niall Dillane***

BSc. Computer Systems student, under the supervision of James Patten, researching the topic of Procedural Story Generation

# Original Summary

- Research into procedural generation methods
- A technique that has applications across a variety of domains
- Structure and language of stories… “What are the elements of a good story?”, “How can these elements be codified in a manner that they can be used by a procedural generation technique?”

# Outline, Plan, Revise



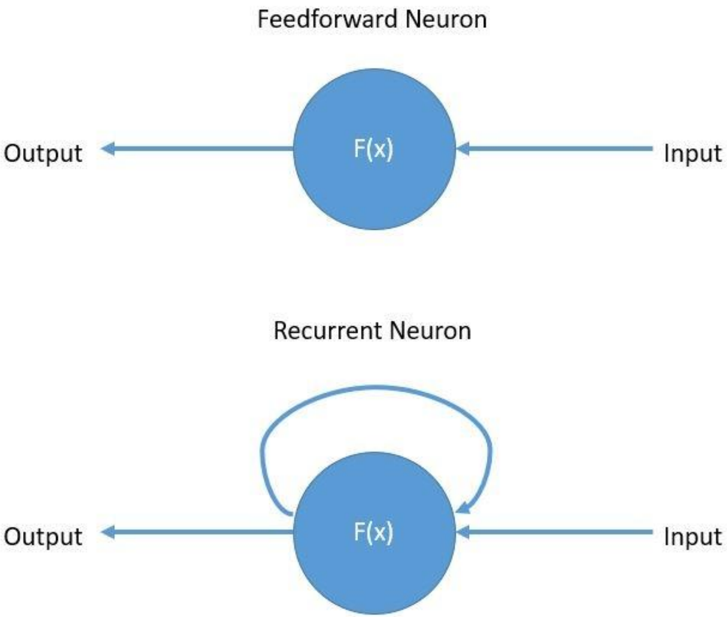
nownovel.com

From a seed (either user input or autonomously generated), create an outline with events that must occur, fill this out further into a detailed plan, then complete it with sentences, made of words

A tree-like approach.

# Neural Networks

Why try to teach a program the English language and creativity, when it can learn itself?



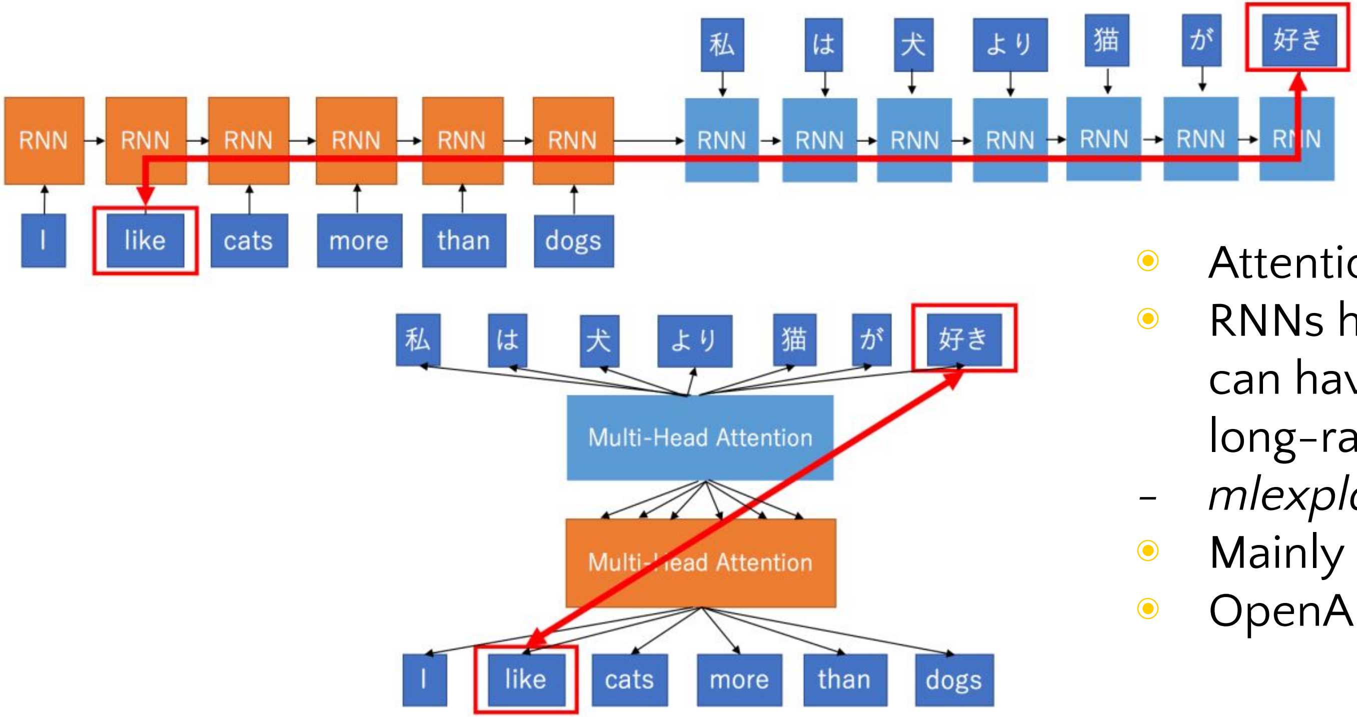
wiki.tum.de

## RNNs have persistence, memory

Traditional neural networks have struggled with using reasoning from previous events to inform later ones, but RNNs have loops, passing information from one step to the next, allowing information to persist.

These are ideal for video and language related tasks, where you don’t want to start afresh every time. (Young, Tom et al., 2018)

# Transformers – even better?



- Attention Is All You Need
- RNNs hard to parallelize and can have difficulty learning long-range dependencies
- – *mlexplained.com*
- Mainly applied to translation
- OpenAI GPT-2

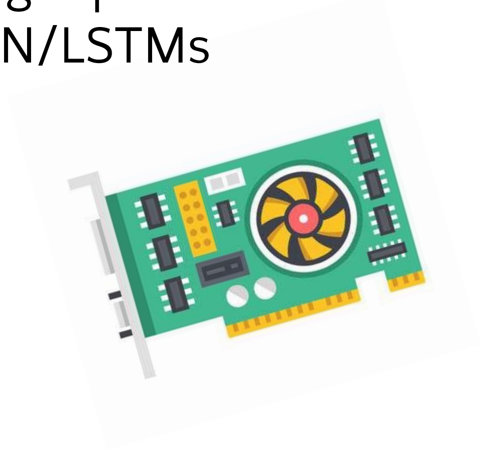
# Technology to be used

## TensorFlow

Developed by Google. Has a larger userbase and longer history, and more support available. Also looks to be ahead in the race with Transformers, the latest NLP technique

## PyTorch

Developed by Facebook. More user-friendly, easier to debug, stronger performance and with RNN/LSTMs



# Work So Far

## Research

Heavy topic with lots of large companies and universities behind it.

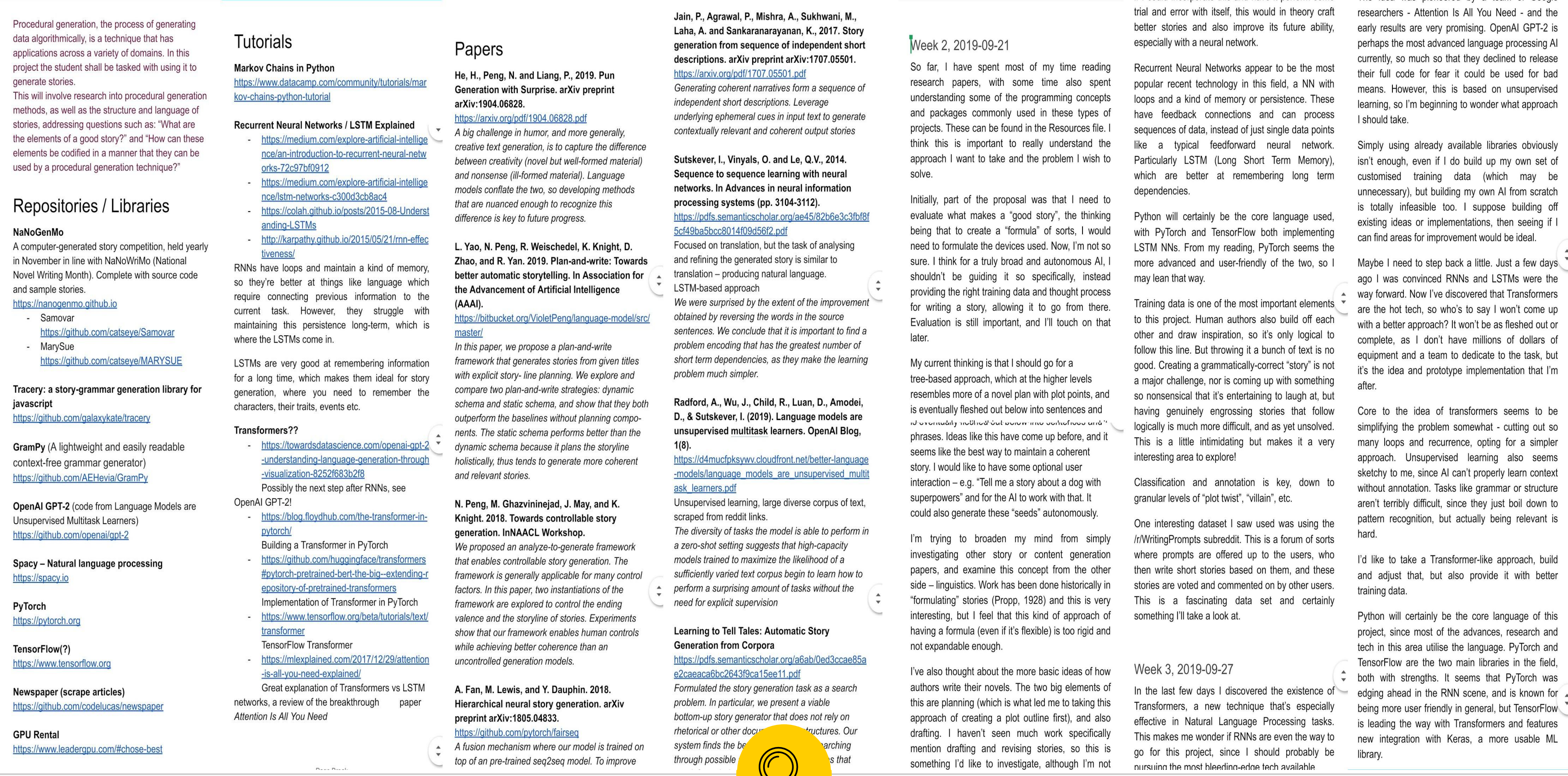
Constantly evolving, RNNs to Transformers to who knows what?

## Documenting

Keeping a document full of links and personal summaries of papers/tutorials, as well as a journal discussing my thoughts each week.

## Prototyping

Trying out existing libraries and solutions in this space, running with sample data, getting a feel for the tools and techniques.



Procedural generation, the process of generating data algorithmically, is a technique that has applications across a variety of domains. In this project the student shall be tasked with using it to generate stories.
This will involve 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?" and "How can these elements be codified in a manner that they can be used by a procedural generation technique?"

## Repositories / Libraries

**NaNoGenMo**
A computer-generated story competition, held yearly in November in line with NaNoWriMo (National Novel Writing Month). Complete with source code and sample stories.
https://nanogenmo.github.io

- Samovar
  https://github.com/catseye/Samovar
- MarySue
  https://github.com/catseye/MARYSUE

**Tracery: a story-grammar generation library for javascript**
https://github.com/galaxykate/tracery

**GramPy** (A lightweight and easily readable context-free grammar generator)
https://github.com/AEHevia/GramPy

**OpenAI GPT-2** (code from Language Models are Unsupervised Multitask Learners)
https://github.com/openai/gpt-2

**Spacy – Natural language processing**
https://spacy.io

**PyTorch**
https://pytorch.org

**TensorFlow(?)**
https://www.tensorflow.org

**Newspaper (scrape articles)**
https://github.com/codelucas/newspaper

**GPU Rental**
https://www.leadergpu.com/#chose-best

## Tutorials

**Markov Chains in Python**
https://www.datacamp.com/community/tutorials/markov-chains-python-tutorial

**Recurrent Neural Networks / LSTM Explained**

- https://medium.com/explore-artificial-intelligence/an-introduction-to-recurrent-neural-networks-72c97bf0912
- https://medium.com/explore-artificial-intelligence/lstm-networks-c300d3cb8ac4
- https://colah.github.io/posts/2015-08-Understanding-LSTMs
- http://karpathy.github.io/2015/05/21/rnn-effectiveness/

RNNs have loops and maintain a kind of memory, so they're better at things like language which require connecting previous information to the current task. However, they struggle with maintaining this persistence long-term, which is where the LSTMs come in.

LSTMs are very good at remembering information for a long time, which makes them ideal for story generation, where you need to remember the characters, their traits, events etc.

**Transformers??**

- https://towardsdatascience.com/openai-gpt-2-understanding-language-generation-through-visualization-8252f683b2f8
  Possibly the next step after RNNs, see OpenAI GPT-2!
- https://blog.floydhub.com/the-transformer-in-pytorch/
  Building a Transformer in PyTorch
- https://github.com/huggingface/transformers#pytorch-pretrained-bert-the-big--extending-repository-of-pretrained-transformers
  Implementation of Transformer in PyTorch
- https://www.tensorflow.org/beta/tutorials/text/transformer
  TensorFlow Transformer
- https://mlexplained.com/2017/12/29/attention-is-all-you-need-explained/
  Great explanation of Transformers vs LSTM networks, a review of the breakthrough paper *Attention Is All You Need*

## Papers

**He, H., Peng, N. and Liang, P., 2019. Pun Generation with Surprise. arXiv preprint arXiv:1904.06828.**
https://arxiv.org/pdf/1904.06828.pdf
*A big challenge in humor, and more generally, creative text generation, is to capture the difference between creativity (novel but well-formed material) and nonsense (ill-formed material). Language models conflate the two, so developing methods that are nuanced enough to recognize this difference is key to future progress.*

**L. Yao, N. Peng, R. Weischedel, K. Knight, D. Zhao, and R. Yan. 2019. Plan-and-write: Towards better automatic storytelling. In Association for the Advancement of Artificial Intelligence (AAAI).**
https://bitbucket.org/VioletPeng/language-model/src/master/
*In this paper, we propose a plan-and-write framework that generates stories from given titles with explicit story- line planning. We explore and compare two plan-and-write strategies: dynamic schema and static schema, and show that they both outperform the baselines without planning components. The static schema performs better than the dynamic schema because it plans the storyline holistically, thus tends to generate more coherent and relevant stories.*

**N. Peng, M. Ghazvininejad, J. May, and K. Knight. 2018. Towards controllable story generation. InNAACL Workshop.**
*We proposed an analyze-to-generate framework that enables controllable story generation. The framework is generally applicable for many control factors. In this paper, two instantiations of the framework are explored to control the ending valence and the storyline of stories. Experiments show that our framework enables human controls while achieving better coherence than an uncontrolled generation models.*

**A. Fan, M. Lewis, and Y. Dauphin. 2018. Hierarchical neural story generation. arXiv preprint arXiv:1805.04833.**
https://github.com/pytorch/fairseq
*A fusion mechanism where our model is trained on top of an pre-trained seq2seq model. To improve*

**Jain, P., Agrawal, P., Mishra, A., Sukhwani, M., Laha, A. and Sankaranarayanan, K., 2017. Story generation from sequence of independent short descriptions. arXiv preprint arXiv:1707.05501.**
https://arxiv.org/pdf/1707.05501.pdf
*Generating coherent narratives form a sequence of independent short descriptions. Leverage underlying ephemeral cues in input text to generate contextually relevant and coherent output stories*

**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).**
https://pdfs.semanticscholar.org/ae45/82b6e3c3fbf85cf49ba5bcc8014f09d56f2.pdf
Focused on translation, but the task of analysing and refining the generated story is similar to translation – producing natural language.
LSTM-based approach
*We were surprised by the extent of the improvement obtained by reversing the words in the source sentences. We conclude that it is important to find a problem encoding that has the greatest number of short term dependencies, as they make the learning problem much simpler.*

**Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., & Sutskever, I. (2019). Language models are unsupervised multitask learners. OpenAI Blog, 1(8).**
https://d4mucfpksywv.cloudfront.net/better-language-models/language_models_are_unsupervised_multitask_learners.pdf
Unsupervised learning, large diverse corpus of text, scraped from reddit links.
*The diversity of tasks the model is able to perform in a zero-shot setting suggests that high-capacity models trained to maximize the likelihood of a sufficiently varied text corpus begin to learn how to perform a surprising amount of tasks without the need for explicit supervision*

**Learning to Tell Tales: Automatic Story Generation from Corpora**
https://pdfs.semanticscholar.org/a6ab/0ed3ccae85ae2caeaca6bc2643f9ca15ee11.pdf
*Formulated the story generation task as a search problem. In particular, we present a viable bottom-up story generator that does not rely on rhetorical or other docu structures. Our system finds the be arching through possible s that*

## Week 2, 2019-09-21

So far, I have spent most of my time reading research papers, with some time also spent understanding some of the programming concepts and packages commonly used in these types of projects. These can be found in the Resources file. I think this is important to really understand the approach I want to take and the problem I wish to solve.

Initially, part of the proposal was that I need to evaluate what makes a "good story", the thinking being that to create a "formula" of sorts, I would need to formulate the devices used. Now, I'm not so sure. I think for a truly broad and autonomous AI, I shouldn't be guiding it so specifically, instead providing the right training data and thought process for writing a story, allowing it to go from there. Evaluation is still important, and I'll touch on that later.

My current thinking is that I should go for a tree-based approach, which at the higher levels resembles more of a novel plan with plot points, and is eventually fleshed out below into sentences and phrases. Ideas like this have come up before, and it seems like the best way to maintain a coherent story. I would like to have some optional user interaction – e.g. "Tell me a story about a dog with superpowers" and for the AI to work with that. It could also generate these "seeds" autonomously.

I'm trying to broaden my mind from simply investigating other story or content generation papers, and examine this concept from the other side – linguistics. Work has been done historically in "formulating" stories (Propp, 1928) and this is very interesting, but I feel that this kind of approach of having a formula (even if it's flexible) is too rigid and not expandable enough.

I've also thought about the more basic ideas of how authors write their novels. The two big elements of this are planning (which is what led me to taking this approach of creating a plot outline first), and also drafting. I haven't seen much work specifically mention drafting and revising stories, so this is something I'd like to investigate, although I'm not

trial and error with itself, this would in theory craft better stories and also improve its future ability, especially with a neural network.

Recurrent Neural Networks appear to be the most popular recent technology in this field, a NN with loops and a kind of memory or persistence. 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), which are better at remembering long term dependencies.

Python will certainly be the core language used, with PyTorch and TensorFlow both implementing LSTM NNs. From my reading, PyTorch seems the more advanced and user-friendly of the two, so I may lean that way.

Training data is one of the most important elements to this project. Human authors also build off each other and draw inspiration, so it's only logical to follow this line. But throwing it a bunch of text is no good. Creating a grammatically-correct "story" is not a major challenge, nor is coming up with something so nonsensical that it's entertaining to laugh at, but having genuinely engrossing stories that follow logically is much more difficult, and as yet unsolved. This is a little intimidating but makes it a very interesting area to explore!

Classification and annotation is key, down to granular levels of "plot twist", "villain", etc.

One interesting dataset I saw used was using the /r/WritingPrompts subreddit. This is a forum of sorts where prompts are offered up to the users, who then write short stories based on them, and these stories are voted and commented on by other users. This is a fascinating data set and certainly something I'll take a look at.

## Week 3, 2019-09-27

In the last few days I discovered the existence of Transformers, a new technique that's especially effective in Natural Language Processing tasks. This makes me wonder if RNNs are even the way to go for this project, since I should probably be pursuing the most bleeding-edge tech available

researchers - Attention Is All You Need - and the early results are very promising. OpenAI GPT-2 is perhaps the most advanced language processing AI currently, so much so that they declined to release their full code for fear it could be used for bad means. However, this is based on unsupervised learning, so I'm beginning to wonder what approach I should take.

Simply using already available libraries obviously isn't enough, even if I do build up my own set of customised training data (which may be unnecessary), but building my own AI from scratch is totally infeasible too. I suppose building off existing ideas or implementations, then seeing if I can find areas for improvement would be ideal.

Maybe I need to step back a little. Just a few days ago I was convinced RNNs and LSTMs were the way forward. Now I've discovered that Transformers are the hot tech, so who's to say I won't come up with a better approach? It won't be as fleshed out or complete, as I don't have millions of dollars of equipment and a team to dedicate to the task, but it's the idea and prototype implementation that I'm after.

Core to the idea of transformers seems to be simplifying the problem somewhat - cutting out so many loops and recurrence, opting for a simpler approach. Unsupervised learning also seems sketchy to me, since AI can't properly learn context without annotation. Tasks like grammar or structure aren't terribly difficult, since they just boil down to pattern recognition, but actually being relevant is hard.

I'd like to take a Transformer-like approach, build and adjust that, but also provide it with better training data.

Python will certainly be the core language of this project, since most of the advances, research and tech in this area utilise the language. PyTorch and TensorFlow are the two main libraries in the field, both with strengths. It seems that PyTorch was edging ahead in the RNN scene, and is known for being more user friendly in general, but TensorFlow is leading the way with Transformers and features new integration with Keras, a more usable ML library.

# My Approach

## Existing Model

Basic, untrained model (GPT-2)

Open source code

No use starting from scratch

jalammar.github.io

## Modification

Little work done on novel-like story style generation

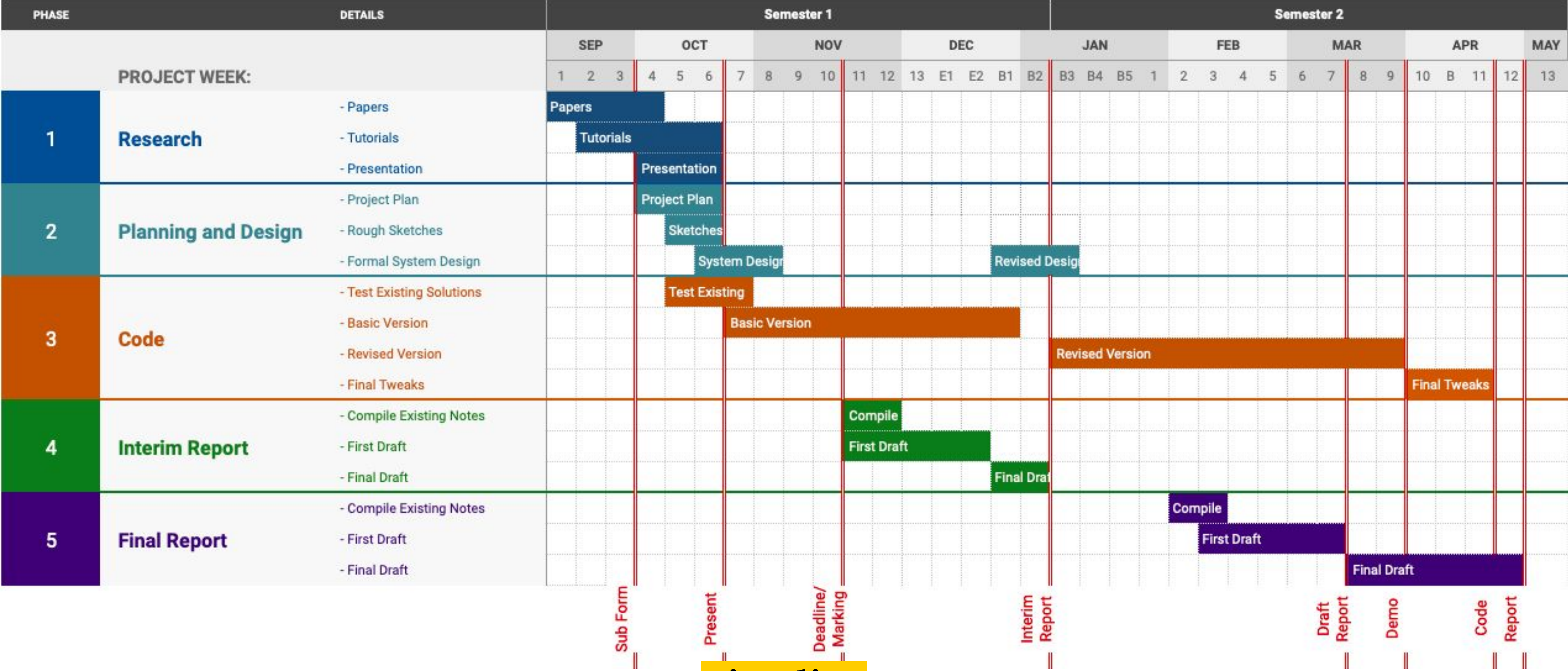
Adjust model to include structure and elements of stories, more focused

Formulate creativity?

## Training

Concentrate on specific type of story, Children’s cautionary tales.

Focus on optimising the model and producing working examples, generalise later



| PHASE | | DETAILS | Semester 1 | | | | | | | | | | | | | | | | | Semester 2 | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | SEP | | | OCT | | | | NOV | | | | | DEC | | | | JAN | | | | | FEB | | | | MAR | | | | APR | | | | MAY |
| | PROJECT WEEK: | | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | E1 | E2 | B1 | B2 | B3 | B4 | B5 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | B | 11 | 12 | 13 |
| 1 | Research | - Papers | Papers | ▨ | ▨ | ▨ | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| | | - Tutorials | | Tutorials | ▨ | ▨ | ▨ | ▨ | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| | | - Presentation | | | | Presentation | ▨ | ▨ | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 2 | Planning and Design | - Project Plan | | | | Project Plan | ▨ | ▨ | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| | | - Rough Sketches | | | | | Sketches | ▨ | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| | | - Formal System Design | | | | | | System Desig | ▨ | ▨ | | | | | | | | Revised Desig | ▨ | ▨ | | | | | | | | | | | | | | | | |
| 3 | Code | - Test Existing Solutions | | | | | Test Existing | ▨ | ▨ | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| | | - Basic Version | | | | | | | Basic Version | ▨ | ▨ | ▨ | ▨ | ▨ | ▨ | ▨ | ▨ | ▨ | | | | | | | | | | | | | | | | | | |
| | | - Revised Version | | | | | | | | | | | | | | | | | | Revised Version | ▨ | ▨ | ▨ | ▨ | ▨ | ▨ | ▨ | ▨ | ▨ | ▨ | ▨ | | | | | |
| | | - Final Tweaks | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | Final Tweaks | ▨ | ▨ | | |
| 4 | Interim Report | - Compile Existing Notes | | | | | | | | | | | Compile | ▨ | | | | | | | | | | | | | | | | | | | | | | |
| | | - First Draft | | | | | | | | | | | First Draft | ▨ | ▨ | ▨ | ▨ | | | | | | | | | | | | | | | | | | | |
| | | - Final Draft | | | | | | | | | | | | | | | | Final Dra | ▨ | | | | | | | | | | | | | | | | | |
| 5 | Final Report | - Compile Existing Notes | | | | | | | | | | | | | | | | | | | | | | Compile | ▨ | | | | | | | | | | | |
| | | - First Draft | | | | | | | | | | | | | | | | | | | | | | | First Draft | ▨ | ▨ | ▨ | ▨ | | | | | | | |
| | | - Final Draft | | | | | | | | | | | | | | | | | | | | | | | | | | | | | Final Draft | ▨ | ▨ | ▨ | ▨ | ▨ | |
| | | | | | Sub Form | | | Present | | | | Deadline/ Marking | | | | | | | Interim Report | | | | | | | | | | Draft Report | | Demo | | | Code | Report | |

# Timeline

# Credits

Special thanks to all the people who made and released these great resources for free:

- Slide templates by SlidesCarnival
- Images from OpenAI, nownovel.com, wiki.tum.de, pixabay.com, mlexplained.com
- Young, T., Hazarika, D., Poria, S. and Cambria, E., 2018. Recent trends in deep learning based natural language processing

# Thanks!

Any questions ?