Goal-Directed Story Generation: Augmenting Generative Language Models with Reinforcement Learning

Anonymous ACL submission

Abstract

The advent of large pre-trained generative language models has provided a common framework for AI story generation via sampling the model to create sequences that continue the story. However, sampling alone is insufficient for story generation. In particular, it is hard to direct a language model to create stories to reach a specific goal event. We present two automated techniques grounded in deep reinforcement learning and reward shaping to control the plot of computer-generated stories. The first utilizes proximal policy optimization to fine-tune an existing transformer-based language model to generate text continuations and be goal-seeking. The second extracts a knowledge graph from the unfolding story, which a policy network uses with graph attention to select a candidate continuation generated by a language model. We report on automated metrics on how often stories achieve a given goal event and human participant rankings of coherence and overall story quality compared to baselines and ablations.

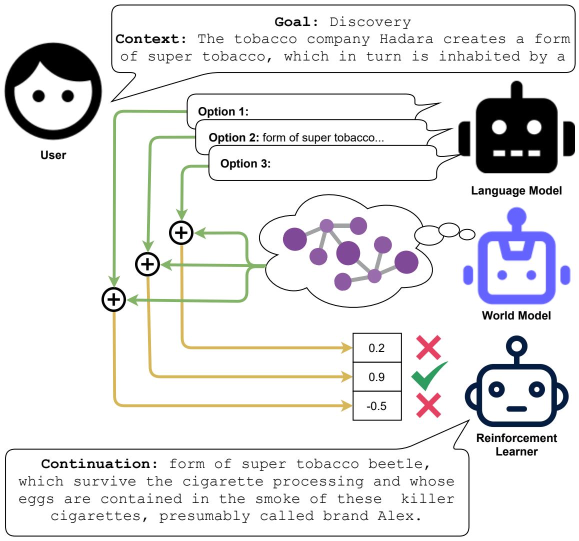

Figure 1: A single iteration in our generation system. Given a goal and a context prompt, a language model is queried for a number of possible continuations. The system also maintains an abstract graph-based representation of the entities and relations in the story world. A reinforcement learning agent has learned to select options based on the abstract state and how likely the option is to move the story toward the given goal. Not shown: the world model is then updated and the continuation is added to the context for the next iteration.

1 Introduction

Automated Story Generation is the challenge of designing an artificial intelligence system that can generate a story from a minimal number of inputs—often just a prompt and some storytelling knowledge and/or storytelling model. *Goal-directed* story generation is the challenge of generating stories with predetermined goals. A goal-driven story generation system provides the model with a goal, generating a story that progresses towards it.

In this paper we have two aims. First, we show that a fine-tuning approach first introduced by Tambwekar et al. (2019) that worked on LSTMs does not directly translate to more modern large pre-trained language models such as GPT-2 (Radford et al., 2019). Large pre-trained language models produce more natural language and can handle a larger range of inputs but, like seq2seq models, are not inherently goal-driven. Unfortunately, we observe that large language models are harder to control; our experiments with reward shaping based fine-tuning toward a given goal only results in a 50% goal achievement rate (although fluency of story outputs is greatly improved).

Our second aim is to introduce a new technique in which we train a second neural *policy model* to guide a non-fine-tuned GPT-2 to a given end-goal, achieving 90+% goal success rate while retaining the language model's fluency. The second model operates on an abstracted state space represented as a knowledge graph—a set

* Denotes equal contribution.

of $\langle subject, relation, object \rangle$ tuples. This knowledge graph state representation explicitly captures the entities in a story and their relations instead of relying on the hidden state of the language model to represent the state of the story world accurately. Given a knowledge graph representing the state of the story world, our policy model predicts the state's utility, which is proportional to the number of sentences needed to achieve a goal. We sample plausible continuations from GPT-2 while our policy model selects continuations based on how they move their role in story progression.

We report on a combination of automated and human participant evaluations. We focus on evaluating our system in the domain of science fiction plots (Ammanabrolu et al., 2020a), consistent with prior work (Tambwekar et al., 2019; Ammanabrolu et al., 2020a). The automated evaluation shows our technique achieves the desired-end goal 98.73% of the time. Our human participant studies show that the full model is perceived to be significantly more coherent than baseline alternatives.

2 Background and Related Work

Many early works on story generation used planning (Meehan, 1976; Lebowitz, 1987; Cavazza et al., 2003; Porteous and Cavazza, 2009; Riedl and Young, 2010; Ware and Young, 2010). In many cases, these systems are provided with a goal or outcome state. For example, a goal might be "characters X and Y are married". These approaches require extensive domain knowledge engineering and templated language. Neural language models have been used for story generation, circumventing the need for knowledge engineering, and can produce relatively fluent and natural language (Roemmele, 2016; Khalifa et al., 2017; Clark et al., 2018; Martin et al., 2018).

We situate our work in neural approaches to story generation. Neural language model based approaches to story generation start with a given text prompt and generate story continuations by sampling from a learned distribution over tokens; models trained on a corpus of the story will produce text that appears to be a story (Roemmele, 2016; Khalifa et al., 2017; Clark et al., 2018; Martin et al., 2018). These techniques have improved with the adoption of large, pre-trained, transformer-based models, such as GPT-2 (Radford et al., 2019), which can be fine-tuned on representative data from a particular domain. However, larger models such as GPT-3 (Brown et al., 2020) may be behind closed APIs that do not allow fine-tuning, making it more important that we have solutions to the problem of controllable text generation that do not rely on fine-tuning of language models.

Neural story generators are inherently "backward-looking" in the sense that they produce tokens or sequences that are likely to occur based on a window of prior tokens. As a result, it is challenging to control the story's direction will unfold and neural generated stories tend to meander or become repetitive. Story generation can be controlled by conditioning generation on high-level plot outlines (Fan et al., 2018; Peng et al., 2018; Rashkin et al., 2020) or story in-filling (Donahue et al., 2020). However, these techniques assume a human or other source has already determined the key plot points, and the generator provides missing details. Coherence can also be increased by systems that generate their plot-level abstractions and then condition a language model on those plot labels (Yao et al., 2019; Fan et al., 2019; Peng et al., 2021). While improving perceptions of narrative coherence, these techniques cannot guarantee goal achievement.

One way to ensure goal achievement is to provide the final event/sentence of a story in addition to the first event/sentence of a story as inputs. Wang et al. (2020) propose a generation-by-interpolation approach to story generation, using GPT-2 to generate several candidates to go in between and then choose based on perplexity. The C2PO system (Ammanabrolu et al., 2021) uses bi-directional search from a given start and given end, operating in the space of character goals as inferred by the COMET commonsense inference model (Bosselut et al., 2019), generating templated "states". The EDGAR system (Castricato et al., 2021) generates backward from a given last sentence but cannot guarantee a given first sentence.

For their goal-driven story generation system, Tambwekar et al. (2019) trained a LSTM-based sequence-to-sequence neural model to generate continuations while also increasing the likelihood of achieving a given goal event. The LSTM model produced *events*—tuples of the form $\langle subject, verb, object, modifier \rangle$. While their model reliably achieved the given goal, sequences of event tuples are not human-readable, requiring either manual rewriting or a second model to translate events into human-readable sentences such as (Ammanabrolu et al., 2020a). A natural progression to this work would be to apply the reward-shaping based fine-tuning approach from Tambwekar et al. to a large pre-trained language model, which we demonstrate in section 4.

Knowledge graphs have been shown to improve natural language understanding in related story-like domains such as interactive narrative game playing (Hausknecht et al., 2020). Knowledge graphs "lift" language to a structured representation, which facilitates forward-looking planning algorithms such as reinforcement learning (Ammanabrolu and Riedl, 2018; Ammanabrolu and Hausknecht, 2020; Ammanabrolu et al., 2020b; Ammanabrolu and Riedl, 2021).

3 Methodology

Our goal is to generate stories that (1) are able to reliably reach a specific goal, (2) follow a logical plot leading to the goal sentence, (3) and have reasonable lengths (ie. do not jump straight to generating a closing sentence). For our system a goal is any sentence that contains a verb that is a member of a specific, given VerbNet (Schuler, 2005) class. For example, the verbs: find, guess, and solve are members of the VerbNet class `discover-84`.

Our method depends on two models: a language model (GPT-2 (Radford et al., 2019)) and a policy model trained via reinforcement learning to select alternative continuations that progress the story incrementally toward the goal. We show that both models are needed to achieve all three objectives.

3.1 Dataset and Preprocessing

We use the science fiction plot corpus (Ammanabrolu et al., 2020a). The dataset contains 1400+ generalized science fiction stories of variable lengths. The stories have named entities replaced with tags denoting category and number within a story, maintaining consistency for named entities. We perform the following pre-processing steps:

1. **Dataset Splitting** We performed a 70:30 split on our training data, resulting in sets of size 1102 and 472 stories, respectively.

2. **Verb class extraction** For each sentence, we parse the sentences using SpaCy (Honnibal et al., 2020), and then we extract the verbs from the parsed sentences. We then lemmatize each verb and match it to its VerbNet class (Schuler, 2005).

3. **Tokenization** We use the Huggingface pre-trained GPT2 tokenizer to tokenize our sentences and prepare our model's input queries.

We fine-tune GPT-2 on the science fiction dataset; we refer to this model as *GPT2-sci-fi*.

3.2 Reward Function

Our policy model is trained using reinforcement learning. The reward function must produce greater reward for sentences that are more likely to be found near a goal and less reward for sentences that are less likely to be found near a goal. For purposes of experimental simplicity, a goal is a sentence that contains a verb that is a member of a given goal VerbNet verb class (e.g., `admire`, which encapsulates verbs such as "love"). Any VerbNet class can be chosen, though the choice affects goal achievement; common verb classes result in high achievement regardless of model, and sparse verbs result in a poor reward signal. For example, "discover" has 43% of the frequency of our most common verb class ("say"), and "admire" 25%. We verified that goal rates are acceptable in our base corpus (Table 1) before proceeding.

Reward shaping refers to the pre-computation of rewards for a particular decision space based on heuristically computed approximations of true utility. We use a reward function adapted from Tambwekar et al. (2019). Given an input prompt—a complete sentence describing an event in the story being generated—a continuation sentence is generated. Output is truncated at the first period or 20 tokens, whichever comes first. The verb is extracted from the continuation and a reward $R(v)$, which is made up of two components. The first component computes the distance a sentence with a given verb class tends to be from the goal:

$$r_1(v) = \log \sum_{s \in S(v,g)} (\text{len}(s) - \text{dist}_s(v,g)) \quad (1)$$

where $s \in S(v,g)$ is the set of all stories containing both the current verb class v and the goal verb class g. $\text{len}(\cdot)$ is the number of sentences in the story and $\text{dist}_s(\cdot)$ is the number of sentences in s between that with v and the sentence with g. The second component computes the frequency that a verb class co-occurs in a story along with the goal:

$$r_2(v) = \log \frac{\text{count}(v,g)}{\text{count}(v)} \quad (2)$$

where $\text{count}(v,g)$ is the number of stories containing the verb class v and the goal verbclass g,

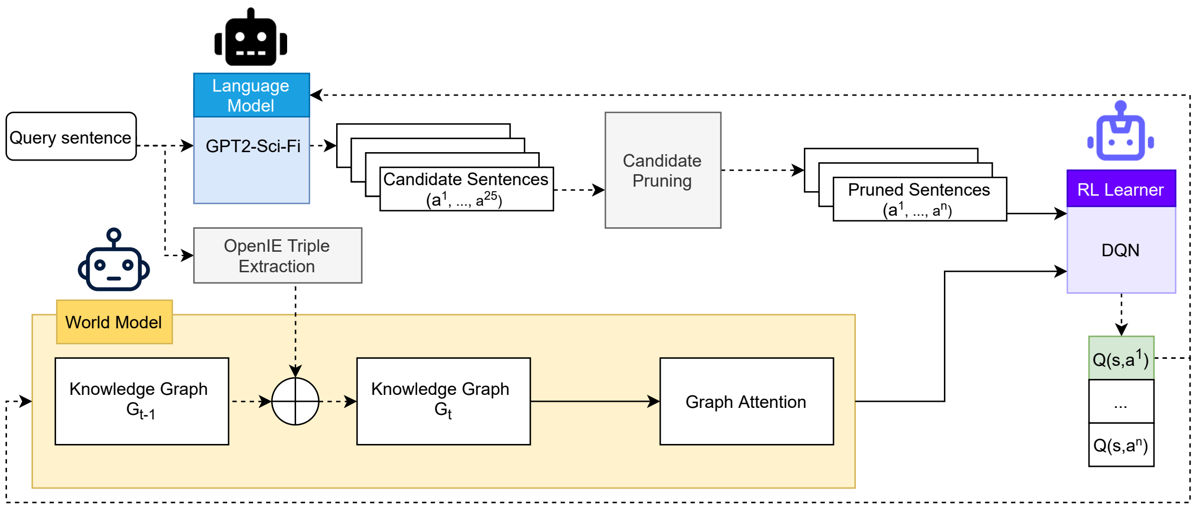

Figure 2: KG-DQN architecture. The DQN model and the graph attention are the only trainable components. The language model is a frozen component. The solid lines indicate gradient flow.

whereas $\mathrm{count}(v)$ is the number of stories containing v. The final reward is given as:

$$R(v) = \frac{1}{|verbs|} \times r_1(v) \times r_2(v) \qquad (3)$$

For efficiency, the reward for all verbs are computed and stored in advance.

3.3 Verb Clustering

Tambwekar et al. (2019) observed that naively rewarding a story generator based on distance to a goal verb induces a model to skip to the end of a story in a single continuation. To ensure that the model generates sequential sentences in the story and does not jump directly to the goal sentence, verbs are clustered depending on their reward value as calculated in equation 3. Following Tambwekar et al., we use the Jenks Natural Breaks optimization technique (Jenks and Caspall, 1971), an off-the-shelf clustering algorithm appropriate for our 1-Dimensional non-uniform data. However, clusters computed with methods such as 1D k-means, or even quantiles or equal intervals would have still resulted in ordered clusters as described above. The result is a set of ordered verb clusters estimating how "close" a verb class is to the goal verb class. These clusters provide further reward guidance, though applied differently depending on the specific reinforcement learning algorithm used. During generation, the "source" is the most recent verb class's cluster, and the "target" is the next consecutive cluster in the direction of the goal cluster.

3.4 Knowledge Graph Guided Deep Q Network

The policy model is trained to pick the best continuation from an un-tuned language model. We formulate the problem of selecting a continuation sentence that moves the story closer to the goal as a Markov Decision Process (MDP) with the state being a knowledge graph that represents the current state of the story world. A knowledge graph is a set of binary relation triples of the form $\langle subject, relation, object \rangle$. This story world state representation explicitly captures the entities in a story and the relations between entities. This constrains the state space to a discrete, but infinite, set of graphs. It also allows the policy model to focus on elements of the story world that are likely to matter for the purposes of maintaining logical coherence and goal achievement.

3.4.1 Policy Model Architecture

Our policy model is a variation of Deep Q Networks (DQN). A DQN takes a world state observation and predicts$Q(s, a_i)$ the utility of an action a_i performed in a state s. In the case of our system, a state is a knowledge graph and an action is a continuation generated by GPT2-sci-fi. Each action/continuation is embedded and concatenated to an embedding of the knowledge graph. The nodes of the knowledge graph are embedded using GPT2-sci-fi word embeddings, then we obtain a single vector representation using multi-headed graph attention (Veličković et al., 2018). The graph is concatenated with the continuation and passed through a fully connected layer to produce the utility for that combination of graph and continuation.

Knowledge graph triples are extracted from sentences using Stanford Open Information Extraction (OpenIE) (Angeli et al., 2015). OpenIE is a common and well-established method of extracting relation tuples, that has been demonstrated on related domains (cf. (Ammanabrolu and Riedl, 2018; Ammanabrolu and Hausknecht, 2020)), and therefore a good proof of concept despite potential noise. The knowledge graph is updated with new triples after each sentence added to the story, thus creating an implicit transition function. We train this policy network model by using the output of the base GPT2-Sci-Fi as if continuation sentences are actions and the knowledge graph as a discrete state, using the DQN algorithm (Mnih et al., 2015). We refer to our full knowledge graph informed policy model as *KG-DQN* and is shown in Figure 2.

3.5 Policy Model Training

KG-DQN takes as input a knowledge graph and a potential action—a sentence generated by a frozen GPT2-sci-fi—and outputs the expected utility $Q_\theta(G, a)$ of taking that action a given the current state knowledge graph G. The standard DQN training loop (Mnih et al., 2015) populates an experience replay buffer, which collects up combinations of states, actions, and rewards. The experience replay buffer is sampled and loss is computed relative to a target network, which is periodically replaced by a frozen version of the DQN being trained. The full training loop is as follows:

1. Given a current query sentence a_t and knowledge graph G_t generate 25 potential candidates using GPT2-sci-fi.
2. Clean candidates that do not contain complete sentences or verbs. Truncate the remaining candidates to form complete sentences.
3. Choose the next action a_{t+1} ϵ-greedily. The chosen action and updated knowledge graph will become inputs for the next time step.
4. Using OpenIE, extract relevant knowledge triples and update knowledge graph to G_{t+1}.
5. Add $\langle G_t, a_{t+1}, G_{t+1}, reward, query\rangle$ tuple to our experience replay buffer.
6. Every 100 stories, sample our replay buffer and calculate $y_j = r_j$ for terminating actions, or $y_j = r_j + \gamma \mathbf{max}_{a'} Q_\theta(G_{j+1}, a')$ for non-terminating ones. Perform gradient descent in Q with loss $(y_j - Q_{\theta_t}(G_j, a_j))^2$ where θ_t is the frozen target network.
7. Every 300 stories, update target network θ_t.

During step 3 in the training loop, the agent can choose exploration with probability ϵ or exploitation. If exploitation is chosen we first tentatively prune out actions that do not lead from the current verb cluster to the next "closest" verb cluster to the goal. If that is impossible, we then fall back to picking the action/sentence according to the highest estimated reward from the set of all candidates. This is contrast to the standard approach of choosing the action with the maximum predicted utility from our policy. In this way we bias the model towards choosing cluster-optimal jumps and discourage behavior like jumping immediately to the goal (which would not create a coherent story).

4 Baselines and Additional Models

All generation models are based on the GPT-2 117M model (Radford et al., 2019) fine-tuned for story generation on NER-replaced science fiction data (Ammanabrolu et al., 2020a), which we refer to as *GPT2-sci-fi*. We experiment with five different models:

1. **GPT2-Sci-Fi**, the baseline model.
2. **GPT2-RS**, the base story generation model additionally fine-tuned with reward shaping based on the reward function in Section 3.2. This model is a near-literal update of Tambwekar et al. (2019) to work on GPT2 instead of a custom-trained seq2seq language model. Details of the model and its training are below.
3. **KG-DQN**, our model (see Section 3).
4. **DQN**, Same as above but ablating/removing the knowledge graph representation. In this case, the state is the same as the action—the input sentence. This model has 36,138 trainable parameters compared to KG-DQN with 177,750 trainable parameters.
5. **KG-DQN-RS**, combining the KG-DQN policy with a frozen GPT2-RS fine-tuned network instead of GPT2-sci-fi.

We did not include systems that use high-level plot guidance inputs (eg. (Fan et al., 2018; Peng et al., 2018; Rashkin et al., 2020)) as baselines because they accept additional guidance inputs that our technique does not have access to. Nor do we include systems that are not goal-directed (eg., (Fan et al., 2019; Yao et al., 2019; Peng et al., 2021)) as baselines because they are not encumbered by the additional success criteria. We also do not include infilling systems (eg. (Donahue et al., 2020; Ammanabrolu et al., 2021; Wang et al., 2020).

We did not include the technique by Tambwekar et al. (2019) directly as a baseline because they used humans to transcribe event tuples into natural language sentences. Our technique, however, is designed to generate natural language. Without the translation, the event tuples are uninterpretable and with it there cannot be a fair comparison. Unfortunately, the technique described by Tambwekar et al. also cannot be directly applied to models that produce full-sentence outputs instead of event tuples. One of the issues with fine-tuning GPT-2 on verb usage is that verbs that move the story closer to a goal ending may be rare. Re-sampling (aka teacher-forcing) can be used when the vocabulary draws from event tuples. However, swapping a verb without rewriting the entire continuation sentence can produce nonsensical results and was found not to improve goal-reaching behavior when applied to GPT-2. Instead, our GPT2-RS baseline acts as an update to Tambwekar et al. to account for neural language models that produce sentence outputs. We update the technique with an alternative verb restriction approach to make sure that our model sees mostly verbs from our clusters during training. During training, for each given query, we generate twenty candidate sentences. We then we check if any of these sentences satisfies the following condition on the order and distance of source and target verbs' clusters:

1. The difference between the target verb cluster and the source verb cluster is one or zero. This sentence will get full reward based on Equation 3.

2. The difference between the target verb and the source verb clusters is positive but higher than one. This means that the verbs are in the correct order but are further than they should be. Here, we discount the rewards by a factor of one over the difference between the clusters.

In all other cases, the model will give no rewards. The clustering ensures that stories have reasonable lengths and reduce the model's bias to immediately produce sentences containing the goal verb.

As an initial step, the model is given a randomly sampled batch of the training sentence as query sentences; the objective is to fine-tune an underlying language model using the reward shaping function. We include more details about the GPT-RS training steps in appendix B.

In time, the GPT2-RS training process fine-tunes the base language to prefer certain continuations based on how likely they are moving the story toward the given end-goal.

5 Automated Evaluation

To evaluate our models, we conduct an extensive set of automatic and human evaluations in order to determine: (1) Does GPT2-RS-based RL fine-tuning with our described reward function work well when applied to GPT-2 for story generation? (2) Is there a significant difference between using a policy gradient method (PPO) versus a value-based method like DQN? (3) Does the explicit inclusion of knowledge graphs affect story generation performance? Our automated evaluation metrics are:

1. Goal Achievement Rate—the percentage of stories that produced a sentence with the target verb.
2. Average Perplexity scores.
3. 4-gram repetition (Guan et al., 2020), measures the fraction of stories with at least one repeated 4-gram.
4. Average Story Length in number of sentences.

5.1 Experimental setup

We selected two verb class goals, `admire-31.2` and `discover-84`, chosen because they are sparse enough to not be generated by accident but not rare for stories. We then trained models for each goal. We include the details of the models training in appendix C.

We use our test set of 472 first sentences from the Sci-Fi stories dataset as seeds to generate stories using our RL trained models and baseline models. All models could generate up to a maximum of 15 continuations to achieve the goal (for a total length of 16 sentences). Story generation terminated when the target goal was reached, the model failed to any more valid sentences, or 15 total sentences had been generated without meeting the goal condition. For generation, KG-DQN-RS, KG-DQN, and

Goal	Model	Goal Rate	Story Length	REP-4
admire	Base Corpus	32.42%	86.51	0.24
	GPT2-Sci-Fi	16.74%	15.01	0.3556
	GPT2-RS	41.95%	8.67	0.0339
	DQN	72.25%	4.54	0.2331
	KG-DQN	91.74%	4.84	0.1314
	KG-DQN-RS	**94.70%**	5.11	0.0445
discover	Base Corpus	49.47%	86.51	0.24
	GPT2-Sci-Fi	18.86%	15.26	0.3369
	GPT2-RS	33.47%	9.71	0.0466
	DQN	96.19%	4.56	0.1123
	KG-DQN	98.09%	4.72	0.1123
	KG-DQN-RS	**98.73%**	4.38	0.0466

Table 1: Results of Automated Experiments.

Goal	Model	Perplexity
admire	GPT2-Sci-Fi	39.36
	GPT2-RS	40.034
discover	GPT2-Sci-Fi	39.36
	GPT2-RS	38.95

Table 2: Perplexity Values

DQN were all run with a "breadth" of 25 candidate stories per step, generating from our test stories, with evaluation taking about 30 minutes per model. For GPT2-sci-fi and GPT2-RS we ran them as generative models.

5.2 Automated Evaluation Results

The results of our automated experiments can be found in Table 1, and perplexity values in Table 2. We do not compute a perplexity for DQN models, as the generated tokens are from the base model, which is frozen. In order to best align results to compare models, in our model-generated stories, story length was only taken from stories that reached the goal. For the base-corpus, REP-4 was calculated over the first 16 sentences only.

As can be seen from Table 1, in all our experiments KG-DQN-RS achieve the goal most often, slightly more often than KG-DQN. For both goals, GPT2-RS provided modest gains over baseline, but not as much as our DQN models. With both goal verbs, DQN models provide significant gains over baseline goal reaching behavior, with the knowledge graph augmented DQN outperforming the vanilla DQN, implying that knowledge graphs are important for goal reaching behavior (although in the `discover` goal the difference is less distinct). Also as can be seen from KG-DQN-RS, our DQN based approach is independent of the underlying language model used to generate candidates.

While the DQN models are able to achieve reductions in REP-4 score, the presence of GPT2-RS finetuning seems to account for the greatest reductions in repetition.

Although GPT2-RS fine-tuning alone was enough to bring goal-reaching over the GPT2-Sci-Fi baseline, perplexity is essentially unaffected,

6 Human Participant Evaluations

Human participant evaluations are believed to be the best practice in evaluating generated story quality. We asked human judges to compare pairs of stories generated by different models given the same input prompts. Judges had to choose the better story (or equal) according to four criteria:

- **Grammar:** This story exhibits correct grammar.
- **Avoids Repetition:** The story avoids repetition.
- **Plausable Order:** This story's events occur in a plausible order.
- **Local Causality:** This story's sentences make sense given sentences before and after them.

These questions have been used in a number of other story generator evaluations (Purdy et al., 2018; Tambwekar et al., 2019; Ammanabrolu et al., 2020a, 2021; Castricato et al., 2021; Peng et al., 2021). Plausible order and local causality questions are surrogates for *coherence*, which can be interpreted by human judges in different ways.

6.1 Experimental Setup and Methodology

As mentioned in Section 3.1, our baseline GPT2-Sci-Fi was fine-tuned on NER-replaced data. This NER replacement makes story comprehension difficult for human subjects. We use FitBERT (Havens and Stal, 2019) which is a pre-trained BERT (Devlin et al., 2019) to "fill in the blanks" using named entities from the original Sci-Fi dataset for the relevant category and taking the option with the best BERT-score (Zhang et al., 2020). We use these re-populated sentences for human evaluation.

We chose a common set of seeds that were successes with all but GPT2-Sci-Fi and sampled a further subset of 80 seeds for human evaluation.

We recruited 64 participants on Prolific (www.prolific.co). We presented each participant with five pairwise comparisons. The story pairs given in randomized order, ensuring that all pairs are seen equally by participants and that order does not impact the participants' answers.

The average completion time for this task is 30

Models		Grammar		Avoids Repetition		Plausible Order		Local Causality	
		Win %	Lose %	Win %	Lose %	Win %	Lose %	Win %	Lose %
admire	KG-DQN-RS vs KG-DQN	24.81*	61.29	25.81*	38.71	9.68*	45.16	9.68*	61.29
	KG-DQN vs DQN	29.03*	29.03	35.48*	22.58	38.71*	25.81	45.16*	22.58
	GPT2-RS vs GPT2-Sci-Fi	6.45*	38.71	35.48*	19.35	19.35*	32.26	16.13*	38.71
	KG-DQN vs GPT2-Sci-Fi	58.06*	12.90	29.03*	25.81	61.29*	19.35	61.29*	19.35
	KG-DQN vs GPT2-RS	45.16*	12.90	48.39*	16.13	41.93*	25.81	38.71*	16.13
discover	KG-DQN-RS vs KG-DQN	12.5*	50.0	12.5*	25.0	15.63*	56.25	15.63*	56.25
	KG-DQN vs DQN	34.38*	28.13	25.0*	31.25	31.25*	25.0	28.13*	43.75
	GPT2-RS vs GPT2-Sci-Fi	16.13*	51.62	62.5*	25.81	22.58*	48.39	16.13*	54.84
	KG-DQN vs GPT2-Sci-Fi	71.88*	9.38	34.38*	9.38	45.16*	22.58	50.0*	12.5
	KG-DQN vs GPT2-RS	34.38*	6.25	38.71*	16.13	50.0*	21.88	46.88*	25.0

Table 3: Human-participant pairwise evaluation results showing the percentage of participants who preferred the first model vs. the second. Each model was conditioned on the same eight first story sentences. * indicates significant results at $p < 0.01$ confidence level using a Wilcoxon sign test on win-lose pairs. Green cells are wins of the first model, and yellow cells are ties.

minutes, and the participants were compensated $6 upon successful completion. To ensure high data quality, we added a checker question to ensure that the person reads and understands the tasks and a text field. In addition, every comparison asks the participants to explain their answers.

6.2 Human Participant Study Results

We show the results of our pairwise comparison experiment in Table 3. Our results show that participants preferred the stories generated GPT2-RS model over the baseline GPT2-sci-fi model in repetition avoidance for both goals. However, participants prefer the baseline in all other dimensions.

The two-network KG-DQN model significantly ($p < 0.01$) outperforms the GPT2-sci-fi baseline and GPT2-RS along all dimensions. We note that participants often report ties; our analyses show that when participants report a difference, the difference significantly favors KG-DQN. One reason for the high occurrence of ties is that it is easy for participants to escape tricky judgements by choosing to report a tie.

The DQN ablation (removing the knowledge graph from KG-DQN) shows a degradation of performance, strongly suggesting that the knowledge graph state representation is the factor in the reinforcement learning of the second network that plays the most important role.

When we add the reward shaped language model fine-tuning into KG-DQN to create KG-DQN-RS, we do not observe much to improving the story generation except for repetition avoidance.

6.3 Qualitative Evaluation of Stories

Even though our models do a great job generating stories that reach a specified goal, we make some observations about the stories. We notice that the longer the story gets, the more hallucinations make their way into the story, diverging from the main topic even when the verbs are still going in the direction of the goal. Stories generated with the GPT2-RS model tend to be longer with many random character introductions, while the KG-DQN-RS, KG-DQN and DQN models are much more concise with a clear direction towards the goal. We see this is Table 1, where the DQN-based models tend to be shorter, achieving a goal verb after on average 4-5 events. This is likely due to the verb clusters, which control how many “hops” the system has to take before arriving at the goal. Tables 4 and 5 in Appendix E exemplify our models’ output for a given prompt when trained on the goals, `Admire-31.2` and `Discover-84`.

7 Conclusions

Large-scale pre-trained language models are difficult to control, especially in the case of neural story generation in which one desires a story to end with a desired goal event. While there has been some progress on making neural story generation models capable of progressing toward a given goal, those techniques do not transfer to large pre-trained language models. We show that reinforcement learning can be used to train policy models that are goal-aware and guide large language models. These results provide a step toward story generation capabilities that benefit from the quality of language generation afforded by large pre-trained language models and also the ability to specify how a story ends, which up until now has not been achieved.

References

Daniel Adiwardana, Minh-Thang Luong, David R So, Jamie Hall, Noah Fiedel, Romal Thoppilan, Zi Yang, Apoorv Kulshreshtha, Gaurav Nemade, Yifeng Lu, et al. 2020. Towards a human-like open-domain chatbot. *arXiv preprint arXiv:2001.09977*.

Prithviraj Ammanabrolu, Wesley Cheung, William Broniec, and Mark O. Riedl. 2021. Automated storytelling via causal, commonsense plot ordering. *CoRR*, abs/2009.00829.

Prithviraj Ammanabrolu and Matthew Hausknecht. 2020. Graph constrained reinforcement learning for natural language action spaces. In *International Conference on Learning Representations*.

Prithviraj Ammanabrolu and Mark Riedl. 2021. Learning knowledge graph-based world models of textual environments. In *Thirty-fifth Conference on Neural Information Processing Systems*.

Prithviraj Ammanabrolu and Mark O. Riedl. 2018. Playing text-adventure games with graph-based deep reinforcement learning. *CoRR*, abs/1812.01628.

Prithviraj Ammanabrolu, Ethan Tien, Wesley Cheung, Zhaochen Luo, William Ma, Lara J Martin, and Mark O Riedl. 2020a. Story realization: Expanding plot events into sentences. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 7375–7382.

Prithviraj Ammanabrolu, Ethan Tien, Matthew Hausknecht, and Mark O Riedl. 2020b. How to avoid being eaten by a grue: Structured exploration strategies for textual worlds. *arXiv preprint arXiv:2006.07409*.

Gabor Angeli, Melvin Johnson Premkumar, and Christopher D. Manning. 2015. Leveraging linguistic structure for open domain information extraction. In *Association for Computational Linguistics (ACL)*.

Antoine Bosselut, Hannah Rashkin, Maarten Sap, Chaitanya Malaviya, Asli Celikyilmaz, and Yejin Choi. 2019. Comet: Commonsense transformers for automatic knowledge graph construction. *arXiv preprint arXiv:1906.05317*.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners. In *Advances in Neural Information Processing Systems*, volume 33, pages 1877–1901. Curran Associates, Inc.

Louis Castricato, Spencer Frazier, Jonathan Balloch, Nitya Tarakad, and Mark Riedl. 2021. Tell me a story like i'm five: Story generation via question answering. In *Proceedings of the NAACL 2021 Workshop on Narrative Understanding*.

Marc Cavazza, Olivier Martin, Fred Charles, Steven J Mead, and Xavier Marichal. 2003. Interacting with virtual agents in mixed reality interactive storytelling. In *International Workshop on Intelligent Virtual Agents*, pages 231–235. Springer.

Elizabeth Clark, Yangfeng Ji, and Noah A Smith. 2018. Neural text generation in stories using entity representations as context. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 2250–2260.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Chris Donahue, Mina Lee, and Percy Liang. 2020. Enabling language models to fill in the blanks. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*.

Angela Fan, Mike Lewis, and Yann Dauphin. 2018. Hierarchical neural story generation. *arXiv preprint arXiv:1805.04833*.

Angela Fan, Mike Lewis, and Yann Dauphin. 2019. Strategies for structuring story generation. *arXiv preprint arXiv:1902.01109*.

Jian Guan, Fei Huang, Zhihao Zhao, Xiaoyan Zhu, and Minlie Huang. 2020. A knowledge-enhanced pretraining model for commonsense story generation. *Transactions of the Association for Computational Linguistics*, 8:93–108.

Matthew Hausknecht, Prithviraj Ammanabrolu, Marc-Alexandre Côté, and Xingdi Yuan. 2020. Interactive fiction games: A colossal adventure. In *Thirty-Fourth AAAI Conference on Artificial Intelligence (AAAI)*.

Sam Havens and Aneta Stal. 2019. Use bert to fill in the blanks.

Matthew Honnibal, Ines Montani, Sofie Van Landeghem, and Adriane Boyd. 2020. spaCy: Industrial-strength Natural Language Processing in Python.

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

Ahmed Khalifa, Gabriella AB Barros, and Julian Togelius. 2017. Deeptingle. *arXiv preprint arXiv:1705.03557*.

Michael Lebowitz. 1987. Planning stories. In *Proceedings of the 9th annual conference of the cognitive science society*, pages 234–242.

Lara Martin, Prithviraj Ammanabrolu, Xinyu Wang, William Hancock, Shruti Singh, Brent Harrison, and Mark Riedl. 2018. Event representations for automated story generation with deep neural nets. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32.

James Richard Meehan. 1976. The metanovel: Writing stories by computer. Technical report, YALE UNIV NEW HAVEN CONN DEPT OF COMPUTER SCIENCE.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin Riedmiller, Andreas K. Fidjeland, Georg Ostrovski, Stig Petersen, Charles Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dharshan Kumaran, Daan Wierstra, Shane Legg, and Demis Hassabis. 2015. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533.

Nedjma Ousidhoum, Xinran Zhao, Tianqing Fang, Yangqiu Song, and Dit-Yan Yeung. 2021. Probing toxic content in large pre-trained language models. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 4262–4274, Online. Association for Computational Linguistics.

Nanyun Peng, Marjan Ghazvininejad, Jonathan May, and Kevin Knight. 2018. Towards controllable story generation. In *Proceedings of the First Workshop on Storytelling*, pages 43–49.

Xiangyu Peng, Siyan Li, Spencer Frazier, and Mark Riedl. 2020. Reducing non-normative text generation from language models. In *Proceedings of the 13th International Conference on Natural Language Generation*, pages 374–383.

Xiangyu Peng, Siyan Li, Sarah Wiegreffe, and Mark Riedl. 2021. Inferring the reader: Guiding automated story generation with commonsense reasoning.

Julie Porteous and Marc Cavazza. 2009. Controlling narrative generation with planning trajectories: the role of constraints. In *Joint International Conference on Interactive Digital Storytelling*, pages 234–245. Springer.

Christopher Purdy, Xinyu Wang, Larry He, and Mark Riedl. 2018. Predicting generated story quality with quantitative measures. In *Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*, volume 14.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9.

Hannah Rashkin, Asli Celikyilmaz, Yejin Choi, and Jianfeng Gao. 2020. Plotmachines: Outline-conditioned generation with dynamic plot state tracking. *arXiv preprint arXiv:2004.14967*.

Mark O Riedl and Robert Michael Young. 2010. Narrative planning: Balancing plot and character. *Journal of Artificial Intelligence Research*, 39:217–268.

Melissa Roemmele. 2016. Writing stories with help from recurrent neural networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 30.

Karin Kipper Schuler. 2005. Verbnet: A broad-coverage, comprehensive verb lexicon.

Abigail See, Aneesh Pappu, Rohun Saxena, Akhila Yerukola, and Christopher D Manning. 2019. Do massively pretrained language models make better storytellers? *arXiv preprint arXiv:1909.10705*.

Emily Sheng, Kai-Wei Chang, Premkumar Natarajan, and Nanyun Peng. 2019. The woman worked as a babysitter: On biases in language generation. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3407–3412, Hong Kong, China. Association for Computational Linguistics.

Pradyumna Tambwekar, Murtaza Dhuliawala, Lara J Martin, Animesh Mehta, Brent Harrison, and Mark O Riedl. 2019. Controllable neural story plot generation via reward shaping. In *IJCAI*, pages 5982–5988.

Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. 2018. Graph attention networks. In *International Conference on Learning Representations*.

Su Wang, Greg Durrett, and Katrin Erk. 2020. Narrative interpolation for generating and understanding stories. *arXiv preprint arXiv:2008.07466*.

Stephen Ware and R Young. 2010. Modeling narrative conflict to generate interesting stories. In *Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*, volume 5.

Lili Yao, Nanyun Peng, Ralph Weischedel, Kevin Knight, Dongyan Zhao, and Rui Yan. 2019. Plan-and-write: Towards better automatic storytelling. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 7378–7385.

Tianyi Zhang, Varsha Kishore, Felix Wu, Kilian Q. Weinberger, and Yoav Artzi. 2020. Bertscore: Evaluating text generation with bert. In *International Conference on Learning Representations*.

Daniel M Ziegler, Nisan Stiennon, Jeffrey Wu, Tom B Brown, Alec Radford, Dario Amodei, Paul Christiano, and Geoffrey Irving. 2019. Fine-tuning language models from human preferences. *arXiv preprint arXiv:1909.08593*.

A Broader Impacts

Our approach guides the pre-trained language models to generate coherent stories that reach a prespecified goal. We focus on generating sentences with the correct verb order. Because we do not fine-tune the pre-trained language model, we do not prevent the prejudicial biases and toxicity (Sheng et al., 2019; Ousidhoum et al., 2021) in these models from surfacing in the generated stories. However, if candidate sentences that are toxic do not move a story toward its goal and the policy model may reduce toxicity as a side-effect. Because our technique works in a plug-and-play fashion with language models, we can reduce potential biases by using a language model trained to mitigate those biases, such as Peng et al. (2020). Moreover, during generation, because we select one from 25 candidate sentences each time, we can use an additional filter for toxicity toxicity or visible biases. This is a strategy increasingly used by systems such as Google's Meena (Adiwardana et al., 2020).

The pursuit of automated story generation is coupled an inherent risk associated with the passing off of fictional generated stories as non-fiction. Malicious actors may use any language modeling approach to create stories for the purpose of persuasion or misinformation. This can currently be done by prompting a language model on a topic, but without an guarantee about the direction the story takes or the end. Our technique gives users more control of how an ending occurs, which potentially gives users more say over the "message" of the story. However, verb class goals are not as precise as the prompts, and the state of the art in automated story direction is still at the stage of short, simple stories.

Our goal-reaching method has applications beyond story generation. In particular, the method can be thought of as a task planner where the elements of the plan can only be expressed indirectly though language. Aside from entertainment contexts, this is a step toward being able to interact with speech dialogue systems to brainstorm task plans—i.e., to plan a set of errands—or to coordinate actions with a robot without programming.

B GPT-RS Training Details

The steps of training for the GPT-RS model are:

1. The base language model generates a target sentence based on a query. The base model uses top-k sampling in the generation. We choose $k = 1000$ to produce better story results with less repetition in transformer models as stated in (See et al., 2019). If this target sentence has a verb with a positive difference then we move on to the next step. Otherwise, we sample a new query sentence from the training data and try again.

2. The reward model then gives the reward amount this sentence should receive according to the criteria described above.

3. To fine-tune our base language model using the reward shaping function, we follow (Ziegler et al., 2019) approach in RL fine-tuning GPT-2 using the Proximal Policy Optimization algorithm (PPO2). We utilize the source sentences, target sentences, and assigned rewards to update the base model's (policy) loss. We utilize the Transformer Reinforcement Learning library[1] which provides an implementation of PPO2 compatible with the Huggingface transformers library.[2]

C Models training details and hyperparameters

The KG-DQN and DQN models were trained for 20 epochs each, and the best model every 5 epochs was taken for evaluation. We used a batch size of 256 and replay buffer of 800 using Adam. Our hyperparameters are discount factor = 0.99, learning rate = 0.001, ϵ = 0.1, with epsilon decaying following $\epsilon = (\epsilon - 0.01)/1000$ every training step.

The GPT2-RS models were trained for 40 epochs each, and the best model every 10 epochs were taken for evaluation. We use similar settings to the ones described in (Ziegler et al., 2019) with $KL_\beta = 0.1$, $KL_{target} = 6.0$, learning rate of 7.07×10^{-8} and 4 PPO epochs at each training epoch.

The KG-DQN-RS models simply combined the already trained components from KG-DQN and GPT2-RS respectively, with no further fine-tuning.

D Computing Infrastructure

Our GPT2-RS models were trained and evaluated using Google Colab, on a Nvidia Tesla V100 GPU. KG-DQN and DQN were trained and evaluated

[1]https://github.com/lvwerra/trl

[2]https://huggingface.co/transformers/index.html

locally on two machines: the first machine had 2 Nvidia GeForce GTX 1080 GPUs, the second had four Nvidia GeForce GTX 2080Ti GPUs.

E Generated Story Samples

In Table 4 and Table 5 we show examples of some generated stories from our models.

KG-DQN-RS A Rogues scientist works in a Cavern, watched by a Rogues general and a guard. Alex and Blake finish working in ten forward. Alex makes an all-out attack on the team, killing all except Cameron. Blake dead, leaving them both shocked. Before Morgan decides to leave. B'le investigator kills Cameron. Alex turns around and heads back with Alex's investigator. Trusted, Alex follows a guard into the murder of Robin.
KG-DQN A Rogues scientist works in a Cavern, watched by a Rogues general and a guard. Alex asks what will happen when the Manbot is removed. Hedgwick believes that Blake will survive.
DQN A Rogues scientist works in a Cavern, watched by a Rogues general and a guard. Alex's name is Blake. Cameron is watching the news. Alex does not believe her and Cameron does not believe her.
GPT2-RS A Rogues scientist works in a Cavern, watched by a Rogues ran away, while Alex watches a happenance. I kept saying i kept from murdered like an fleas; you never turn life on me. Alex complains, but tries to tell her she's safe. J marketa pulls a gun on Alex. Morgan shoots at it, and then immediately throws it into the vapor. Blake notices the same thing from Cameron's office. Cameron postulates that to perfect the finisher after they work together, a normally impossible task. However Morgan base the idea one way or the other. Most especially Robin forcing Riley to put up with his section head problem. Meanwhile Cameron is having trouble depression. Alex and laranth are both having trouble believing the story.
GPT-2 Sci-Fi A Rogues scientist works in a Cavern, watched by a Rogues general and a guard. Alex watches them both die. Morgan and Blake look on. Ultimately killing each other. Bailey is taken to Cameron. Bailey is also saved in a holding Spacebarn, with the guard gone. Ellis and Morgan walk off together. Bailey is taken into custody, while Blake is taken into custody. Ellis is then taken to Hqs. Bailey is arrested, with the guard's body taken away. Ellis is then taken into custody. Blake is then taken into custody while the killer is taken into custody of Robin. Morgan is taken to a Safetech guard. Riley, under guards' knife. Jordan is then taken into custody and held for interrogation. Hqs, where Cameron is later held for questioning in a holding Spacebarn. Hqs, where Cameron is later taken to Morgan.
Goal: Admire **Prompt:** A ORGANIZATION0 scientist works in a LOCATION0, watched by a ORGANIZATION0

Table 4: Story Output Examples from each model, for the goal Admire

KG-DQN-RS: The tobacco company Hadara creates a form of super tobacco, which in turn is inhabited by a form of super tobacco beetle, which survive the cigarette processing and whose eggs are contained in the smoke of these killer cigarettes, presumably called brand Alex. Cameron, acknowledges the smoking man as being Alex. Morgan, the man confirms that he and his partner are actually doing the same thing. Bailey and Blake look down at the dead man, reply that they had better go. Alex seeing Cameron dead but doors closing.
KG-DQN: The tobacco company Hadara creates a form of super tobacco, which in turn is inhabited by a form of super tobacco beetle, which survive the cigarette processing and whose eggs are contained in the smoke of these killer cigarettes, presumably called brand Alex. Blake's research into the beetles also wasashes on the ground. Upon seeing that Blake is not who she is, Blake goes to see Blake.
DQN: The tobacco company Hadara creates a form of super tobacco, which in turn is inhabited by a form of super tobacco beetle, which survive the cigarette processing and whose eggs are contained in the smoke of these killer cigarettes, presumably called brand Alex. Cameron, in Millennia time, discovers the truth.
GPT2-RS: The tobacco company Hadara creates a form of super tobacco, which in turn is inhabited by a Alex woman Blake runs across. Witnesses a b Bothhuman carrying a boy. Went into the autopsy site while he went walking. Cameron finds blood on the wounds and goes Invanova to investigate. Cameron notes red blood evacuate from the victims as well andhead stationed by Morgan. Shortly afterwards, they find Robin heading toward the Carkoon. Cameron says Robin has answered and she's coldly blamed Robin. However, Robin insists that she and she just have a disappearances. I don't know what i'm supposed to do. I'm not supposed to die this time, Blake informs Blake's. Pleased with that, Blake job outlook. K-7i tells them that the incident was simply a result of attacks. Cameron insists that something to do with it. Androgum brushes aside an idea that prove to be impossible. Down on Robin who drops a Grenade on Morgan, but it's too late. Morgan runs over to the fire escape where Morgan finds a resources needed by those releases.
GPT-2 Sci-Fi: The tobacco company Hadara creates a form of super tobacco, which in turn is inhabited by a form of super tobacco beetle, which survive the cigarette processing and whose eggs are contained in the smoke of these killer cigarettes, presumably called brand Alex. Shortly thereafter, the beetle kills Blake, and they open fire in a room. Unfortunately, there is no response. They find a small explosion in the room, just as the beetle destroyed their factory. Cameron, alone with Jorjie, is captured by the insects. Hedril is able to escape. Blake manages to escape. Morgan is captured with Robin, Cameron's body cuffed. Riley and Jordan also find themselves both restrained. Riley and held. Jordan is taken into Jaime's chamber. Alex is taken into custody as a possible suspect. And told of the gun charges Riley brought to Alex's Homeworld's. Shelkonwa is brought to Riley, but then taken to Riley's Homeworld. Where she's questioned by an angry autopsy. Who demands to be sent before Alex's assistant, a state of mind. States that Alex is trying to see if Blake was really Blake a suspect.
Goal: Discover **Prompt:** The tobacco company LOCATION0 creates a form of super tobacco, which in turn is inhabited by a

Table 5: Story Output Examples from each model, for the goal Discover