Amy Read

Generating Controllable Story Plots with Reinforcement Learning

Computer Science Tripos – Part II

Murray Edwards College

May, 2023

Declaration of originality

I, Amy Read of Murray Edwards College, being a candidate for Part II of the Computer Science Tripos, hereby declare that this dissertation and the work described in it are my own work, unaided except as may be specified below, and that the dissertation does not contain material that has already been used to any substantial extent for a comparable purpose. I am content for my dissertation to be made available to the students and staff of the University.

Signed Amy Read

Date May 19, 2023

Proforma

Candidate Number:	**2057G**
Project Title:	**Generating Controllable Story Plots with Reinforceme**
Examination:	**Computer Science Tripos – Part II, May, 2023**
Word Count:	**8976**[1]
Code Line Count:	**1738**
Project Originator:	**Amy Read**
Supervisor:	**Russell Moore, Hope McGovern**

Original Aims of the Project

This project aims to replicate the ideas and methods of Martin *et al.*'s 2019 paper *Controllable Neural Story Plot Generation via Reward-Shaping* [13]: training a neural network to generate story plots from a corpus of plot summaries and using reward-shaping to focus the outputs. I will compare results from different models trained on different datasets.

Work Completed

All that has been completed appears in this dissertation.

[1]This word count was computed using the TeXcount web service (`https://app.uio.no/ifi/texcount/online.php`), excluding proforma, bibliography and appendices.

Contents

List of Figures

Chapter 1

Introduction

This project will focus on using neural networks to generate story plots that move towards a given goal. This motivation is inspired by the current common applications of Artificial Intelligence (AI) tools for Natural Language Generation (NLG) for predictive writing to fulfil prompts and write stories, and the pre-existing culture among writers of using limited 'writing prompts' as inspirational resources when planning a novel.

Using artificial intelligence tools for writing has been a frequent discussion point in recent media coverage as increasingly more sophisticated and easy-to-use text generation models have been released in quick succession. ChatGPT, for example, is an AI chatbot released by OpenAI in 2022, designed to interact with humans in a conversational way [16]. ChatGPT can be used to generate text for story-writing based on an outline using a technique known as prompt-engineering, i.e. by inputting prompts such as "Write a story about a human falling in love with a vampire," and ChatGPT will generate a response based on its training data. The limitations of such AI models for writing have been highlighted as a tendency towards hallucination, and ingrained plagiarism as the black box model is unable to cite its sources.

Novel prose requires authorial skill in writing in an eloquent, enjoyable style; in addition to writing coherent, readable sentences, constant recall of consistent information established earlier in the text, and overall refinement of a likely very long body of text. This task itself is difficult to automate with predictive writing due to its innate drawbacks existing in contrast to the latter two goals: if you let a model create long continuations of any prompt (even a good one), the results will become less relevant to the initial prompt over time due to accumulation of inconsistencies as it contributes to its own prompt. It is difficult to direct and significant human corrections become more and more necessary, which professional users may find frustrating and time-consuming.

Due to these limitations of utilising AI to write prose for novels, this project will focus on instead inspiring a human creator in designing the novel content. This gives a human author more responsibility and freedom in transforming the suggestions into their own unique prose, such that they can better ensure the resulting work can be described as 'original', and they will be encouraged to extrapolate details that might be missed by

an automated writer.

The task of planning fiction stories can be abstracted into three main subtasks: designing characters, worldbuilding (designing setting), and plotting (designing and sequencing events). Designing details for the author to interpret generally requires less information recall and shorter responses than composing prose. This project focuses on automated plotting – generating sequences of events in a story – which is more difficult to automate than the other two since it requires greater information recall for continuity and longer responses, as well as grammatical correctness.

This project aims to replicate the ideas and methods of Martin *et al.*'s 2019 paper *Controllable Neural Story Plot Generation via Reward-Shaping* [13]: training a neural network to generate story plots from a corpus of plot summaries and using reward-shaping to focus the outputs.

I will compare metrics calculated from the outputs of different models, do a participant study where human writers evaluate the outputs of a selected model, and discuss the implications of these results.

Chapter 2

Preparation

2.1 Background

2.1.1 Reinforcement Learning

Reinforcement learning is a goal-directed machine learning paradigm, whereby a learning agent seeks to maximise a numerical reward signal rather than find a hidden structure (like in supervised and unsupervised learning paradigms). The learning agent is able to measure the state of its environment and take actions which affect this state: it discovers which actions yield the highest reward by trying them, and then prefers to take actions that it has tried in the past and found to be effective. [23]

Reinforcement learning [23] aims to solve a Markov Decision Process (MDP), which a learning agent must work through in order to maximise a set of rewards. This MDP is a mathematical framework which models decision-making where outcomes are partially random and partially under the control of an agent, which defines the probability of transitioning into a new state and receiving some reward based on the current state and an action taken. A policy is the probability distribution of actions given its parameters and some state, which defines what actions should be taken in each state in order to maximise the expected future reward [17]. A MDP is defined as a tuple $M = \langle S, A, T, R, \gamma \rangle$ where S is the set of states, A is the set of actions, T is the state transition function $T : S \times A \to P(S)$, R is a reward function $R : S \times A \to R$, and γ is a discount factor $0 \leq \gamma \leq 1$. The result of reinforcement learning is a policy $\pi : S \to A$. [23]

Generating a story plot by iteratively sampling from a language model does not ensure that the plot will move towards or reach a desired goal except by coincidence, and this often leads to divergence instead of convergence to a goal, as it involves continuous actions being taken in high dimensional spaces. Thus, Martin *et al.* use reinforcement learning to guide the events of a generated story [13]. Reinforcement learning addresses progression and control issues in a story generation problem as moves are guided towards a specific goal by rewards provided at each step, based on a learnt policy that maximises future expected reward.

2.1.2 Reward-Shaping

The policy gradient algorithm adjusts the parameters θ (the weights in a neural network) of a policy π in the direction of the performance gradient in order to maximize the expected long-term cumulative reward. [21]

Reward-shaping [15] is a policy gradient approach to reinforcement learning, whereby sparse rewards, which are provided to the learning agent only when a given goal is reached, are supplemented with additional rewards at intermediate states leading to the goal whenever progress is made. These intermediate rewards are backpropagated into the pre-trained language model, encouraging generation of events that make significant progress towards the goal. The agent maximises the policy gradient, thus learning a policy which maximises the future expected reward.

Martin *et al.* [13] present a reward-shaping approach for a training corpus to be analysed to construct a dense reward function for guiding the underlying language model toward a given goal. This reward function is used in their policy gradient descent function, using the REINFORCE algorithm [24], to reward the network whenever a generated plot event makes it more likely to achieve a given goal and thereby specialise the language model to prefer to take steps which move the plot toward a given goal.

2.2 Tools Used

Stanford CoreNLP: A natural language processing Java library for linguistic annotation of text. I use the Part-of-Speech (PoS) tagging, Named Entity Recognition (NER) and dependency parsing components to transform a dataset of raw natural language story summaries into a special 'event' format. [11]

Keras: A deep learning API in Python. I use this to define a neural network. I chose to use this as it is the machine learning framework with which I am most familiar.

WordNet: A large lexical database of semantic relations between words. Cognitive synonyms are grouped into 'synsets'. [14]

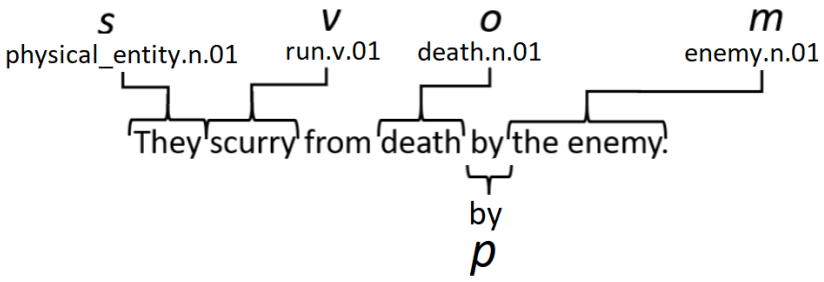
2.3 Dataset Transformation

2.3.1 Event Representation

A story is represented as a sequence of 'events' derived from a single summary. Extended from the definition of Martin *et al.*, an event is a semantic abstraction of a sentence, represented as a tuple $e = \langle s, v, o, p, m \rangle$, where v is the verb of a sentence, s is the subject of the verb, o is the object of the verb, p is a preposition, and m is a propositional object, indirect object, causal complement, or any other significant noun modifier. [1] The parameters s, o and m may take the value None but s and v must always have values.

From the summary corpus, texts are broken down into sequences of events; such that sentences may be split into multiple events, with a one-to-many relationship between a sentence and the events it produces. [13]

Figure 2.1: An example sentence and its event representation [13]



To transform raw text summaries into this event format, they are first parsed in the `dependency_parse` function using the Stanford CoreNLP [11] parser for tokenisation and lemmatisation, dependency parser to get a dependency parse tree for each sentence, and the POS tagger.

In the `eventify` function, event translation of the annotated summaries is performed: extract events from each sentence using the parse tree relations and POS labels to identify verbs and the words depending on those verbs. Provided that each of the verbs have an identifiable subject, verbs have a 1:1 relationship with derived events.

2.3.2 Data

In an earlier paper on a similar subject to their 2019 paper that this project is based on [13], Martin *et al.* [12] used the CMU Movie Summary Corpus [1] as a base corpus for transformation into described event format. [1] is a dataset of movie plot summaries extracted from Wikipedia. Here, I aim to implement the same strategy. They found it to be too diverse; as there is high variance between stories, which dilutes event patterns, and thus used subsets of it from clusters derived using Latent Dirichlet Analysis. I used a subset of this same dataset, limited to stories tagged with the genre 'romance', to achieve a similar purpose. All of the functions for transforming the corpus (detailed below) are found in `process_data.py`.

2.3.3 Named Entity Recognition

Specific names of people and places hold little relevance to the goal of generating plot events; they add unnecessary noise to the training data, they occur very sparsely and greatly inflate the size of the vocabulary, and they may also correspond to intellectual property when used together. Named Entity Recognition (NER) [3] is performed on the raw text summaries using the Stanford CoreNLP NER Parser [11] to label proper nouns and named entities in the `create_entity_mask` function, so they can be represented similarly as identical values for $\langle PERSON \rangle$, $\langle LOCATION \rangle$, $\langle TIME \rangle$, etc.

Martin *et al.* [12] use this technique in their 2018 paper to increase the level of abstraction within events (and thus decrease sparsity in the vocabulary). They implement 'sentence NEs', where $\langle PERSON \rangle$ entities are numbered per-sentence, and 'continued NEs', where $\langle PERSON \rangle$ entities are numbered per-story; however I do not use numbered entities in my unigram model, as it only generated events based on each singular previous one, and thus would not use enough information for this to be useful when generating new stories - such would only add unnecessary noise to the model.

2.3.4 Generalising nouns and verbs using WordNet

The `generalise` function generalises verbs, nouns and pronouns by replacing them with their WordNet [14] synset. A WordNet synset is a grouping of synonymous words that express the same concept and are thus interchangeable in some context. The result of this is that synonyms are grouped together, thus reducing the size of the vocabulary and focusing the model; and words which failed to generalise are dropped, which increases the grammatical accuracy of the dataset as many erroneously parsed words are dropped. After this step, an event $e = \langle wn(s), wn(v), wn(o), p, wn(m) \rangle$, where the function $wn(x)$ gives the WordNet synset of the argument.

This is done simply, where the synset is disambiguated as to be the most common context of the word as denoted by WordNet (namely its lowest ennumerated synset), regardless of its context within the sentence. I chose to use WordNet rather than VerbNet [18] as used by Martin *et al.* [13] as words successfully generalised more frequently: 94.3% of the time using WordNet (see Figure 2.1), versus 61.7% of the time using VerbNet (see Figure 2.2). Additionally, Martin *et al.* disambiguate the context of a verb with multiple meanings by choosing the VerbNet class with the most synonyms, which offered less accurate results than choosing the most common application of the word (made possible with WordNet ordering of synsets) as many of the most frequent words were not replaced with their most probable use: i.e. *have* is translated to $devour - 39.4 - 1$ instead of $own - 100$. Writing a WordNet semantic context parser like Shi and Mihalcea [20] would be more accurate than either method, but such was beyond the scope of this project.

The `more_vague` parameter is an option to further generalise these synsets by attempting to return a WordNet synset one level up in the hypernym tree: namely the parent synset. A hyponym is a word defining a subset of a set which is defined by its hypernym - in common language, a hypernym is a *blanket term*: i.e. *parent* has this relation to *mother* and *father*. This results in grouping related words, meaning that instances of a word affects occurrences of closely related words, but outputs are more ambiguous and abstract: i.e. $male, female \rightarrow person$.

Figure 2.2: How well do verbs generalise using WordNet?

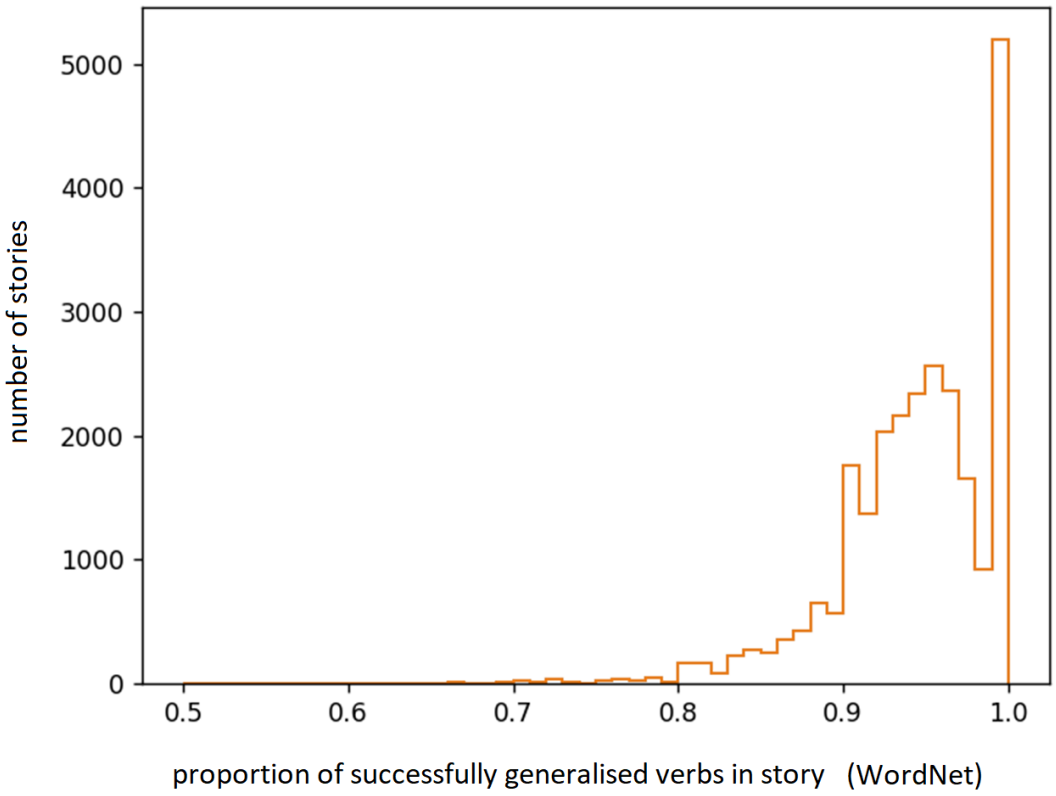
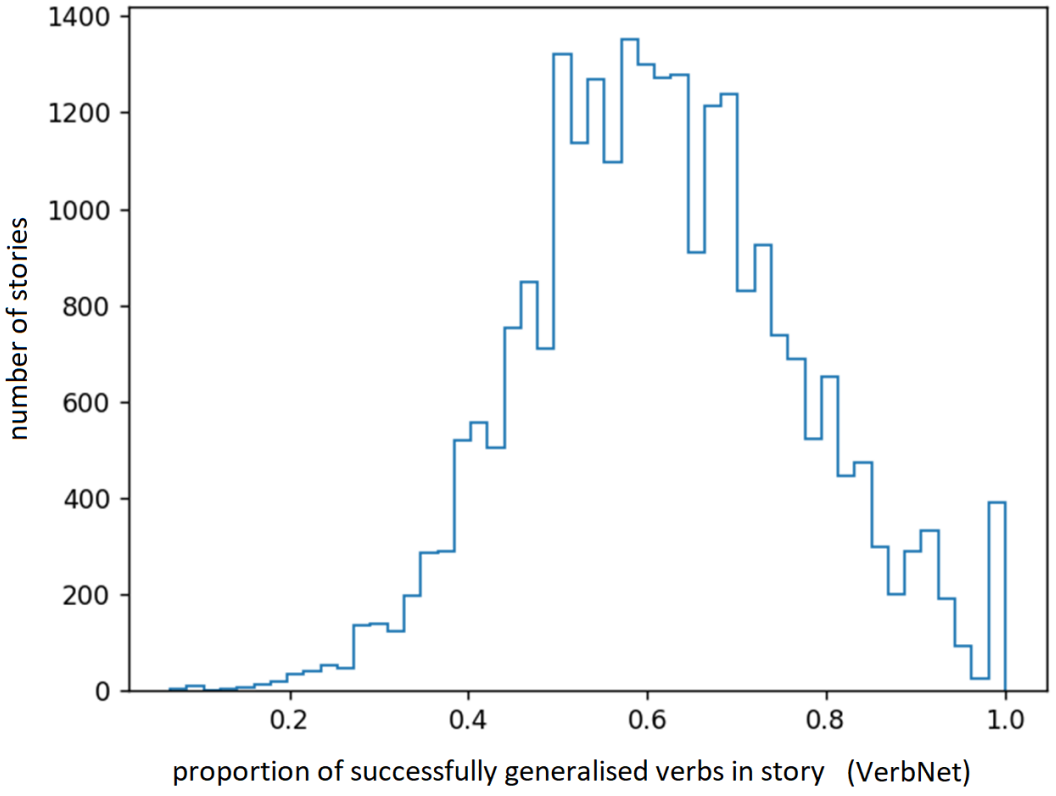


Figure 2.3: How well do verbs generalise using VerbNet?



Chapter 3

Implementation

To run the code in the repository, `process_data.py` is run first to transform the CMU corpus into the training dataset (as detailed in the preparation chapter); then `train_model.py` is run to shape and encode the input data, and to train a model by calling methods from `reinforce.py`; and finally, `predict.py` is run to test the model and evaluate the results.

3.1 Calculating reward values

In the function `calc_rewards` in `reinforce.py`, static rewards are calculated for each verb in the corpus vocabulary. In this implementation, I have chosen the verbs *love*, *marry* and *kiss* (corresponding to WordNet classes *love.v.*01, *marry.v.*01 and *snog.v.*01) as the list of targets verbs v_t, as they represent satisfactory endings to romance stories.

First, reward distances $r_1(v, v_t)$ are calculated based on the distance from every verb v within each story to a given target verb v_t, subtracting the distance to the target from the length of the story to proportionally reward lower distances more. The logarithm of this is taken to damp off higher values. The same formula is used as in equation (2) in Martin *et al.*'s implementation [13]:

$$r_1(v, v_t) = log \sum_{s \in S_{v,v_t}} l_s - d_s(v, v_t) \tag{3.1}$$

where S_{v,v_t} is the subset of stories in the corpus where v occurs before the target verb v_t, l_s is the length of story s, and $d_s(v, v_t)$ is the smallest number of events between v and v_t occurring in story s.

Reward frequencies $r_2(v, v_t)$ are calculated based on how often each verb v in the corpus vocabulary appears anywhere before a given target verb v_t. This encourages the model to choose verbs which often appear before the target verbs. Each reward frequency is scaled against the overall frequency N_v of that word appearing in the corpus, and the logarithm of this is taken to damp off higher values. The same formula is used as in

equation (3) in Martin *et al.*'s implementation [13]:

$$r_2(v, v_t) = log\frac{k_{v,v_t}}{N_v} \tag{3.2}$$

where k_{v,v_t} is the sum of the number of times a verb v appears before the target verb v_t in each story.

The frequency reward multiplier m allows for weighting rewards on the frequency they occur in the corpus. A higher value of its parameter c will reward common words and cut off the tail of very low frequency words, and there will be no effect if $c = 0$. It is calculated as:

$$m(v) = 1 + arcsinh(c * f(v)) \tag{3.3}$$

where $f(v)$ is the frequency of the verb v occurring in the corpus.

Then, the base reward r is calculated for each verb v with a modified version of equation (4) in Martin *et al.*'s implementation [13], with the addition of m:

$$R(v) = m(v) \sum_{g=v_t} r_1(v, g) r_2(v, g) \tag{3.4}$$

Finally, rewards are adjusted so the target verbs v_t have the highest rewards, and then normalised and offset to a -1, 1 range. By offsetting to this range, the model will track a typical good value and punish anything beneath it, not only reward high values. The *update_policy* function tracks an average reward value over time by taking the mean of historical rewards and using that as a 'zero' point to do this, such that good moves offer a positive reward versus bad moves offering a negative reward. Without this normalisation, the model is likely to settle on sub-optimal policies, ending up in a biased optimum.

3.2 Clustering verbs using rewards

In the function `cluster_verbs`, verbs in the corpus vocabulary are clustered based on reward values using the Jenks Natural Breaks (JNB) optimisation technique [7] - a data clustering method designed to determine the best arrangement of values into a given number of classes. Martin *et al.* [13] use this method to both discourage the model from generating a target verb too quickly, and to encourage it to reach a target verb within some number of steps. The output vocabulary of verbs generated by the model is restricted to the set of verbs in the $k + 1$th cluster, where k is the index of the class to which the previous verb input to the model belongs. The higher k is, the higher the average reward of the verbs in that class, with the target verbs belonging to the class with the maximum k. The gradient update in equation (3.6) will thus take greater steps toward verbs that are more likely to occur next, given a known goal.

Replacing the older Jenks-Caspall algorithm [7] used by Martin *et al.* [13], this implementation uses the Fisher-Jenks algorithm [6]; which clusters data by seeking the

best arrangement of values into different classes which minimise each class's average deviation from their class mean, while maximising each class's deviation from the means of all other classes. Unlike the Jenks-Caspall algorithm, the Fisher-Jenks algorithm is guaranteed to produce an optimal classification for a specified number of classes. [6]

The goodness of variance fit (GVF) for a given classification can be calculated as a measure of certainty on how well-classified the data is: this is maximised to determine the optimal number of classes that should be used. GVF ranges from 0 (worst fit) to 1 (perfect fit), and is defined as:

$$GVF = \frac{SSDAM - SSDCM}{SSDAM} \tag{3.5}$$

where SSDCM is the sum of squared deviations from a class mean, and SSDAM is the sum of squared deviations from the overall array mean.

I additionally used the function `merge_clusters` to merge lower value classes produced by the clustering until all classes except the last contained a minimum number of values; because for my reward values, this algorithm tended to produce one or two large clusters and then many tiny clusters containing only one value each, which would otherwise result in generated stories being very monolithic as they would always be limited to the same sequence of verbs for the last few events.

3.3 Encoder-decoder network

A sequence-to-sequence (Seq2Seq) model which uses an Recurrent Neural Network (RNN) based encoder-decoder architecture [22] is trained to predict the next token in a sequence given one or more input tokens, by pairing successive tokens in a corpus and learning a set of weights to capture the relationship between them [22]. Following the methods of Martin *et al.* [13], an encoder-decoder network is used as the base language model, generating a sequence of events as opposed to a sequence of words (as for a standard text-generation application).

To achieve this, an encoder inference Long Short-Term Memory (LSTM) network [5] is used to read an input sequence one event at a time, and encode this by mapping it to a fixed-dimensional vector representation, and then a decoder inference LSTM is used to decode the target output sequence from that vector. LSTMs are chosen to be used for the encoder-decoder network as they are suitable for this type of sequence-to-sequence application with long-range temporal dependencies, which would be found in story plots as multiple plotpoints may be set up early before being executed later. However, in my unigram model, only the previous event is used to generate the next event, as I did not complete the extension task of extending it to an n-gram model. To achieve this, an encoder inference Long Short-Term Memory (LSTM) network [5] is used to read an input sequence one event at a time, and encode this by mapping it to a fixed-dimensional vector representation, and then a decoder inference LSTM is used to decode the target output

sequence from that vector. LSTMs are chosen to be used for the encoder-decoder network as they are suitable for this type of sequence-to-sequence application with long-range temporal dependencies, which would be found in story plots as multiple plotpoints may be set up early before being executed later. However, in my unigram model, only the previous event is used to generate the next event, as I did not complete the extension task of extending it to an n-gram model. To achieve this, an encoder inference Long Short-Term Memory (LSTM) network [5] is used to read an input sequence and encode it by mapping all of the events in the sequence to a fixed-length vector representation, which is the output of the encoder, and then a decoder inference LSTM is used to decode the target output sequence from one event at a time using the fixed-length vector as the starting hidden state. LSTMs are chosen to be used for the encoder-decoder network as they are suitable for this type of sequence-to-sequence application with long-range temporal dependencies, which would be found in story plots as multiple plotpoints may be set up early before being executed later. However, in my unigram model, only the previous event is used to generate the next event, as I did not complete the extension task of extending it to an n-gram model.

The decoder output is connected to 5 dense layers, which each output a different feature of the 5-tuple event: one each for subjects, verbs, objects, prepositions and modifiers.

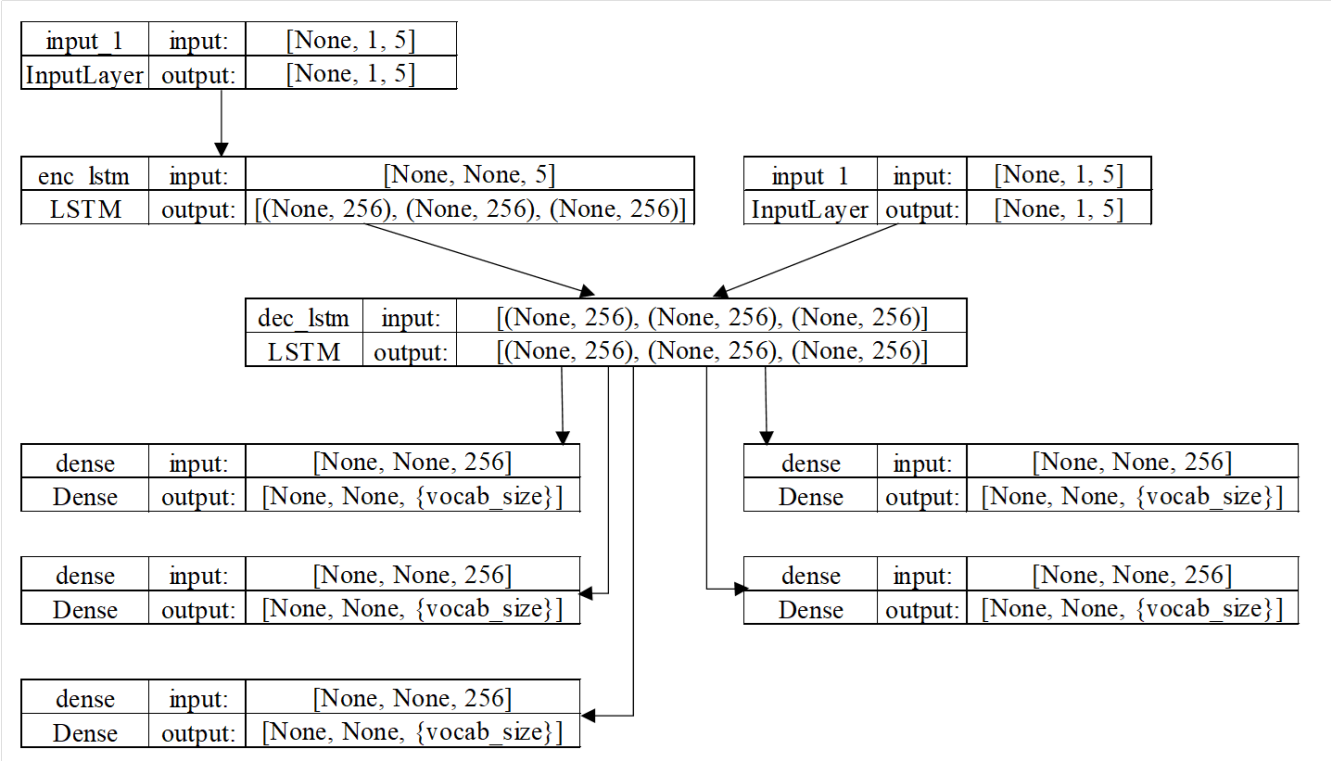
Event sequences in the transformed corpus are encoded and reshaped into a suitable format to be input to the model for training in the `encode_data` function. Events are encoded as integer 5-tuples, with $\langle START \rangle$ and $\langle END \rangle$ tokens added to the ends of each sequence, and sequences are padded to a maximum length. The encoder input is a list of the first events of each story. For training, the decoder input is a list of all event sequences without the first events, and the decoder outputs are event sequences for each individual feature (subjects, verbs, objects, prepositions and modifiers) again without the first events.

This sequence-to-sequence model is compiled with a general purpose Adam (Adaptive Momentum) optimiser [8] and either Keras's sparse categorical implementation of the cross-entropy loss function [4] or a simple custom loss function equal to the sum of the product of advantages and log likelihoods of generated outputs.

3.4 Training the model

The base sequence-to-sequence model is pre-trained, fitting on the training dataset by feeding the observed sequence values into a Teacher Forcing algorithm [9]: this tells the neural network what the 'correct' answer for each generation in the sequence is after it makes a prediction, which speeds up conversion. In reality, the 'correct' answer is not known, and thus in the decode step output events are fed back in one at a time for comparison.

Figure 3.1: Encoder-decoder model diagram



Next, the model is trained again using reinforcement learning, attempting to learn an optimum policy for reaching the goal of seeing a specified target verb at the end of a generated plot (see: Reward-Shaping section). The REINFORCE [24] algorithm generates policies for taking actions: first generate plots (sampled randomly using the probability distribution as opposed to most probable event sequence, to encourage variation), calculate cumulative rewards based on the generated plots, then feed those rewards back into network (backpropagation) and update the network based on how the generated rewards compare to historical rewards, and repeat.

For each event e_i in the corpus used as input, an action consists of choosing a probable next event e_{i+1} by randomly sampling from the probability distribution of the language model $P(e_{i+1}|e_i;\theta)$. The final gradient ($\nabla_\theta J(\theta)$) for updating the parameters (θ) of the network and shifting the distribution of the language model is calculated as:

$$\nabla_\theta J(\theta) = R(v(e_{i+1}), v_t)\nabla_\theta log P(e_{i+1}|e_i;\theta) \tag{3.6}$$

where e_{i+1} is the event chosen at timestep $i+1$, and $R(v(e_{i+1}), v_t)$ is the reward for the verb in that event given target verbs v_t. This is the same as equation (1) used by Martin *et al.*. [13].

The `decode_sequence` function is called to generate an output sequence of events based on each previous event (unigram only) for a given input event fed in, by sampling from the current policy. This function implements the reward-shaping: the output

vocabulary for verbs is limited to verbs from the cluster $k + 1$ following from the previous verb's cluster k (unless k is already the highest-valued cluster, in which case it continues to sample from only cluster k). We define the maximum length of a generated story to be 15 here to prevent infinite generation if a target verb is not reached.

The sample probabilities for the output vocabulary may additionally be scaled by a function of word frequency (how often a given word appears in the training corpus) m_2 based on a constant scaling factor c_2. A higher value of the constant c_2 will favour generating words which occur more commonly in the corpus and cut off the tail of very low frequency words, and there will be no effect if $c = 0$. This is similar to the frequency scaling option for rewards, but is implemented in a different place and potentially different constant, with a similar equation:

$$m_2(v) = 1 + arcsinh(c_2 * f(v)) \tag{3.7}$$

where $f(v)$ is the frequency of the verb v occurring in the corpus.

The function `update_policy` implements the policy gradient descent, backpropagating any reward received when attempting to generate new events with the current policy through the weights of the policy model, and updating the policy based on reward history.

The loss function used in this step is:

$$\texttt{loss} = L_{CE}(e_{i+1}, P(e_{i+1}|e_i; \theta)) * A \tag{3.8}$$

where e is the action observed, P is the action probability, and A is the advantage, and L_{CE} is the cross-entropy loss function:

$$L_{CE} = -\sum_{i=1}^{N} t_i log(p_i) \tag{3.9}$$

for N classes (which are words in the vocabulary for this model), where t_i is the truth label and p_i is the Softmax probability for the ith class. [4]

Cross-entropy loss takes a set of generated output probabilities and measures the distance from the truth values for the original training data, with a logarithmic penalty based on how far the model output is from the actual expected value [4]. The cross-entropy loss is scaled by using A as the sample weight parameter when fitting the neural network, which effectively weights the loss up or down as if the network were seeing A many samples). This results in the model learning the policy: if $A > 0$, it has the effect of increasing the loss for the action observed (i.e. the action taken), which causes the action probability to increase as the neural network strives to correct the loss; and if $A < 0$ then it conversely discourages the move.

3.5 Generating new plots

The `decode_sequence` function is called by the `predict` function in `predict.py` for generating new plots, with the input being a single event to act as the first event of the plot. This outputs a sequence of new events that follow from that first event. If the verb or object (which may take the form of a complimentary verb) in any event output by the model is the one of the target verbs within the maximum number of events (15), then the call that generated it is considered successful.

The user has to manually translate results back to a human-understandable format. This is done by selecting from the potential meanings of each generated synset, and then stitching together the constituent parts of one or more events into a sentence based on the lexical roles (subject, verb, object...) of each word in an event.

This translation is a time-consuming task and requires understanding of this event format to do so, and thus for the user study, I did the translations myself so participants could understand the outputs with little effort. To prevent the worst case behaviour for the user study of producing offensive or otherwise upsetting results by reinforcing potential biases in the training set, I manually vetted results that would be shown to participants.

3.6 Extensions

I set more specific and achievable extensions than the ones initially detailed in my project proposal, and was able to complete some of them within time constraints.

Extensions which I completed include:

- I extended the event format defined by Martin *et al.* [13] to include prepositions.
- Compared to the VerbNet implementation of Martin *et al.* [13], I improved the generalisation method for replacing verbs with suitable synsets, using WordNet instead[1].
- I modified the reward function to accept and combine rewards from multiple target verbs.
- I improved the verb-clustering function, which in its pure form would categorise verbs with an excessively light tail: the majority of the clusters would only contain one or a very small number of verbs, which would result in overly similar outputs as only a single verb can be chosen from a cluster and this continues over a sequence of clusters. I achieved this with my *merge_clusters* function[2].

[1] This is detailed previously in the Preparation chapter.
[2] This is detailed previously in the Implementation chapter.

- I designed and implemented a hyperparameter for scaling rewards and/or output probabilities based on the frequency with which words occur in the corpus. A higher value of the hyperparameter will reward common words and cut off the tail of very low frequency words, and there will be no effect if equal to zero. This is implemented in two ways: as separate frequency scaling constants for rewards[3] (c) or output probabilities[4] (c_2).

Extensions which I came up with but did not complete include:

- Write a WordNet semantic context parser such as that created by Shi and Mihalcea [20]. This would be more accurate than my own context-choosing method[5], but such was beyond the scope of this project as it would have required significant work in itself.

- Add support for coreference resolution to the generalisation of named entities. A coreference occurs when two or more expressions refer to the same entity in a text. Coreference resolution systems try to look for mentions that corefer to the same entity and group them into clusters. I attempted to do this using CoreNLP [11] tools, but did not manage to get this working.

- Extend the basic unigram model, in which only the previous event is used to generate the next event, to an n-gram model. Martin *et al.* [13] do this in the paper which this project replicates the methods of.

- Identify a suitable alternative NLP task which the the model results could be applied to, and use this for extrinsic evaluation[6].

[3]See equation 3.3 for details.
[4]See equation 3.7 for details.
[5]This is detailed previously in the Preparation chapter.
[6]This is defined and explained in the Evaluation chapter.

Chapter 4

Evaluation

Intrinsic evaluation involves finding some metric to evaluate the language model itself, without focusing on the specific tasks the model is designed to be used for [2]. The quality of generated text is judged against some predefined ground truth: which in this case, constitutes using the corpus as reference. However, it can be difficult to find an acceptable ground truth, and I cannot easily quantify how well movie summaries function as a ground truth for story plot descriptions aimed at inspiring writers - as such, I have made the assumption that these are acceptable for this purpose.

Extrinsic evaluation involves evaluating a model based on based on the impact of its outputs on the performance of other NLP systems, such as translation benchmarks [2]. However, this requires high computational resources to train and test the models, and additionally, I would have to come up with a plausible application this system could be transformed to complete. I considered this to be an extension as it would require significantly more effort, and in the end I did not have time for this.

My evaluation strategy is to first compare the performance of three different models trained with different parameters using a selection of intrinsic evaluation metrics, then judge how well a selected 'best-performing' model is considered to actually perform by human writers through the results of the participant questionnaire. I have chosen this strategy because this is a system designed to be used by humans and there is no real quantitative 'correct answer' to what is a good plot or an inspiring prompt.

4.1 Intrinsic Evaluation

In this section, I will evaluate three different models (*unclustered*, *dense_clusters*, and *small_clusters*), all of which were trained on a subset of 200 stories, with 32 epochs of pre-training and 10 epochs of reinforcement learning. The *unclustered* model does not involve clusters in the prediction step, instead always sampling verbs from the full distribution. The *dense_clusters* model utilises JNB clustering of verbs with 30 breaks, with clusters being merged until they reach a minimum of 20 verbs per cluster (except for the last cluster which may be smaller). The *small_clusters* model utilises JNB clustering

with 20 breaks, only merging to a minimum of 10 words per cluster.

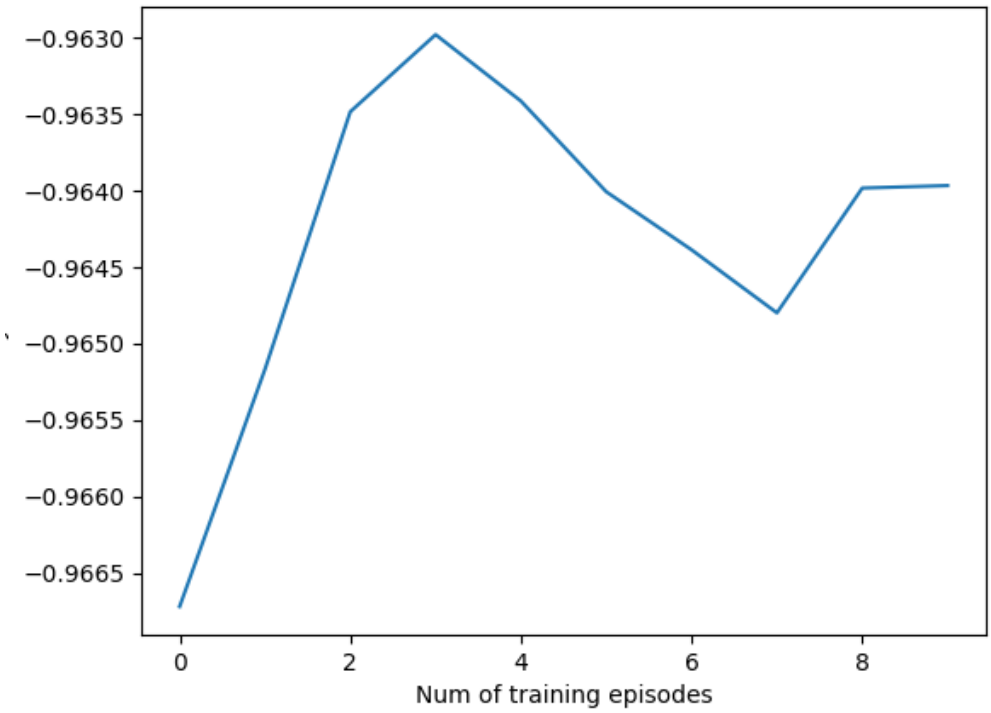
Figure 4.1: Distribution of verbs across clusters in *dense_clusters* model

cluster number	0	1	2	3	4
number of words	356	22	21	21	7
mean reward	-1	-0.9995	-0.9925	-0.8574	0.4369

Figure 4.2: Distribution of verbs across clusters in *small_clusters* model

cluster number	0	1	2	3	4	5	6	7
number of words	356	15	11	10	11	10	10	4
mean reward	-1	-0.9997	-0.9987	-0.9951	-0.974	-0.9028	-0.0169	0.6872

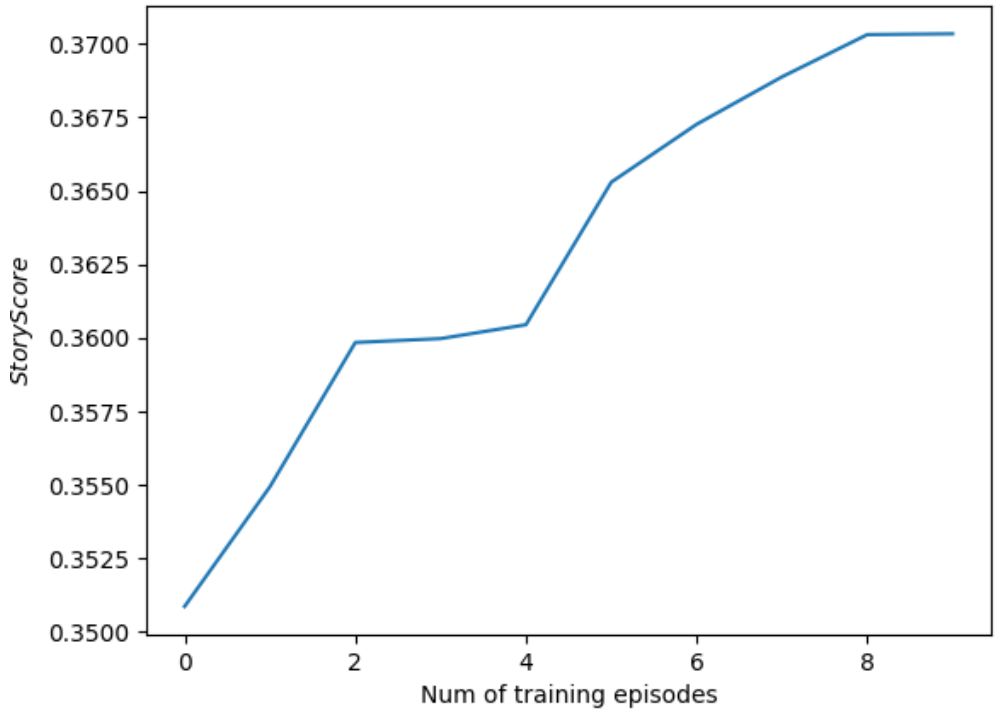
Figure 4.3: Average rewards of output over training episodes for *unclustered* model



Figures 4.3, 4.4 and 4.5 depict training rewards over time for each model during the reinforcement learning step.

For the *unclustered* model, the training rewards experience a sharp increase followed by a small decrease which then levels out. This may correspond to overfitting on the training data then correcting that as more rewards are provided, however the rewards only vary over a very small range (of magnitude around 0.0035), and

Figure 4.4: Average rewards of output over training episodes for *dense_clusters* model



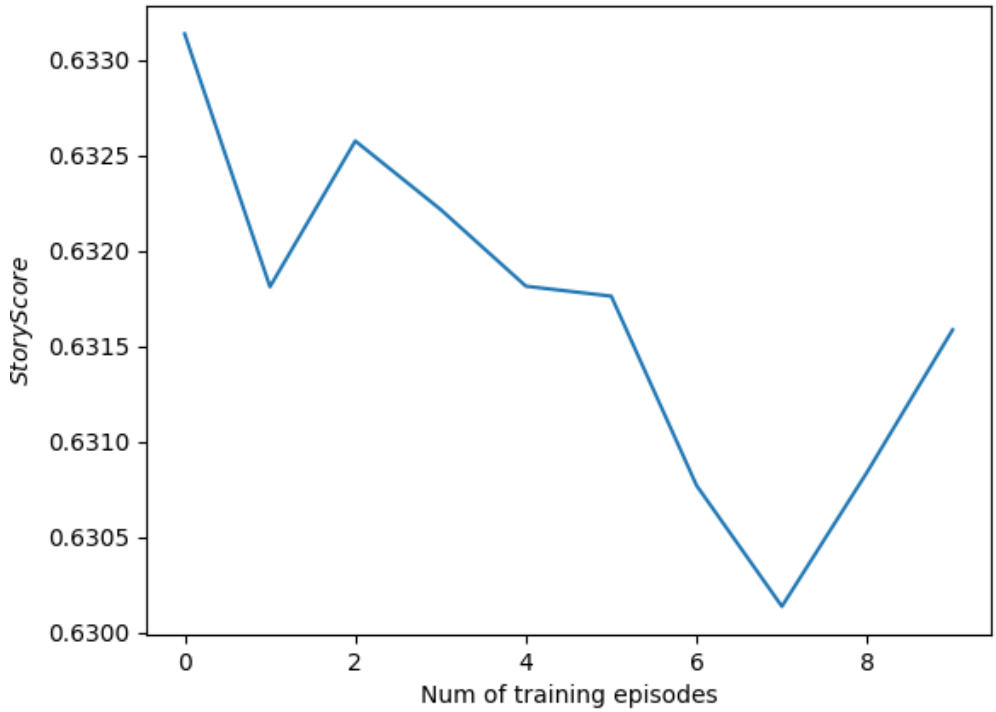
it may be unreliable to pass judgement on the implications of any variation within such a small range. The training rewards are consistently very low for the *unclustered* model, with the minimum possible value being -1 and the maximum possible value being 1.

For the *dense_clusters* model, the training rewards experience a constant increase, implying steady improvement. The values of the training rewards are considerably higher than for the *unclustered* model, suggesting that this model is learning better how to reach its goals. For this model, the range of values for the training rewards experience the most change of any of the three models, but this is still only a very small variation (of magnitude around 0.02).

For the *small_clusters* model, the training rewards decrease at first but either level out or start to increase again. This could correspond to overfitting on the training data then correcting that as more rewards are provided. Training rewards are averagely much higher for this model than either of the other two, implying that it is most capable of reaching its goals as directed by the rewards.

Ideally, I would look at these graphs to identify whether an optimal policy was reached, marked by a visible jump in average values of training rewards, however: the number of training episodes is far too small to reliably judge or compare any long-term trend; and the range of average training rewards is very small for all of the models, marking very little change, implying that the reinforcement learning step may have been

Figure 4.5: Average rewards of output over training episodes for *small_clusters* model



ineffective in the first place.

Figure 4.6: Table of statistics for model outputs

	model		
	unclustered	*dense_clusters*	*small_clusters*
Target hit rate	5.26%	13.16%	18.75%
Average output story length	14.132	14.079	13.1875
Average output story length *(hits only)*	6	11	13.3
Average output perplexity	1.093	1.119	1.177
Average output perplexity *(hits only)*	1.095	1.122	1.216

The target hit rate is defined as how often a model, within the limit of some given maximum number of generations (which is taken as 15 here), eventually outputs a target verb and terminates a story successfully. The *unclustered* model had the lowest target hit rate, as was to be expected since the verb clustering was designed to focus the model output towards its goal. However, this is notably lower than the 19.935% hit rate (averaged over both reward verbs) achieved by Martin *et al.*'s corresponding *unrestricted* model [13]. The *small_clusters* model had the highest target hit rate, however, both clustered models had significantly lower hit rates in comparison to the massive 93.82% achieved by Martin *et al.*'s corresponding *clustered* model [13]. This shows that while

the clustering method is useful for directing the plot towards a pre-specified ending, in order for frequent success to be achieved, more work likely needs to be done in tuning the parameters of the model and training the model over more epochs - due to constraints on resources and time, all three models evaluated here were trained on a small subset of training data and over a very low number of epochs, with minimal hyperparameter tuning, unlike in Martin *et al.*'s paper.

The maximum length of any generated story was 15 events for all three models, as generated stories were terminated at that length to avoid continuing infinitely with no guarantee of ever reaching a goal. The average length of generated story where the goal was reached was the highest for the *small_clusters* model - this was the expected result as the model had to navigate through more intermediate clusters before it could possibly reach a target verb only in the final cluster. The *unclustered* model was more likely to quickly jump to the goal in the few instances it did output a target verb - however, this comparison could be biased as only a few of test generations achieved the goal with the *unclustered* model and thus this statistic is derived from an insignificant number of datapoints.

Surprisal, also called self-information or information content, is a measurement of the predictability of an event in context: how surprising its occurrence is. In this application, an event is a word being chosen as the next word in a sequence, and we specifically look at verb sequences. If a word is has a 100% probability of occurring in a given context, then it conveys no new information and has a surprisal value of 0, with such being the minimum value. A very unpredictable word for a given context has a very low probability of occurrence, thus having a high surprisal value, with a maximum limit of infinity. [10] Surprisal can also be used as a measure of how difficult a sentence is to understand for humans, through considering word-to-word predictability.

The surprisal of a word w is calculated as:

$$\texttt{surprisal} = -log(p(w)) \tag{4.1}$$

where w is a word, and $p(w)$ is the probability of seeing that word within its given context.

Entropy is a measurement of how much information is conveyed on average for each word in a text [19], considering each word choice as a random trial where words are chosen by sampling from a distribution. The entropy of a story W (where W is a sequence of words w) is calculated as:

$$\texttt{entropy}(W) = \sum_{w} p(w) * -log(p(w)) \tag{4.2}$$

which is equal to:

$$\texttt{entropy}(W) = \sum_{w} p(w) * \texttt{surprisal}(W)) \tag{4.3}$$

Perplexity is a measurement of how well a probability distribution predicts a sample. In this application, it is the inverse probability of a generated story. A low perplexity is an

indicator that the probability distribution the text is sampled from is a good model for the distribution of the training data: a minimal perplexity means a maximal probability. Low perplexity values are preferred over high perplexity values since predictable results are preferred over randomness: lower entropy corresponds to less randomness, and the perplexity is the exponentiation of the entropy. It is a useful metric for comparing different probability models for the same data. The perplexity of a story W is calculated as:

$$\texttt{perplexity}(W) = \frac{1}{P(w_1, w_2, ...w_N)^{\frac{1}{N}}} \tag{4.4}$$

which can be rearranged as:

$$\texttt{perplexity}(W) = 2^{\sum_w p(w) * -log(p(w))} \tag{4.5}$$

which is equal to:

$$\texttt{perplexity}(W) = 2^{\texttt{entropy}(W)} \tag{4.6}$$

The average perplexity for stories where the goal of reaching a target verb was not achieved was higher than that for stories where the goal was successfully achieved, for all three models. This result defies expectations, which were for stories which stayed on track and moved towards a goal to have lower perplexity values as adjacent verbs would be more closely related. The values are very close, but nonetheless lower. Additionally, the average perplexity values were lower for the *unclustered* and *dense_clusters* models which performed less preferably than the *small_clusters* model in most other aspects, but these values were still very close. The *unclustered* model having a lower average output perplexity than the clustered models could be attributed to the fact that rewards are based on subsequent verbs in the corpus instead of being restricted to clusters, resulting in that model better replicating the distribution of the training data as it doesn't encourage or enforce these specific outcomes. Overall, however, all of these values for average perplexity of generated stories are too close to reliably compare.

It should be noted that all perplexity values presented in Figure 4.6 are concerningly low. The maximum value for perplexity is infinity, corresponding to a zero-valued probability of occurring, and the minimum value is 1, corresponding to absolute certainty. Low perplexity can be a result of overfitting the training data: this is the most realistic expectation for the low values for all 3 models, due to the small amount of training data used and the low number of epochs all of the models were trained over.

Other than in regards to perplexity, which may be discounted as a metric here due to the extremely low values, the *small_clusters* model is consistently shown to be the most preferable model, and thus it is used to generate stories for the human study.

4.2 Human Study

15 fiction writing enthusiasts were recruited for a participant study, with the goal of collecting qualitative feedback about how useful the intended users of the system (writers)

found the model results. Participants were presented with a selection of 6 different stories output by the best performing model (*small_clusters*) from simple prompts, translated into an easily readable prose format. The stories were manually filtered for any sexual or otherwise upsetting content, as requested by the Ethics Committee. In a Google Forms questionnaire, the participants were asked about attributes of these stories: for each story, they had to give a rating on a 0 - 10 scale of a selection of attributes. A high rating (10) would imply that the generated story strongly held this attribute, and a low rating (0) would imply that the attribute did not hold for the story at all.

These attributes were chosen to be: 'Likability', 'Total Utility', 'Partial Utility', 'Plausibility', 'Local Causality', 'Elaboration', 'Novelty', 'Non-Repetition', 'Cohesiveness', 'Grammaticality', 'Commonsensicality' and 'Genre-Relevance'. I chose specific attributes which I considered to each cover different aspects of what a 'useful writing prompt' might entail, and which could be described in a relatively non-vague definition - as opposed to 'Creativity', for example, which would be considered too subjective to define. In a pilot study for this questionnaire, I received feedback that participants were taking longer to comprehend the few attributes with negative polarity - i.e. repetition, where a high score would be a negative outcome - so I chose only attributes with positive polarity to avoid any related confusion and allow users to complete the long questionnaire faster. In order to disambiguate the meaning behind each attribute and minimise such ambiguity creating noise in participants' answers, the participants were given a glossary defining each attribute, and explaining what a 0 or 10 rating would imply for that attribute specifically, as follows:

Likability: The story appeals to you personally.
0: I either actively dislike or am entirely disinterested in the contents of this story.
10: I really like the contents of this story. I find it interesting, inspirational or enjoyable.

Total Utility: The story in its entirety is usable for deriving a more detailed plot.
0: It would be impossible to use all or most of the story to write a plot that makes sense. All sentences of the story are useless to me or anyone else.
10: I could easily use all or most of the story to write a plot that makes sense.

Partial Utility: At least some event within the story is usable for coming up with a more detailed plot.
0: I could not use any sentence of the story to write a plot that makes sense.
10: I could easily use at least some part of the story to write a plot that makes sense.

Plausibility: How reasonable or probable the events of the story are given the initial state and the ending.
0: The events within the story are unrelated to or actively work against transforming

the first sentence to the last one.
10: Events within the story plausibly transform the first sentence to the last one.

Local Causality: The contribution of any state or event within the story to the occurrence of any other later state or event. Linkage between pairs of successive sentences.
0: No circumstance, theme, character or happening are linked within or between sentences.
10: All or most circumstances, themes, characters and happenings are linked between sentences, and there is an understandable 'chain of events'.

Elaboration: The level of detail in which ideas are presented.
0: The events of the story are simple and sparse. It may be ambiguous what is happening because they are too simple.
10: There is a beyond-basic level of detail about events that happen in the story.

Novelty: The quality of being new, original, or unusual.
0: The events of the story are unexciting or overused in media, or the sentences don't conveying any information beyond mundane activities.
10: The events of the story are wacky or original.

Non-Repetition: Each story avoids repetition of events or earlier information within itself. Each of the stories are also significantly different from each other.
0: Earlier events or information are constantly repeated.
10: Every sentence in the story only contains new unseen information from earlier sentences.

Cohesiveness: The story has the properties 'entailment' (events are sequenced so they logically follow) and 'progression' (the state of the world changes through sentences)
0: The text is just a series of words/sentences with no connections between them.
10: The text tells a story of events that happen which transform a situation.

Grammaticality: Syntactic correctness.
0: Verbs, nouns and prepositions are constantly used in the wrong contexts and in place of each other.
10: Every sentence uses correct grammar.

Commonsensicality: Ideas can be perceived, understood, and judged easily.
0: Ideas in individual sentences are conceptually or physically nonsensical.
10: Ideas in individual sentences are conveyed regardless of grammar and have easily comprehensible meaning behind them.

Genre-Relevance: How connected the words, events and ideas are to themes prevalent in the romance genre.
0: The story could not be described as having any relation to the romance genre.
10: The story contains words and concepts relevant to the romance genre.

The generated stories were translated to simple prose from the abstract event format by myself, and I attempted to match the grammar of the translations to that of the output events. The stories were then presented to participants in the format of: 'Prompt... *Completion.*'

The specific stories were as follows:

1. A man works at a restaurant... *applying fragrances between customers' cuff-links. His ideas transform the local hospital between gossip. One day, his family member is hosted after an occultist, and the army judges that they must go bowling abroad a bomb. The bomber establishes a group until he finds himself yearning: the french fries improve this building in its features. His name is not clean, but his demand are to his apron. The man lies about his possession of a business, but the questioning ends when he registers his ownership of it through inheritance. A lack of concern kisses those who follow in the investor's footsteps.*

2. When Janet leaves her job... *this moment stumbles into the pursuit of a Casanova. Janet passes up her paramour with a request: that his money is to be involved in her property, despite his hypnotising charm. He repeats his hypnosis skills upon her chest, requesting that she covers him through the valley. The delusion of marriage throws them into failure.*

3. A woman is staying at the palace... *when a truck drives around her balcony. A quotation is being smuggled in by a race. A ladybird brings angst in the form of a bet. This break drives her to pray for anything except marriage. An informant harasses the male traveller, and his investigation corrects a shirt alongside some fish eggs. This defeat resembles a serenade as it follows beer. The treatment is completed by marrying as the throne.*

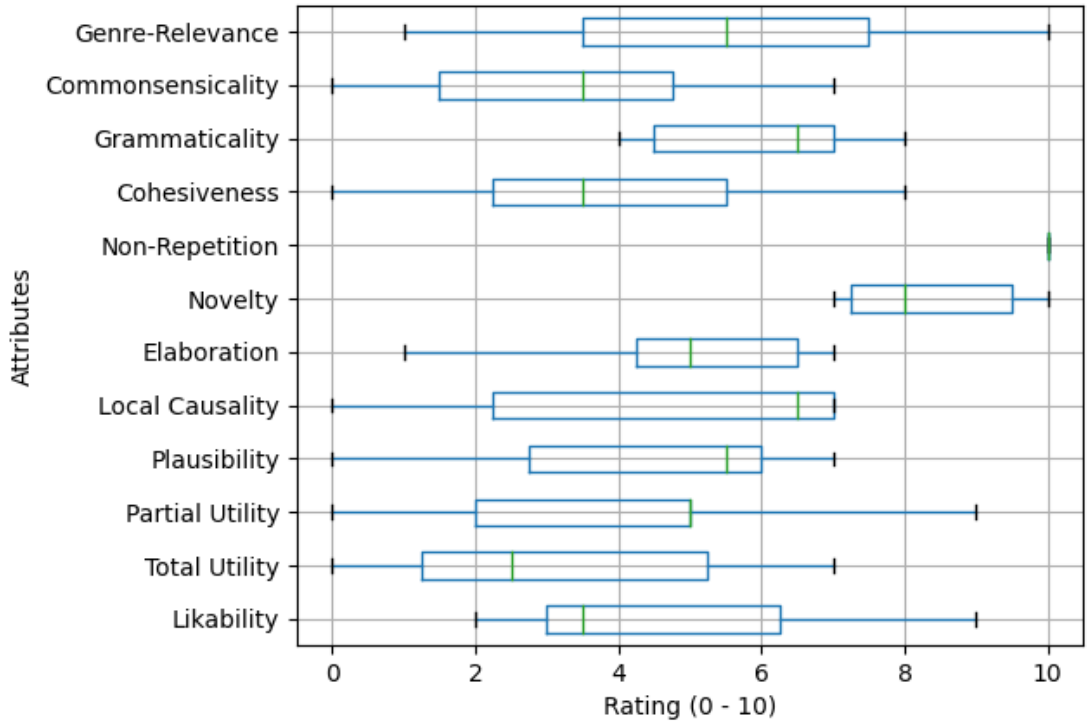
4. Sandra has a baby... *That information chokes and discourages a while ago a pawn. A fried chicken enriches Sandra's creativity through the suspect. This conclusion highlights the land of righteousness. A man marries an airplane by telephone.*

5. A woman boasts about her business... *Her old schoolfriend names abandonment towards this leak. Her husband engages in separated rendezvous in accordance with his manners. His request is that she confess as secret. The dub flirts a bank cheque into new beginnings. An offer breaks their kiss due to the wreckage of a boat.*

6. A girl takes care of a baby... *Her grandparents try to hold off the inevitable divide between them despite the pouring rain. A gymnastic horse believed to be tired of*

opening. The Lord recognises the lake next to their misfortune. The guard tries to marry through this god.

Figure 4.7: Participants rated a selection of stories generated by the *small_clusters* model by a selection of attributes, on a scale from 0 to 10



The results of the questionnaire are graphed in Figure 4.4.

The stories were rated as being highly novel and as having no repetition of information within a single story. Combined with a low commonsensicality, meaning that ideas were often conceptually or physically nonsensical, these ratings were likely affected by the output featuring more obscure words at frequencies much higher than expected and in unexpected contexts, with common or expected words other than reward verbs not appearing often. Some repetition of common words in similar roles may be expected in similar systems, however, every story was rated 10 for non-repetition by every participant. A suggested cause of such skewed word frequencies in the output would be the model overfitting to the training data, in which outlier word frequencies could be inflated due to mistakes persisting through singular stories, for example - such as generalising the name of a girl '*Olive*' to '*olive.n.*01' instead of '$\langle PERSON \rangle$'. This is likely given the small size of the sampled dataset that the model was trained on (due to constraints in time and resources). This could be amended by prioritising the highest frequency words even more steeply and punishing the lowest frequency words more, through using a higher

value for the frequency scaling constants for rewards[1] (c) or output probabilities[2] (c_2).

The commonsensicality was also likely worsened by the grammaticality being less than perfect, albeit tolerable to some extent: with some verbs and nouns being used in incorrect contexts, the meaning of the sentence can become even harder to interpret and verge into nonsense.

Although the stories in the questionnaire are commonly rated higher in grammaticality than other attributes, this is a score which would have to be closer to 10 than the other attributes in order for a story to be useful, as it is directly linked to readabilty. Additionally, grammaticality would be an important factor to consider if I were planning to take this project a step further and automate the translation of these event-format output stories back to formal prose.

Cohesiveness ratings skewed towards the lower end, even including zero values, implying that some texts were considered to have no connections between clauses or sentences and thus individual events in those stories were often irrelevant with respects to each other. Related to this idea is plausibility, which was also averagely rated low, meaning that events in the story often did not plausibly transform the first sentence to the last one, and as such the prompt, middle and ending were often unrelated. An important factor to consider here is the fact that a unigram model was used: initial state was not considered again after it had been transformed, and if an irrelevant event was generated then the important state information would be lost from thereon as the model only generates from the single most recent event. This would result in generated stories commonly losing coherence as each story progresses. Stories averagely had a higher rating for local causality than plausibility, thus implying that adjacent events were at least related sometimes, as the unigram model would encourage. Potential improvements in regards to the relevance issue discussed here would be to train the model for more epochs (especially as it was only done over 10 epochs for this model) so it will learn how better to maximise the reward function and thus start generating more words that are relevant to reaching its goals, to train on a larger corpus, and to extend it to an n-gram model.

Reinforcement learning over more epochs could also improve genre-relevance: with the reward-shaping, the direction towards using specific target verbs ensured some relevance, however the relative obscurity of a disproportional number of words in the output (such as $honky\_tonk.n.01$, $vaulting_h orse.n.01$ and $thorax.n.02$) which had no close semantic relation to either the romance genre or the rest of the words in the previous or same sentence, resulted in a lot of individual sentences which each individually were non-relevant to the romance genre. Stories with fewer sentences had less chance to go off-topic in the middle as such, as the final event and occasionally the initial event were guaranteed to contain genre-relevant vocabulary.

[1] See equation 3.3 for details.
[2] See equation 3.7 for details.

Likability is the most conceivably subjective measure used in this questionnaire, as it's based on one person's unique taste of what they alone find inspiring or interesting, and a wide range of very different values is to be expected from such. Despite a low-to-mid-range average likability rating, stories were assigned high ratings occasionally, to say that the rater found the generated story 'interesting, inspirational or enjoyable'. I would judge this to be strongly indicative of usefulness for the purpose they're designed to fill, since I am specifically considering the use of this system as being to generate prompt continuations for writing inspiration. When using writing prompt-fills, a lot of offered ideas which don't appeal to the user may be discarded before finding one which does - and it's the 'finding one [which appeals to the user]' that marks success in that situation, not whether it was the first or thirtieth idea which managed to incite inspiration.

Chapter 5

Conclusions

Based on intrinsic evaluation considering target hit rate, average output story length for stories which successfully reached the goal, and the observed average reward values during training; *small_clusters* was the best-performing model, and then *dense_clusters*, and *unclustered* was the worst-performing model. However, the model still had significant issues and ideally needed to be trained over a much larger number of episodes, on a larger corpus than the small subset used.

Reinforcement learning is less efficient in large action spaces such as the corpus vocabulary, as individual actions have to be sampled in order to assess their quality [25]. Considering constraints on resources and this inefficiency, I was unable to train my models until an optimal policy is reached, instead settling for a computationally manageable number of epochs. As such, I was unable to judge from the average training rewards graphs whether the models were actually coming up with policies for better rewards, since there were not enough datapoints (epochs) to judge from.

With the results of the participant study study, I concluded that this model was more useful than not for its intended purpose of generating prompt continuations for writing inspiration despite the issues noted. However, it should be noted that this system has very little real world usability in its base form, where the output is in the abstract event format and not translated to be human-readable, which currently has to be done manually.

I chose to favour the statistics from human evaluation over mathematical evaluation when judging the effectiveness of the model because it's a system designed to be used by humans, and one which does not have much non-subjective correctness beyond grammaticality (and state consistency based on common sense, which is significantly more difficult to evaluate and can very simply be negated as negations and adjectives are not included in the event format).

Another point to note is that although WordNet [14] enabled significantly more verbs to be generalised than VerbNet [18], this may not have necessarily been a improvement: this allowed more obscure verbs to be picked up by the model, making it more

broadly usable as the limited vocabulary is larger, however a model which could only use more common verbs might have better performance than the model I ended up using. Regardless of that, however, training over more episodes (preferably hundreds rather than just 10) and with a larger dataset should improve that issue.

Bibliography

[1] David Bamman, Brendan O'Connor, and Noah A. Smith. Learning latent personas of film characters. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 352–361, Sofia, Bulgaria, August 2013. Association for Computational Linguistics.

[2] Andrew G. Clark, Chris Fox, and Shalom Lappin. The handbook of computational linguistics and natural language processing. 2010.

[3] Jenny Rose Finkel, Trond Grenager, and Christopher Manning. Incorporating non-local information into information extraction systems by Gibbs sampling. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL'05)*, pages 363–370, Ann Arbor, Michigan, June 2005. Association for Computational Linguistics.

[4] I. J. Good. Rational decisions. *Journal of the Royal Statistical Society. Series B (Methodological)*, 14(1):107–114, 1952.

[5] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Computation*, 9(8):1735–1780, 1997.

[6] George F. Jenks. Optimal data classification for choropleth maps. *Department of Geographiy, University of Kansas Occasional Paper*, 1977.

[7] George F. Jenks and Fred C. Caspall. Error on choroplethic maps: Definition, measurement, reduction. *Annals of the Association of American Geographers*, 61(2):217–244, 1971.

[8] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980, 2014.

[9] Alex M. Lamb, Anirudh Goyal, Ying Zhang, Saizheng Zhang, Aaron C. Courville, and Yoshua Bengio. Professor forcing: A new algorithm for training recurrent networks. In D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 29. Curran Associates, Inc., 2016.

[10] Matthew Lowder, Wonil Choi, Fernanda Ferreira, and John Henderson. Lexical predictability during natural reading: Effects of surprisal and entropy reduction. June 2017.

[11] Christopher D. Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J. Bethard, and David McClosky. The Stanford CoreNLP natural language processing toolkit. In *Association for Computational Linguistics (ACL) System Demonstrations*, pages 55–60, 2014.

[12] Lara J. Martin, Prithviraj Ammanabrolu, Xinyu Wang, William Hancock, Shruti Singh, Brent Harrison, and Mark O. Riedl. Event representations for automated story generation with deep neural nets. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence and Thirtieth Innovative Applications of Artificial Intelligence Conference and Eighth AAAI Symposium on Educational Advances in Artificial Intelligence*, AAAI'18/IAAI'18/EAAI'18. AAAI Press, 2018.

[13] Lara J. Martin, Pradyumna Tambwekar, Murtaza Dhuliawala, Animesh Mehta, Brent Harrison, and Mark O. Riedl. Controllable neural story plot generation via reward shaping. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence*. International Joint Conferences on Artificial Intelligence Organization, August 2019.

[14] George A. Miller. WordNet: A lexical database for English. In *Human Language Technology: Proceedings of a Workshop held at Plainsboro, New Jersey, March 8-11, 1994*, 1994.

[15] Andrew Y. Ng, Daishi Harada, and Stuart J. Russell. Policy invariance under reward transformations: Theory and application to reward shaping. In *Proceedings of the Sixteenth International Conference on Machine Learning*, ICML '99, page 278–287, San Francisco, CA, USA, 1999. Morgan Kaufmann Publishers Inc.

[16] OpenAI. Introducing ChatGPT, 2022.

[17] Jan Peters. Policy gradient methods. *Scholarpedia*, 5:3698, January 2010.

[18] Karin Schuler. Verbnet: A broad-coverage, comprehensive verb lexicon. January 2005.

[19] C. E. Shannon. Prediction and entropy of printed english. *Bell System Technical Journal*, 30(1):50–64, 1951.

[20] Lei Shi and Rada Mihalcea. Putting pieces together: Combining framenet, verbnet and wordnet for robust semantic parsing. In Alexander Gelbukh, editor, *Computational Linguistics and Intelligent Text Processing*, pages 100–111, Berlin, Heidelberg, 2005. Springer Berlin Heidelberg.

[21] David Silver, Guy Lever, Nicolas Heess, Thomas Degris, Daan Wierstra, and Martin Riedmiller. Deterministic policy gradient algorithms. In *Proceedings of the 31st International Conference on International Conference on Machine Learning - Volume 32*, ICML'14, page I–387–I–395. JMLR.org, 2014.

[22] Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. Sequence to sequence learning with neural networks. *CoRR*, abs/1409.3215, 2014.

[23] Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. A Bradford Book, Cambridge, MA, USA, 2018.

[24] Ronald J. Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Mach. Learn.*, 8(3–4):229–256, May 1992.

[25] Asaf Yehudai, Leshem Choshen, Lior Fox, and Omri Abend. Reinforcement learning with large action spaces for neural machine translation. In *Proceedings of the 29th International Conference on Computational Linguistics*, pages 4544–4556, Gyeongju, Republic of Korea, October 2022. International Committee on Computational Linguistics.

Appendix A

Project Proposal

Computer Science Tripos – Part II – Project Proposal

Story-Planning NLG: Generating Story Plots with Reinforcement Learning

16 October 2022

Project Supervisor: Russell Moore, Hope McGovern
Director of Studies: Luana Bulat
Project Overseers: Ferenc Huszar, Andreas Vlachos

Introduction

Inspired by the current common uses of natural language generation AI for predictive writing to fulfil prompts and write stories, and my own use of limited 'writing prompts' resources when planning a novel, I have chosen to train a neural network to generate story plots.

Novel prose requires authorial skill in writing beautifully in addition to fluently and coherently, constant recall of consistent information established earlier in text, and overall refinement of a likely very long body of text. This task itself is difficult to automate with predictive writing due to its innate drawbacks existing in contrast to the latter two goals: if you let a model create long continuations of any prompt (even a good one), it's going to get less coherent and detail over time due to accumulation of inconsistencies as it contributes to its own prompt. It is difficult to direct and significant human corrections become more and more necessary, which professional users may find frustrating and time-consuming. This project will thus not be about utilising AI to write the novel itself, but rather to assist a human creator in designing it.

The task of planning fiction stories can be abstracted into three main subtasks: designing characters, worldbuilding (designing setting), and plotting (designing and sequencing events). Designing details for the author to interpret generally requires less information recall and shorter responses than composing prose. This project will be constrained to automated plotting – generating sequences of events – which is more difficult to automate, as the more complex of the subtasks, since it requires greater information recall for continuity and longer responses.

The core of my project will be based off replicating the main ideas of L. J. Martin et al.'s Controllable Neural Story Plot Generation via Reward Shaping [1] – attempting to train a neural network to generate a novel plot from a corpus of encoded plot summaries. I will compare the outputs from models trained on different datasets.

Starting Point

I have requisite knowledge on the creative writing side, with years of experience planning, plotting and writing my own fiction.

The technical knowledge required for this project will be:

- Already acquired in previous units or findable from base knowledge established in those, especially: Artificial Intelligence, Machine Learning and Real World Data, Formal Models of Language, Data Science...
- Studied in Michaelmas term in my Natural Language Processing unit
- Further refined by reading the recommended reading for:
  - Interaction with Machine Learning: Moral Codes by Alan Blackwell (`https://moralcodes.pubpub.org`)
  - Natural Language Processing: Computational Linguistics by Schubert, Lenhart (`https://plato.stanford.edu/entries/computational-linguistics/`)
- Specific AI/ML algorithms/techniques (and any other roadblocks of knowledge I come across) researched online and in the library
- I have not previously written any code to be used for this project. Generally, I have not programmed any natural language models before.

Resources required

Equipment

I plan to use my own computer for my dissertation and writing code. The specifications of my laptop are: 1.8GHz quad-core CPU, 8GB RAM, 256GB SSD storage with 48.3GB

free space remaining, Windows 10 OS. My contingency plans against data loss are that everything will be backed up to Dropbox, which will automatically upload any changes as soon as they are saved. This also protects against hardware failure as I can easily access my work from the Cloud on department workstations in the Intel Lab and progress on it using those instead if necessary.

Libraries & Tools

- Keras, to build the encoder-decoder neural network.
- For the reward-based learning, Williams' REINFORCE algorithm [7] or Q-learning.
- jenkspy Python library for Jenks natural breaks optimisation, for clustering verbs.
- Gensim Python library for Latent Dirichlet Allocation (LDA)

Datasets

L. J. Martin et al. [1] used the CMU Movie Summary Corpus [10]. It was found to be too diverse; as there is high variance between stories, which dilutes event patterns – thus, Latent Dirichlet Analysis was used to cluster the stories from the corpus into 100 categories. I will explore this similarly.

Alternatively, I could use IMDB movie synopses (additionally tagged with genres), with the same preprocessing steps as outlined by Lara Martin regarding the Sci-Fi TV show dataset [1] – thus deriving a multi-genre dataset, in which genres are another facet of the input. Regarding the extension task on incorporating additional knowledge on characters, etc., I could follow a similar method to the aforementioned Sci-Fi dataset of scraping the fandom.org wiki of shows, but also take character and background information to use as well.

I have downloaded any pre-existing datasets listed above, and backed them up on a 128GB USB flash drive.

Special Resources

None.

Human Subjects

As I plan to involve human subjects (through a questionnaire, or several depending on extensions), this will need approval by Ethics Committee.

Core Components

Although the core of this project will be mainly based on L. J. Martin et al.'s Controllable Neural Story Plot Generation via Reinforcement Learning [1], ideas and resources may also be explored from the same group's follow-up 2019 [2] and 2021 [3] papers.

The input is a single goal, representing the intended last event in the story, and simple background information to start from. This is input as natural language. The system converts the input sentences into 'events' – which are semantic abstractions of sentences – then attempts to create a plot, generating a sequence of events that progress to that goal event.

As defined by L. J. Martin et al., an event is a tuple e = ¡wn(s), vn(v), wn(o), wn(m)¿, where v is the verb of the sentence, s is the subject of the verb, o is the object of the verb, and m is a propositional object, indirect object, causal complement, or any other significant noun. The parameter o may take the special value of empty to denote there is no object of the verb, and m may be empty to denote any additional sentence information. [1]

wn(·) gives the WordNet [4] Synset of the argument two levels up in the hypernym tree (i.e. the grandparent Synset). The function vn(·) gives the VerbNet [5] class of the argument. Sentences may be split into multiple events, with a one-to-many relationship between a sentence and the events it produces. Events are used sequentially. [1]

A subtask will be to break down sentences in a chosen summary corpus to event objects.

I will model story generation as a planning problem: to find a sequence of events that transitions the state of the world into one in which the desired goal holds. L. J. Martin et al. [1] define the input goal to be that a given verb (e.g. kill, find, escape) is seen at the end of a plot – that a given VerbNet class is generated in the final event of the story.

The event sequencer is effectively a language model – deciding a sequence of events with a corpus, whereas normal LMs would decide a sequence of words. L. J. Martin et al. [1] use an encoder-decoder network [6] as a starting language model and baseline. Encoder-decoder networks can be trained to generate sequences of text for dialogue or story generation by pairing one or more sentences in a corpus with the successive sentence and learning a set of weights that captures the relationship between the sentences. [1]

Reinforcement learning addresses progression and coherence issues in story generation. By providing a reward each time a goal is achieved, a reinforcement learning agent learns a policy that maximizes the future expected reward.

L. J. Martin et al. [1] use reward shaping to reward the network whenever a gen-

erated event makes it more likely to achieve the input goal. Reward shaping [8] is a technique whereby sparse rewards—such as rewarding the agent only when a given goal is reached—are replaced with a dense reward signal that provides rewards at intermediate states in the exploration leading to the goal. Intermediate rewards are backpropagated into the pre-trained language model, encouraging generation of events that make significant progress towards the goal.

Note that the model I plan to implement doesn't produce a natural language output – thus, for human reading, a human translator is required to interpret the output. I will write the translations in order to present to judges in the human study, elaborated under Evaluation.

Success Criteria

I will evaluate the effectiveness of outputs and compare for models trained on different datasets. This will be done both quantitatively (statistically scoring the output on different metrics, well-defined tasks or performance-benchmarks) and qualitatively (further judgement by human participants).

I will come up with questions for judging responses to engage a panel of human judges – volunteering writers sourced from: people I know, various literary societies and book clubs I attend, and on social media if still considered insufficient.

I will discuss whether chosen metrics, scores or benchmarks (to be selected later) say the model performs well at its task for each dataset, against how the human judges and I think it actually performed in comparison.

Possible Extensions

Add multiple intermediate goals for a generated plot. This is simple in concept, as it could be achieved by feeding the goal of a previously generated event sequence back into the input of another generation as the next starting point.

Try different reinforcement learning algorithms, such as Q-learning. Find alternative reward techniques (to VerbNet in [1]) for plot plausibility and staying on topic. L. J. Martin et al. 2021 [3] explores Knowledge Representation and Reasoning. Another potential idea is to use reader models [9].

Evaluate the difference which genre makes to output quality, using single-genre subsets of a summary corpus. This may require another questionnaire, but could otherwise be evaluated by comparing scores quantitatively only.

Increase the complexity of the base 'events' given as input. This could potentially

include ground truths about the world of the story, requirements for characters to be involved, varying settings, chapter numbers, themes, emotional tone, specifying for development to be positive or negative, etc. To be effective, the goal space would have to be populated more densely. Discuss how well this more complex background information is actually incorporated in practice. For this task, I would likely have to consider balancing context-relevance (relatedness to input information) against surprisal (creativity/temperature of output).

Timetable

Supervisor meetings: Every Tuesday, 2:30pm

1. **Week 0** Find a new supervisor. Find a UTO. Discuss gaps in and feedback for proposal draft with supervisor. Proposal should be finished and submitted by deadline
2. **Weeks 1 - 2** Research libraries, statistical evaluation, human participant questionnaires, etc. Draw diagram to understand inputs/output/processes of intended model. Familiarise myself with Keras and reinforcement learning algorithms
3. **Weeks 3 - 4** Submit ethics review for committee approval.
4. **Weeks 5 - 6** Choose metrics/scores/benchmarks for evaluation. Gather questionnaire/study participants.
5. **Weeks 7 - 8** Finish code to process/prepare dataset. Gather questionnaire/study participants. Resubmit ethics review for committee approval with edits required.
6. **Weeks 9 - 10** Finish code to process/prepare dataset. Prepare statistical evaluation plan for quantitative results.
7. **Weeks 11 - 12** Write participant study questionnaires.
8. **Weeks 13 - 14** Finish participant study questionnaires.
9. **Weeks 15 - 16** Finish model definition and training code. Send preparatory emails to research volunteers, advising on what they will likely have to do and when. Write progress report and prepare presentation.
10. **Weeks 17 - 18** Finish model definition and training code. Generate results.
11. **Weeks 19 - 20** Generate results.
12. **Weeks 21 - 22** Translate generated results for questionnaire. Distribute questionnaires to study participants.
13. **Weeks 23 - 24** Code additional graphs for evaluation chapter.
14. **Weeks 25 - 26** Analyse and graph questionnaire results from participants. Finish dissertation.

15. **Weeks 27 - 28** Finish dissertation.

16. **Weeks 29 - 30** Finish and LaTeX dissertation.

Appendix B

Participant Study Questionnaire

Judging AI-generated Story Plots

Preface

You will be presented with a series of stories output by an A.I. model designed for coming up with fiction story plot ideas. The model is tuned to output romance stories specifically, but the content may also be mixed with other genres. Outputs have been manually filtered for any sexual or otherwise upsetting content.

You will be asked about attributes of these stories. Rate each of the outputs by each of these characteristics on a [0 – 10] scale. A high rating (10) would imply that the generated story strongly holds this characteristic, and a low rating (0) would imply that the characteristic does not hold for the story at all.

For your reference, each characteristic is defined in the glossary below, including a description of what a low or high value would mean for it. It will help to answer the questions easier if you take some time to read this glossary first.

This questionnaire should take around 20 – 30 minutes to complete.

Glossary

Likability: The story appeals to you personally.
0: I either actively dislike or am entirely disinterested in the contents of this story.
10: I really like the contents of this story. I find it interesting, inspirational or enjoyable.

Total Utility: The story in its entirety is usable for deriving a more detailed plot.
0: It would be impossible to use all or most of the story to write a plot that makes sense. All sentences of the story are useless to me or anyone else.

10: I could easily use all or most of the story to write a plot that makes sense.

Partial Utility: At least some event within the story is usable for coming up with a more detailed plot.
0: I could not use any sentence of the story to write a plot that makes sense.
10: I could easily use at least some part of the story to write a plot that makes sense.

Plausibility: How reasonable or probable the events of the story are given the initial state and the ending.
0: The events within the story are unrelated to or actively work against transforming the first sentence to the last one.
10: Events within the story plausibly transform the first sentence to the last one.

Local Causality: The contribution of any state or event within the story to the occurrence of any other later state or event. Linkage between pairs of successive sentences.
0: No circumstance, theme, character or happening are linked within or between sentences.
10: All or most circumstances, themes, characters and happenings are linked between sentences, and there is an understandable 'chain of events'.

Elaboration: The level of detail in which ideas are presented.
0: The events of the story are simple and sparse. It may be ambiguous what is happening because they are too simple.
10: There is a beyond-basic level of detail about events that happen in the story.

Novelty: The quality of being new, original, or unusual.
0: The events of the story are unexciting or overused in media, or the sentences don't conveying any information beyond mundane activities.
10: The events of the story are wacky or original.

Non-Repetition: Each story avoids repetition of events or earlier information within itself. Each of the stories are also significantly different from each other.
0: Earlier events or information are constantly repeated.
10: Every sentence in the story only contains new unseen information from earlier sentences.

Cohesiveness: The story has the properties 'entailment' (events are sequenced so they logically follow) and 'progression' (the state of the world changes through sentences)
0: The text is just a series of words/sentences with no connections between them.
10: The text tells a story of events that happen which transform a situation.

Grammaticality: Syntactic correctness.

0: Verbs, nouns and prepositions are constantly used in the wrong contexts and in place of each other.
10: Every sentence uses correct grammar.

Commonsensicality: Ideas can be perceived, understood, and judged easily.
0: Ideas in individual sentences are conceptually or physically nonsensical.
10: Ideas in individual sentences are conveyed regardless of grammar and have easily comprehensible meaning behind them.

Genre-Relevance: How connected the words, events and ideas are to themes prevalent in the romance genre.
0: The story could not be described as having any relation to the romance genre.
10: The story contains words and concepts relevant to the romance genre.

Stories

These are the generated stories, each presented in the format of: 'Prompt... *Completion.*':

1. A man works at a restaurant... *applying fragrances between customers' cuff-links. His ideas transform the local hospital between gossip. One day, his family member is hosted after an occultist, and the army judges that they must go bowling abroad a bomb. The bomber establishes a group until he finds himself yearning: the french fries improve this building in its features. His name is not clean, but his demand are to his apron. The man lies about his possession of a business, but the questioning ends when he registers his ownership of it through inheritance. A lack of concern kisses those who follow in the investor's footsteps.*

2. When Janet leaves her job... *this moment stumbles into the pursuit of a Casanova. Janet passes up her paramour with a request: that his money is to be involved in her property, despite his hypnotising charm. He repeats his hypnosis skills upon her chest, requesting that she covers him through the valley. The delusion of marriage throws them into failure.*

3. A woman is staying at the palace... *when a truck drives around her balcony. A quotation is being smuggled in by a race. A ladybird brings angst in the form of a bet. This break drives her to pray for anything except marriage. An informant harasses the male traveller, and his investigation corrects a shirt alongside some fish eggs. This defeat resembles a serenade as it follows beer. The treatment is completed by marrying as the throne.*

4. Sandra has a baby... *That information chokes and discourages a while ago a pawn. A fried chicken enriches Sandra's creativity through the suspect. This conclusion highlights the land of righteousness. A man marries an airplane by telephone.*

5. A woman boasts about her business... *Her old schoolfriend names abandonment towards this leak. Her husband engages in separated rendezvous in accordance with*

his manners. His request is that she confess as secret. The dub flirts a bank cheque into new beginnings. An offer breaks their kiss due to the wreckage of a boat.

6. A girl takes care of a baby... *Her grandparents try to hold off the inevitable divide between them despite the pouring rain. A gymnastic horse believed to be tired of opening. The Lord recognises the lake next to their misfortune. The guard tries to marry through this god.*

Questions

Rating questions have been omitted for this report. The format of these is simply a draggable slider from value 0 to 10, for each attribute.