Controllable and Editable Neural Story Plot Generation via Control-and-Edit Transformer

JIN CHEN, GUANGYI XIAO, (Member, IEEE), XU HAN, AND HAO CHEN

College of Computer Science and Electronic Engineering, Hunan University, Changsha 410082, China

Corresponding author: Guangyi Xiao (gyxiao@hnu.edu.cn)

This work was supported in part by the National Science Fund of Hunan Province under Grant 2020JJ4219.

ABSTRACT

Language-modeling-based methods for story plot generation aim to generate a plot with a language model (LM). LM methods have limitations of user-assist plot generation of goal control, refinement for editing, causing the generated plots not clear sense for specific goal, lack coherence, and edit flexible. We present a control-and-edit transformer technique which uses controlled imitation learning of editing distance from dynamic programming to support deleting policy, inserting policy, a weighting-reward with prepossess of corpus statistic, and measures continues reward for the controlled goal. Automated evaluation and Haman judgement show our method is promising in comparison with the baselines.

INDEX TERMS

Neural story plot generation, controllable, editable, edit distance.

I. INTRODUCTION

Automated plot generation as a challenging and significant natural language processing task that attracts more and more scholars to study. Given a domain or a situation with a set of specifications, automated plot generation’s role is creating a sequence of main plot points which are supposed to compliance with preset. Prior approaches to plot generation count on constructing [1]. These generated stories which approaches required extensive domain knowledge are unlively for much guiding knowledge to control the story’s coherence.

With machine learning approaches prevalently applied in NLP tasks, the story plot generator could learn storytelling in particular domain. Existing stories and plots provide lots of knowledge in particular domain and plot construction to generator, however, the story plots constructed by these methods especially neural networks tend to construct goal basing on experiences and often against the will of the user for lacking the abilities to receive guidance [2]. Applied reinforcement learning on plot generating, they invented a reward-shaping technique that adjusted the weight of the neural language model in training so that the generation of plot points could align with intended objective. In fact, the user’s goal in the generation of the plot will not be fixed. For instance, in a story generated by story plot generator, a woman hated a man in it, however, users want a romantic story so user can delete some parts of this story and insert verbs like ‘love’, ‘adore’ in it. In short, users can reverse the plot of story by edit. So, our model has the ability to edit development if plot is needed for a story plot generator.

Recently, as RNNembed in [3] presented a character-level sequence-to-sequence learning method, modern neural sequence generation models are trained either generate tokens from scratch or modify the sequence of tokens. The flexibility of current decoding models has been raised by [4] which replaced standardized decoding mechanism with two operations—deletion and insertion using imitation learning algorithms. The characteristic of these two operations is that deletion and insertion are adversarial also complementary.

The researchers are exploring open-domain story generation where a given story’s topic is as input for plot generator.
How to generate interesting stories full of twists and turns without running away from the theme is challenging these years. Seq2seq models perform not well in story generations in recent studies due to the complex and underspecified dependencies between the prompt and the story. [5] constructed large human generated stories and introduce fusion and self-attention mechanisms to generate stories human judges more like. [6] present an automated story generation technique whereby the authors decomposed the problem into the generation of successive events (event2event) and the generation of natural language sentences from events (event2sentence). [7] proposed a plan-and-write framework and compared two strategies: dynamic and static planning to generated stories which were around a given topic. The primary pitfall of neural language model approaches for story generation is that the space of stories that can be generated is huge, which in turn, implies that, in a textual story corpora, any given sentence will likely only be seen once.

Interactive story generation is the latest research hotspots for researches think a controllable story generator is humanized and so that users can control the plot development. [8] introduced content-inducing Approaches to let users have more control over the generation. [9] proposed PLOTMACHINES, a novel narrative transformer that learns to transform outlines to full stories with dynamic plot state tracking. An interesting model was proposed by [10] where users could design the emotions the protagonist experienced without sacrificing story quality.

In this paper, we take advantage of this characteristic in the previous neural story plot generation and use imitation learning to design a controllable and editable Neural Story Plot Generator with transformer’s help. During the story plot generation, when user wants to edit the plot such as revise, add, replace, or delete any part of their generated plot, the edit operation will be trained as one policy. This policy’s adversary’s output at the previous iteration will be input in next iteration. For instance, user gives the generator a specific target to the generator but replace a verb with another one in the process of generator construction. So, this operation will decompose into one deletion and one insertion. The imitation learning in transformer training is proved to be effective to give our model the ability to be editable through the following experiments. Meanwhile, our model achieves comparable results both in generator’s execution time and model’s speed of convergence.

The remainder of the paper is organized as follows. First, we discuss related work on automated story generation, editing-based models and transformers. Then in our approach we introduce the edit actions: insert & delete and our model based on control-and-edit transformer and provide an evaluation of the event representation in the context of story generation. We carry out experiments to show our model’s performance in story generation with control-and-edit ability. We end with a conclusions about these experiments and expectation on real-world application.

The main contributions of this paper can be summarized as follows:

- We first add editable capability of the story plot generator, we allow the generation process to be editable and controlled, this significantly increases user’s engagement.
- The Control-and-Edit Transformer is proposed as a reward reshaped imitation learning from the training of goal, input plot sequence and the ground truth of output plot sequence.
- We evaluate our techniques in two ways such as automatic evaluation with the goal achievement rate and the perplexity of language modeling technique, and human judgement. The results show that our model is comparable to the state-of-the-art models.

II. RELATED WORK

A. NEURAL PLOT GENERATION

Symbolic planning [1], [11]–[13], [14]–[16] and case-based reasoning [17], [18] are the main means to story plot generation in the early time. These methods rely on a combination of rigorous algorithms and domain-specific knowledge so that the coherence of story plot gets well controlled on the logic of cause and effect. The popularity of machine learning allows story line generators to learn how to tell stories from an existing library of stories. Recently, reinforcement learning is used in text processing issues, especially Markov decision process (MDP) problem has been proved to be solved efficiently. In MDP problem, a tuple $M = <S, A, T, R, y>$ where $S$ is world states, $A$ is possible actions, $T$ is a transition function. To some extend, reinforcement learning addresses some issues of preserving coherence for text generation. In addition, with MDP’s constructing, reinforcement learning provides the ability to specify a goal. [2] used deep reinforcement learning in story plot generation, they give a goal to story plot generation and used reward shaping to force the strategy gradient search to find longer term reward.

B. EDITING-BASED MODELS

Incorporating “editing” operations for sequence generation tasks has been investigated by some prior works. Using convolutional neural network, [19] predict and apply token substitutions iteratively on phase-based MT system. [20] and [21] propose oracle policy which used Levenshtein distance with dynamic programming. Levenshtein transformer which simultaneously inset and delete multiple tokens iteratively was proposed by [4].

C. IMITATION LEARNING

In traditional reinforcement learning tasks, learning the optimal policy is usually done by calculating cumulative rewards, which is straightforward and performs well when more training data is available. However, in multi-step decision, the learner cannot be rewarded frequently, and there is a huge search space based on cumulative rewards and learning. The Imitation Learning approach has been developed over the
years and can solve the multi-step decision problem well. Imitation learning has been widely applied in NLP tasks. Imitation learning means learning from the examples provided by the demonstrator, it uses states as features and uses actions as labels to obtain the optimal policy model. Thus we use imitation learning to train the Controlled Distance Transformer which empowers neural story plot generation Controlled and Editable abilities.

D. TRANSFORMER

Different from Seq2Seq [22] which inputs a word to the encoder in order in each step to generate one word output at a time in the decoder. In 2017, Breakthrough transformer framework used attention mechanism to overcome the context limitation of LSTM. Besides, it provided more throughput because the inputs were processed in parallel and there was no order dependency which solved the problem of long term dependence. The subsequent improved transformer model, such as gpt-1 [23] and bert [24] which were pre-trained in large data sets. Subsequently, transfer learning was used to fine tune these models for specific task characteristics, thus significantly improving the performance of several NLP tasks such as language modeling, sentiment analysis, question answering and natural language reasoning.

In Natural Language Generation (NLG) task, transformer also performs well. A denoising autoencoder BART [25] incorporates two-stage pre-training: (1) Corruption of original text via a random noising function, and (2) Recreation of the text via training the model. Turing-NLG trains more kinds and quantities of data in advance, and its resources are shared among different tasks to enhanced zero-shot learning. GShard [26] allows more than 600 billion parameters to be expanded. Through sparse gating and mixed multi language machine translation, it uses automatic slicing method to classify experts (MOE) with low computing cost and compilation time. In our work, we chose representative BERT as our model’s transformer. which showed nice effect in story plot generation.

III. OUR APPROACH

We take advantage of the characteristic in previous interactive story generation and use imitation learning to design a controllable and editable Neural Story Plot Generator with transformer’s help.

A. NOTATIONS AND DEFINITIONS

The plot generation problem is a planning of finding a sequence of editable events ended with a desired goal, where we using the verb (e.g., like, love, marry) of the event to represent the goal. The challenges include user-control and user-edit assist story event generation.

Specifically, we use a Markov Decision Process (MDP) of a tuple \((Y, A, E, R, g, y_0)\) to model controllable and editable plot sequence generation. The MDP has a setting of an environment \(E\) for the agent interaction with editing action and the designed goal \(g\) to generate output plot sequence. \(Y \in \mathcal{V}_{\text{max}}\) is the plot sequence generate set with a max length \(N_{\text{max}}\), and \(\mathcal{V}\) is a word vocabulary. For each decoding process, the model receives an input of the plot sequence and goal verb tuple \((y, g)\) where \(y\) is drawn from scratch or uncompleted plot sequence generation and \(g\) is the desired goal, selects an action \(a\) with a continues reward \(r\). \(A\) is the editable action set and \(R\) is the expected reward function. Usually, \(R\) is the optimal edit distance between the ground-truth plot sequence \(y^*_g\) given goal \(g\) and the plot generation sequence \(y\), \(R(y) = -D(y^*_g, y)\) by Dynamic Programming. As the initial
actions such as insertion and deletion are defined to output
As the definition of MDP model, suppose given the
π, which with condition of current plot generation and the
verb goal to a policy probability to produce the probability for
different edit actions: \( \pi(a|y, g) \).

B. EDIT ACTIONS: INSERT & DELETE
As the definition of MDP model, suppose given the
current plot sequence \( y^k = \{y_1, y_2, \ldots, y_n\} \), the edit actions such as insertion and deletion are defined to output
\( y^{k+1} = \mathcal{E}(y^k, a^{k+1}) \).

1) DELETE
The deletion policy reads the input plot sequence \( y \), and for
every token \( y_i \in y \), the deletion policy \( \pi_{\text{del}}(d|i, y, g) \) makes a
binary decision such as 1 (delete this token) or 0 (keep it).

2) INSERT
The insertion policy consists two phases: place-holder sub-
policy and token classification sub-policy which can collaboratively insert multiple words for some slots. (1) For each
slot \( y_i \) in plot sequence \( y \), \( \pi_{\text{plth}}(p|i, y, g) \) calculate the
number of tokens required by the slot. (2) for each place-
holder calculated in the slot, a word classification sub-policy
\( \pi_{\text{tok}}(t|i, y, g) \) is defined to replace the default place-holders
to the classified word in the vocabulary.

3) COMBINED EDIT POLICY
So totally three steps are combined for our controlled plot
sequence generation: delete words by deleting sub-policy -
insert default placeholders - replace the default placeholders
with work classification. More precisely, given the current
plot sequence \( y = \{y_1, y_2, \ldots, y_n\} \), and the edit action of
\( a = \{d_1, \ldots, d_{n+1}; p_1, \ldots, p_n; w_1, \ldots, w_{n+1}, w_n\} \), the combined
policy for editing is:

\[
\pi(a|y, g) = \prod_{d \in d} \pi_{\text{del}}(d|i, y, g) \\
\cdot \prod_{p \in p} \pi_{\text{plth}}(p|i, y, g) \\
\cdot \prod_{w \in w} \pi_{\text{tok}}(w|i, y, g')
\]

(1)

where with interaction of deletion action, \( y' = \mathcal{E}(y, d) \) is the
environment output; and with the interaction of place-holder,
\( y'' = \mathcal{E}(y', p) \) is the environment output by adding default
place-holder tokens.

C. CONTROL-AND-EDIT TRANSFORMER
The key model of Control-and-Edit Transformer and the dual-
policy learning algorithm are introduced in this subsec-
tion. In a simplified word, in each iteration of our model, the input
is a designed goal with either a plot sequence or an initial
empty sequence, the model using the inserting or deleting to
generate or refinement a output sequence. Until the model of
two combined policies are converge.

1) OUR MODEL
Firstly, Transformer pre-trained by BERT is defined as the
backbone block to encoding the concatenate sequence of
goal \( g \) and input plot sequence \( y_0 \). The states from the \( l \)-th
block with the first token of \( g \) are:

\[
h_0^{l+1} = \begin{cases} 
E_g + P_0, E_{y_0} + P_1, \ldots, E_{y_n} + P_{n+1} & l = 0 \\
\text{TransformerBlock}(h_0^n, \ldots, h_{n+1}^l) & l > 0
\end{cases}
\]

(2)

where word embedding \( E \in \mathbb{R}^{|V| \times d_{\text{model}}} \) and position embed-
ding \( P \in \mathbb{R}^{n_{\text{max}} \times d_{\text{model}}} \) are defined in Transformer model,
respectively, and the first token embedding is the embedding of
plot generate goal \( E_g \).

Controlled Policy Classifiers. The decoder outputs the
action with the encoder \( (h_0, \ldots, h_n) \) and designed goal \( g \)
by three controlled policy predictions:

(1) Controlled Delete Prediction: this will read the con-
trolled goal and the input sequence of each word and will predict whether to delete the word or keep the word,

\[
\pi_{\text{del}}^{d}(d|i, y, g) = \text{softmax}(h_i \cdot W_A^T), \quad i = 2, \ldots, n,
\]

(3)

where \( g \) is the generate goal and \( W_A \in \mathbb{R}^{2 \times d_{\text{model}}} \) is the
training parameter matrix, and the goal verb and the boundary
words are always kept without deleting. We start the deletion
position from the second index since the first one is the
designed goal.

(2) Controlled Placeholder Prediction: it classify the num-er of words as a \( K_{\text{max}} \) classification in each slot:

\[
\pi_{\text{plh}}^{p}(p|i, y, g) = \text{softmax}(\text{concat}(h_i, h_{i+1}) \cdot W_B^T), \quad i = 2, \ldots, n,
\]

(4)

where \( g \) is the generate goal and \( W_B \in \mathbb{R}^{(K_{\text{max}}) \times 3d_{\text{model}}} \) is the
trained parameter matrix. With the number \( (0 - K_{\text{max}}) \)
of words predicted by this policy, the correspond number of
placeholders are inserted in the slot. In our implementation,
special token \(<\text{PH}>\) is the preserved word in vocabulary to
added as default placeholder token in the generation or refine-
ment process.

(3) Controlled Word Prediction: after adding the \(<\text{PH}>\) in
the slots, the model will classify the \(<\text{PH}>\) with different
context slots to the words in the vocabulary by the vocabulary
classification:

\[
\pi_{\text{tok}}^{w}(w|i, y, g) = \text{softmax}(h_i \cdot W_C^T), \quad \forall y_i = <\text{PLH}>,
\]

(5)

where \( g \) is the generate goal and \( W_C \in \mathbb{R}^{|V| \times d_{\text{model}}} \) is the
trained parameters matrix for word classification.

In summary, the learning parameters of our model includ-
ing the parameters of the Transformer Blocks and the decoding
parameters such as \( (W_A, W_B, W_C) \).

2) CONTROLLED AND REINFORCED LEARNING
Control-and-Edit Transformer are trained with imitation
learning. Suppose there exist an expert policy \( \pi_\star \),
we trained the agent of the Control-and-Edit Transformer to learn it by imitation learning. By the dynamic programming of controlled input sequence and the ground-truth output sequence or the random word dropping of the output sequence by knowledge distillation we can get the optimal expert policy. Given the plot sequences of $y^0$, $y^*$, and the controlled goal $g$, the objective is to maximize the follow exception:

$$R(v(y^*), g) \cdot [L_{del}(y^0, y^*, g) + L_{ins}(y^0, y^*, g)]$$  \hspace{1cm} (6)

where $R(v(y^*), g)$ is the reward of the controlled goal $g$ and the verb of the ground truth plot sequence $y^*$, $L_{del}(y^0, y^*, g)$ is the controlled delete learning, and $L_{ins}(y^0, y^*, g)$ is controlled insert learning.

$a$: CONTROLLED DELETE LEARNING

The input plot sequence is either from the initial plot input sequence $y^0$ or from the insert generate sequence of the previous iteration. Formally, the controlled delete learning is defined as:

$$L_{del}(y^0, y^*, g) = \mathbb{E}_{d^\text{del} \sim d^\text{del}, y^0} \sum_{d^* \in \pi^*} \log \pi^\text{del}_\theta(d^*_i | y^0, \gamma_{del}, g)$$  \hspace{1cm} (7)

where the delete distribution $d^\text{del}$ is defined as follows:

$$d^\text{del} = \{y^0 \text{ if } \mu < \delta_1 \text{ else } \mathcal{E}(y^*, p^*, w), p^* \sim \pi^*, w \sim \pi_\theta\}$$  \hspace{1cm} (8)

where $\mu \sim \text{Uniform}[0, 1]$ is the uniform distribution, $\delta_1 \in [0, 1]$ is a mixture factor, and $y^*$ is input sequence of insert process.

$b$: CONTROLLED INSERT LEARNING

The input plot sequence is either from the delete generate sequence of the previous iteration or from the random word dropping of the ground truth of the output plot sequence $y^*$. Formally, the controlled insert learning is defined as:

$$L_{ins}(y^0, y^*, g) = \mathbb{E}_{y_{ins} \sim d_{ins}, y^0} \left[ \sum_{p^* \sim \pi^*, w^* \sim \pi^*} \log \pi^\text{ins}_\theta(p^*_i | y_{ins}, g) \\
+ \sum_{w^* \in \pi^*} \log \pi^\text{ins}_\theta(w^*_i | y_{ins}, g) \right]$$  \hspace{1cm} (9)

where $y^*_{ins}$ is the output of the inserting placeholders $p^*$ upon $y_{ins}$, the insert distribution $d^\text{ins}$ is defined as follows:

$$d^\text{ins} = \{\mathcal{E}(y^0, d^*), d^* \sim \pi^*, \text{ if } \mu < \delta_2 \text{ else } \mathcal{E}(y^*, d), d \sim \pi^\text{RND}\}$$  \hspace{1cm} (10)

where $\mu \sim \text{Uniform}[0, 1]$ is the uniform distribution, $\delta_2 \in [0, 1]$ is a mixture factor, and $p^\text{RND}$ is a random dropping of words in the ground truth sequence $y^*$.

$c$: EXPERT POLICY

The expert policy is the optimal edit distance between the initial plot input sequence and the ground truth of plot output sequence by the dynamic programming. Formally, the expert policy $\pi^*$ with the ground-truth $y^*$ is defined as:

$$a^* = \arg \min_a D(y^*, \mathcal{E}(y, a))$$  \hspace{1cm} (11)

where $D$ is the Edit distance, which can be obtained by dynamic programming.

d: CONTROLLED REWARD SHAPING

For different output plot sequence and different designed goal verb, we introduce different continues reward weight for the generating sequence pairs of the training dataset. Formally, to construct the continues weighting reward measure, we analysis the stories in the corpus by two sub-reward functions by prepossess: (1) for each ground-truth output sequence, we measure the distance of the verb in the ground-truth output sequence from the goal verb in story, and (2) for each ground-truth output sequence, we also need to count the frequency of the verbs of the output sequence in the corpus. By analysis the stories has the goal verb $g$, The statistic verb distance of $\upsilon$ and goal verb $g$ is calculated by:

$$r_1(\upsilon, g) = \log \sum_{s \in \mathcal{S}_{\upsilon, g}} l_s - d(\upsilon, g)$$  \hspace{1cm} (12)

where $\mathcal{S}_{\upsilon, g}$ is the subset of stories in the corpus that contain $\upsilon$ before the goal verb $g$, $l_s$ is the number of events in the story $s$, and $d(\upsilon, g)$ is the number of events between $\upsilon$ and $g$ in each story $s$. For each story $s$ if the event verb $\upsilon$ and $g$ are closer, the large reward is produced.

By analysis the stories has the goal verb $g$, the statistic verb frequency of $\upsilon$ and goal verb $g$ is calculated by:

$$r_2(\upsilon, g) = \log \frac{k_{\upsilon, g}}{N_{\upsilon}}$$  \hspace{1cm} (13)

where $N_{\upsilon}$ is the count of events with $\upsilon$ in the corpus, and the $k_{\upsilon, g}$ is the count of co-occur of them in the corpus with a precedent order constraint of $(\upsilon, g)$ in the story.

The totally continues reword for each verb in ground-truth output sequence is:

$$R(\upsilon, g) = \gamma \times r_1(\upsilon, g) \times r_2(\upsilon, g)$$  \hspace{1cm} (14)

where $\gamma$ is the normalization constant.

IV. EXPERIMENTS

The effectiveness and flexibility of Control-and-Edit Transformer across the neural plot generation task are evaluated in our experiments.

A. EXPERIMENTAL SETTINGS

1) DATASET

A corpus of movie plots from Wikipedia [27] was used in our experiments. We cleaned this corpus up to remove any extraneous Wikipedia syntax, such as links for which actors played which characters. The number of stories in the corpus is 42170, with an average of 14.52 sentences per story.

Not only our event translation process can extract a single event from a sentence, also we can pick up multiple events from one sentence. We picked up multiple events for there were more than one verbal in sentence or the sentence contained conjunctions. For instance, in this sentence “Henry and Jerry played basketball.”, our algorithm would
extract two events: Henry, play, basketball, and Jerry, play, basketball.

2) MODELS
In our experiments, Tensorflow is used to train the encoder-decoder network. Both the encoder and the decoder comprised of Transformer units, with a hidden layer size of 2048, word embedding size of 512, head size of 8 and layers of 6. With mini-batch gradient descent approach, this network was pre-trained for 300 epochs.

We compare with three baseline models:
- (1) Seq2Seq [28]: As our baseline, this model is considered to be a “generalized multiple sequential event2event” model.
- (2) DRL-clustered [2]: Starting from the weight of seq2seq, we continue to use the strategy gradient technique and reward function for training. At the same time, we cluster and restrict the verb positions described in the previous section, while keeping all network parameters unchanged.
- (3) DRL-unrestricted [2]: This is the same as DRL-clustered but there is no lexical restriction when sampling verbs for the next event during training.
- (4) Our model: Starting from the weight of seq2seq and using settings above.

3) EXPERIMENTAL SETUP
Every event in our data-set is a clue, we generated stories with above models. For all models, the generating process terminates in any of the following cases when: (1) the model outputs an end-of-story token; (2) an event generated with a target verb; (3) the length of the story is already 15 lines. The goal achievement rate indicates the percentage of stories containing the target verb in the total number of generated stories. In addition, we compare the average story length generated with the average story length in the test data where the target event occurred (set the length to 15 if it did not occur). Finally, we measured the confusion of all models except the test data, because it is not a model.

### TABLE 1. The three indexes’ results of three target verbs in our model.

| Verb   | Blue4 | Goal achievement rate | Average perplexity |
|--------|-------|-----------------------|--------------------|
| marry  | 0.17  | 93.75%                | 6.37               |
| command| 0.19  | 94.23%                | 5.89               |
| drilling| 0.21 | 94.76%                | 5.16               |

4) RESULTS AND DISCUSSION
We chose three verbs (marry, command and drilling) as plot generation’s goal. The detailed results of three goals are listed in Table 1. The ‘drilling’ verb performed best in blue4, goal achievement and perplexity. These show our method has the consistency effect as controllable and editable neural plot generation. The three verbs’ generation results are averaged in Table 2 for performance comparison.

The overall comparison results are showed in Table 2. 39.92% of the stories in the testing set, end in our desired goals, which shows that our chosen goals were not frequent in the corpus. Our model generated the given goals on average 94.72% of the time compared to 93.35% on average for DRL-Clustered model and 24.64% for the DRL-unrestricted model. Seq2seq is a baseline for story generation, and DRL-Clustered is a strong baseline of goal achievement rate for story generation. Our model outperforms this strong baseline by the improvement of 1.27% as well as outperforms the basic baseline by the improvement of 70.08%. Compare with the strong baseline, our model has the new characters such as controlled plot story refinement and the inverse goal verb controlling in generation since the Control-and-Edit Transformer, see the case study.

Perplexity, this standard is used to be a metric to calculate how accurate the learned distribution is for predicting unseen data. We observe that perplexity values drop substantially for the DRL models (7.05 for DRL-clustered and 9.78 for DRL-unrestricted) when compared with the Seq2Seq baseline (48.06). Our model achieve the lowest perplexity at 5.76. Our model outperforms the Strong baseline of DRL-Clustered by an improvement of 1.29 point of Perplexity. This low perplexity can be attributed to the fact that the Controlled-and-Edit transformer as well as the reward function, which is based on the distribution of verbs in the story corpus, improving the model’s ability to recreate the corpus distribution. For the reason our method’s rewards are based on subsequent verbs in the corpus instead of verb clusters, this causes a lower perplexity, as well as we can achieve the goal often.

The average story length in generation is also important since short story is trivial. However if the story length is too long, the controlled-and-edit model has bad performance for editing the goal verb in story generation. See the case study for inverse goal verb control. Even though our model’s average story length is a little short than the strong baseline of DRL-Clustered, our model can change the story generation for inverse goal verb control by editing the control verb and the story refinement from the previous positive goal verb control. This shows our model has a great trade-off of the average story length and the story generate speed for refinement of goal verb and the generated story.

### TABLE 2. The comparison in each standards.

| Model       | Goal achievement rate | Average perplexity | Average story length |
|-------------|-----------------------|--------------------|----------------------|
| Seq2Seq [28]| 39.92%                | 48.06              | 6.98                 |
| Unrestricted [2]| 24.64%                | 9.78               | 7.35                 |
| Clustered [2]| 93.35%                | 7.05               | 5.34                 |
| Our model   | 94.25%                | 5.8                | 4.89                 |

### TABLE 3. The comparison in Blue4 of models.

| Model       | Event2Event | EventToSentence |
|-------------|-------------|-----------------|
| seq2seq [28]| 0.0432      | 0.0016          |
| our model   | 0.19        | 0.032           |

In addition of Table 3, our model has better BLUE4 learning effect compared with the baseline of seq2seq model.
B. HUMAN EVALUATION

Human evaluation is supposed to be the most intuitive and convincing standard in plot generation’s evaluation. We first convert event sequences into natural language and then give generated plots to human judges. Through the concise, grammatically and semantically guaranteed artificial translation of the generated plot events, we know that artificial evaluation is to evaluate the original events, not the creativity of sentence writing.

We invited 100 participants who could give fair evaluation to generated stories. At a time, participants were given one of the translated plots, rating each of the following statements with a score range of 1–5 representing how much they agreed (the higher the score, the more agree):

1. GRAMMATICAL CORRECTNESS.
2. The sequence of the stories seems REASONABLE.
3. In line with the LOGIC before and after the story.
4. UNIQUE plot drama.
5. story’s INTERESTING.
6. Superior QUALITY.
7. This story WORTH SHARING.

This seven above standards are refer to a software which was designed to evaluate performance of generated stories [29].

![Figure 3. Average score of per model.](image)

Each participant in line with the principle of the more in line with the indicators, the higher the score. The detailed score these models achieved in 7 aspects are showed in Figure 3. As we can see, our model performs better than Seq2Seq and DRL, especially, 100 participants give a high score in local causality, this shows our model pays more attention to the logical relationship when reach the goal verb.

C. CASE STUDY

In this part, we will provide a successful and a failure case analysis for the benefit of future work propositions. In our experiments, we set a verb ‘love’ in the beginning of the plot generation, and we change the verb to ‘hate’ during the generation.

We pick a successful and a failure story to show the performances of our generator. The first story is: ‘Tom and Mary met at a party. Mary’s charming appearance and elegant temperament attracted Tom. Under Tom’s pursuit, they fell in love. However Tom went out with his friend Lucy behind Mary’s back. Once Mary’s good friend June saw him in a restaurant. June told Mary, Mary was heartbroken and hated Tom. Now she is deciding to break up with Tom.’ This story is a common sense story, and the plot develops according to the given direction, the protagonist Mary first fell in love with Tom and then hated him.

The first story is a successful one. And the other one story is: ‘Henry is riding a bicycle in the park when he hears a girl calling for help. He looks back and finds that a girl is drowning. Henry immediately stops the car, takes off his clothes and jumps into the river to save the girl. Henry raised the girl on his shoulder and successfully rescued her ashore. The girl introduced her name as Mary and found that they both graduated from the same high school. In order to thank Henry for his help, the girl left his contact information and wanted to invite him to dinner together in the future. After several dates, Mary finds herself in love with Henry. In the end, they became a sweet couple.’ This story is failure for that it is ending up with ‘love’ not ‘hate’. We analysis it and find that maybe the story is too long to generate the object verb ‘love’ and so that there are not enough words to edit the plot. This failure inspire us to study the speed of story generation.

V. CONCLUSION

In this paper, we present a novel approach (Control-and-Edit Transformer model) for controllable and editable neural story plot generation. The scope of this approach is the assumption of existing a control goal verb as the ending verb of the story. Unlike previous methods based on controllable neural plot generation, the proposed method performs the flexible use-assist neural plot generation such as remove some plots and add some plots by the plot refinement by a novel decoding of edit by imitation learning. The ablation analysis shows our proposed components are significant. In the future work, we still need to build a comprehensive use-assist neural generation tool for real-world application.

REFERENCES

[1] M. Lebowitz, “Experiments with incremental concept formation: UNIMEM,” Mach. Learn., vol. 2, no. 2, pp. 103–138, Sep. 1997, doi: 10.1007/BF00114264.
[2] P. Tambwekar, M. Dhuliawala, L. J. Martin, A. Mehta, B. Harrison, and M. O. Riedl, “Controllable neural story plot generation via reward shaping,” in Proc. 28th Int. Joint Conf. Artif. Intell. (IJCAI), S. Kraus, Ed. Macao, China, Aug. 2019, pp. 5982–5988, doi: 10.24963/ijcai.2019/829.
[3] H. Zhang, J. Li, Y. Li, and H. Yue, “Understanding subtitles by character-level sequence-to-sequence learning,” IEEE Trans. Ind. Informat., vol. 13, no. 2, pp. 616–624, Apr. 2017.
[4] J. Gu, C. Wang, and J. Zhao, “Levenshtein transformer,” in Advances in Neural Information Processing Systems, H. M. Wallach, H. Larochelle, A. Beygelzimer, F. d’Alché-Buc, E. B. Fox, and R. Garnett, Eds. Vancouver, BC, Canada, 2019, pp. 11179–11189. [Online]. Available: http://papers.nips.cc/paper/9297-levenshtein-transformer
[5] A. Fan, M. Lewis, and Y. Dauphin, “Hierarchical neural story generation,” in Proc. 56th Annu. Meeting Assoc. Comput. Linguistics, 2018, pp. 1–11.
[6] L. J. Martin, P. Ammanabrolu, X. Wang, W. Hancock, S. Singh, B. Harrison, and M. O. Riedl, “Event representations for automated story generation with deep neural nets,” in Proc. AAAI Conf. Artif. Intell., 2017, pp. 1–8.
[7] L. Yao, N. Peng, W. Weischedel, K. Knight, D. Zhao, and R. Yan, “Plan-and-write: Towards better automatic storytelling,” in Proc. AAAI Conf. Artif. Intell., 2018, pp. 1–8.

[8] F. Brahman, A. Petrusca, and S. Chaturvedi, “Cue me in: Content-inducing approaches to interactive story generation,” in Proc. 1st Conf. Asia-Pacific Assoc. Comput. Linguistics, Conf. Empirical Methods Natural Lang. Process. Suzhou, China: Association for Computational Linguistics, Dec. 2020, pp. 588–597. [Online]. Available: https://aclanthology.org/2020.aacl-main.59

[9] H. Rashkin, A. Celikyilmaz, Y. Choi, and J. Gao, “PlotMachines: Outline-conditioned generation with dynamic plot state tracking,” in Proc. Conf. Empirical Methods Natural Lang. Process. (EMNLP), 2020, pp. 1–22.

[10] L. Yao, N. Peng, R. Weischedel, K. Knight, D. Zhao, and R. Yan, “Planning complex stories: Towards better automatic storytelling,” in Proc. Conf. Empirical Methods Natural Lang. Process. (EMNLP), 2020, pp. 1–19.

[11] R. M. Meehan, “Tale-spin, an interactive program that writes stories,” in Proc. 5th Int. Joint Conf. Artif. Intell., R. Reddy, Ed. Cambridge, MA, USA: William Kaufmann, Aug. 1977, pp. 91–98. [Online]. Available: http://ijcai.org/Proceedings/77-1/Papers/013.pdf

[12] G. A. Miller, “WordNet: A lexical database for English,” Commun. ACM, vol. 38, no. 11, pp. 39–41, Nov. 1995. doi: 10.1145/219717.219748.

[13] J. J. Mendel, “Tale-spin, an interactive program that writes stories,” in Proc. 5th Int. Joint Conf. Artif. Intell., R. Reddy, Ed. Cambridge, MA, USA: William Kaufmann, Aug. 1977, pp. 91–98. [Online]. Available: http://ijcai.org/Proceedings/77-1/Papers/013.pdf

[14] J. J. Porteous and M. Cavazza, “Controlling narrative generation with planning trajectories: The role of constraints,” in Interactive Storytelling, Second Joint International Conference on Interactive Digital Storytelling (Lecture Notes in Computer Science), I. Iurgel, N. Zagalo, and P. Petta, Eds., vol. 5915. Guimarães, Portugal: Springer, Dec. 2009, pp. 234–245, doi: 10.1007/978-3-642-10643-9_28.

[15] M. O. Riedl and R. M. Young, “Narrative planning: Balancing plot and character,” J. Artif. Intell. Res., vol. 39, pp. 217–268, Sep. 2010, doi: 10.1613/jair.2989.

[16] S. G. Ware and R. M. Young, “CPOCL: A narrative planner supporting conflict,” in Proc. 7th AAAI Conf. Artif. Intell. Interact. Digit. Entertainment (AIIDE), V. Bulitko and M. O. Riedl, Eds. Stanford, CA, USA: The AAAI Press, Oct. 2011. [Online]. Available: https://www.aaai.org/ocs/index.php/AIIDE/AIIDE11/paper/view/4058

[17] R. P. Y. Pérez and M. Sharplees, “MEXICA: A computer model of a cognitive account of creative writing,” J. Exp. Theor. Artif. Intell., vol. 13, no. 2, pp. 131–131, Apr. 2001, doi: 10.1080/09528130010029820.

[18] P. Gervás, B. Díaz-Agudo, F. Peinado, and R. Hervás, “Story plot generation based on CBR,” Knowl.-Based Syst., vol. 18, nos. 4–5, pp. 235–242, Aug. 2005, doi: 10.1016/j.knosys.2004.10.011.

[19] R. Novak, M. Auli, and D. Grangier, “Iterative refinement for machine translation,” 2016, arXiv:1610.06602. [Online]. Available: https://arxiv.org/abs/1610.06602

[20] S. Sabour, W. Chan, and M. Norouzi, “Optimal completion distillation for sequence pre-training for natural language generation, translation, and comprehension,” in Proc. 58th Annu. Meeting Assoc. Comput. Linguistics, (ACL), J. Jurafsky, J. Chai, N. Schuler, and J. R. Tetreault, Eds., Jul. 2020, pp. 7871–7880, doi: 10.18653/v1/2020.acl-main.703.

[21] S. Thomson, B. O’Connor, J. Flanigan, D. Bamman, J. Dodge, S. Wayne, S. Schneider, C. Dyer, and N. A. Smith, “CMU Arc-factor, discriminative semantic dependency parsing,” in Proc. 8th Int. Workshop Semantic Eval., P. Nakov and T. Zesch, Eds. Dublin, Ireland, Aug. 2014, pp. 176–180, doi: 10.3115/v1/i14-2027.

[22] I. Sutskever, O. Vinyals, and Q. V. Le, “Sequence to sequence learning,” in Proceedings of the 28th International Conference on Machine Learning (ICML-11), 2011, pp. 1146–1153.

[23] A. Radford, K. Narasimhan, T. Salimans, and I. Sutskever, “Improving language understanding by generative pre-training,” OpenAI, Tech. Rep., 2018.

[24] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, “BERT: Pre-training of deep bidirectional transformers for language understanding,” 2018, arXiv:1810.04805. [Online]. Available: https://arxiv.org/abs/1810.04805

[25] M. Lewis, Y. Liu, N. Goyal, M. Ghazvininejad, A. Mohamed, O. Levy, V. Stoyanov, and L. Zettlemoyer, “BART: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension,” in Proc. 58th Annu. Meeting Assoc. Comput. Linguistics, (ACL), J. Jurafsky, J. Chai, N. Schuler, and J. R. Tetreault, Eds., Jul. 2020, pp. 7871–7880, doi: 10.18653/v1/2020.acl-main.703.