MEGATRON-CNTRL: Controllable Story Generation with External Knowledge Using Large-Scale Language Models

Peng Xu[‡*], **Mostofa Patwary**[§], **Mohammad Shoeybi**[§], **Raul Puri**[§],
Pascale Fung[‡], **Anima Anandkumar**[§], **Bryan Catanzaro**[§]
[‡]The Hong Kong University of Science and Technology, [§]NVIDIA
peng.xu@connect.ust.hk, mpatwary@nvidia.com

Abstract

Existing pre-trained large language models have shown unparalleled generative capabilities. However, they are not controllable. In this paper, we propose MEGATRON-CNTRL, a novel framework that uses large-scale language models and adds control to text generation by incorporating an external knowledge base. Our framework consists of a keyword predictor, a knowledge retriever, a contextual knowledge ranker, and a conditional text generator. As we do not have access to ground-truth supervision for the knowledge ranker, we make use of weak supervision from sentence embedding. The empirical results show that our model generates more fluent, consistent, and coherent stories with less repetition and higher diversity compared to prior work on the ROC story dataset. We showcase the controllability of our model by replacing the keywords used to generate stories and re-running the generation process. Human evaluation results show that 77.5% of these stories are successfully controlled by the new keywords. Furthermore, by scaling our model from 124 million to 8.3 billion parameters we demonstrate that larger models improve both the quality of generation (from 74.5% to 93.0% for consistency) and controllability (from 77.5% to 91.5%).

1 Introduction

Text generation has recently attracted significant attention from the research community as large pre-trained language models, such as GPT-2 (Radford et al., 2018, 2019) demonstrated promising results for generating long, grammatically correct, and fluent text. Finetuning these models has shown significant improvements in downstream tasks, such as persona chat (Wolf et al., 2019). However, one non-negligible drawback of these large models is the lack of knowledge which humans use to produce natural text. For example, GPT-2 based models produce degraded generations that are illogical and ungrammatical for knowledge-driven generation tasks, such as story generation. Guan et al. (2020) therefore introduced commonsense knowledge to the pre-trained language model by further finetuning on commonsense datasets. Although implicit encoding of knowledge is helpful for knowledge incorporation, there is still a lack of training mechanism to teach the model when and what to incorporate from external knowledge.

Context	[FEMALE] was on a long road trip .
Guan et al. (2020)	she had just driven into town . she had been driving for hours . she was so tired she decided to take a nap . when she woke up , she still had n't slept well .
MEGATRON-CNTRL-124M keyword: driving	she was on her way to see a strange country. she had never seen anything like that before. finally she saw a small town that was so small. it was a great little town with a nice park !
MEGATRON-CNTRL-8B keyword: driving	she was driving in the middle of [PLACE]. all of a sudden the tire pressure light came on. she got out to check her tire. it was flat so she used the roadside assistance.
MEGATRON-CNTRL-8B-ANT keyword: attract	she really wanted to see a few attractions. the first one she saw was a giant water park. it was amazing. it ended up being a fun experience.

Table 1: Stories generated by models of increasing capacity and controllability. As the model size grows, story quality becomes increasingly coherent, fluent, and logically consistent. The last row demonstrates how MEGATRON-CNTRL-8B-ANT model controls the story generation with a new keyword, "attract". Note that [MALE] and [FEMALE] denote names and [PLACE] denotes locations.

In addition, these large pre-trained language models are hard to control. Recently, plug-and-play language models Dathathri et al. (2019) addressed whole document controllability by adding a linear classifier on top of GPT-2 to predict whether generated text observes a particular style or property. Keskar et al. (2019) controlled a 1.2B parameter language model generation via the use of

[*]This work was done during the internship of Peng Xu at NVIDIA. Corresponding authors: Peng Xu, Mostofa Patwary.

Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing, pages 2831–2845,
November 16–20, 2020.

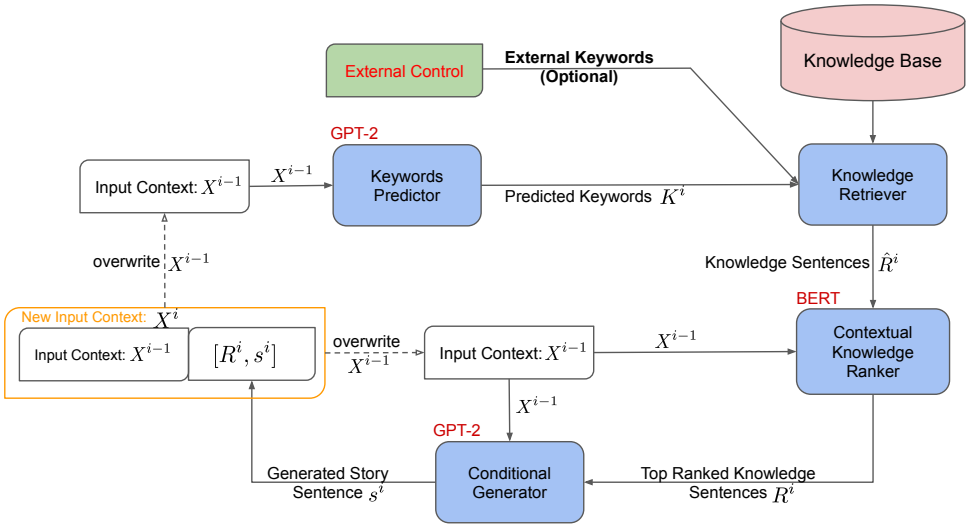


Figure 1: Overview of our generation process. Based on an input context, we generate keywords for future context, use the keywords to retrieve the relevant knowledge from an external knowledge-base, filter them based on their relevance to the context, and use the top scored knowledge sentences to guide the generation.

control codes prepended to the model input. Boyd et al. (2020) controlled the personality of a dialogue agent by conditioning it on prior conversations of a target actor. However, these controlling conditions are predefined, limited in their capability, and are only used once at the beginning to condition the generation of the rest of the document. They do not provide control granularity at either a sentence or sub-document level.

In this work, we address these shortcomings and develop an efficient controllable text generation framework that we apply to the story generation task. In order to provide manual control to users through a set of interpretable keywords, we first develop a keyword predictor model for the next sentence. These keywords are then used to retrieve knowledge sentences from an external knowledge base. Not all the retrieved knowledge is relevant to the story context and often it is noisy. To this end, we introduce a novel contextual ranker that ranks knowledge sentences based on the relevance to the context. As we do not have access to ground-truth supervision for this contextual knowledge ranker, we make use of sentence embedding for weak supervision. The top-ranked knowledge sentences from the knowledge ranker are then fed to the conditional text generator to guide generation. By giving the knowledge in addition to the context, we provide rich information for the generator to attend to and help the model better understand the rationale between sentences. Table 1 shows an example of controllable story generation with increasing model capacity.

Summary of Contributions:

- We propose a novel generation framework that allows dynamical incorporation of external knowledge into language model as well as control for text generation.
- Using both automatic metrics as well as human evaluations, we demonstrate that our model generates more fluent, consistent, and coherent stories with lower repetition rate and higher diversities compared to the previous state-of-the-art on ROC story datasets (Mostafazadeh et al., 2016).
- We showcase the controllability of our model by replacing the keywords used to generate stories. Human evaluation results show that up to 91.5% of the generated stories are successfully controlled by the new keywords .
- We scale our model from 124 million to 8.3 billion parameters and demonstrate that both qualities, as well as controllability of the generations, improve as the model size increases.

2 Framework

In our problem setup, we complete a story using the first sentence as input, similar to Guan et al. (2020). We augment the generation process with an external knowledge-base and develop a methodology that can guide and control the story generation. Our approach consists of the following four steps connected together as shown in Figure 1:

1. Given the story context, a keyword predictor model first predicts a set of keywords for the next sentence yet to be generated.

2. A knowledge retriever then takes the generated keywords and queries an external knowledge-base where each knowledge triple is converted into natural language "knowledge sentences" using templates.

3. A contextual knowledge ranker then ranks the external knowledge sentences based on their relevance to the story context.

4. Finally, a generator takes both the story context as well as the top-ranked knowledge sentences as input and generates the next sentence in the story. The output sentence is appended to the story context and steps 1-4 are repeated.

This formulation naturally allows controllability by replacing the keyword prediction process with manual external keywords. This work uses dynamic planning of the keywords and knowledge at each generation step. This allows the users to participate and control the generation on the go. As a result, they don't need to pre-specify the keywords explicitly. We also note that it is challenging to statically plan all the knowledge needed for generation at the beginning. This issue becomes severe for long generations. To formalize this method, we start by introducing notation used throughout the paper and then detail each aforementioned four steps in the following subsections.

Notation: A knowledge-base, G is defined as a set of knowledge triples $t = (\text{subject}, \text{relation}, \text{object})$. A knowledge sentence, r is defined as $r = T(t)$ by mapping t using predefined templates T. For example, *(eiffel tower, AtLocation, paris)* is transformed into *eiffel tower is at paris*. We should highlight that since our framework transforms the triple knowledge database into natural language sentences, any knowledge base in natural language format can be readily incorporated into our framework. We use superscripts to index *story sentences* and define a story S of length l as a sequence of individual story sentences s^i where $S = \{s^1, s^2, \cdots, s^l\}$. We use $K^i = \{k^i_1, \cdots, k^i_q\}$ to denote the keywords associated with story sentence s^i. A keyword k^i_q is made up of subword tokens from our language model's vocabulary. Note that the number of keywords q per sentence varies and can be zero. We define $R^i = \{r^i_1, \cdots, r^i_v\}$ as the knowledge associated with s^i, where r^i_j denotes the j-th *knowledge sentence* associated s^i. The number of knowledge sentences v varies per sentence and can be zero. Note that $v \neq q$ because a keyword can have multiple knowledge triples associated with it. Given this notation, we define the story context $X^i = \{x^1, \cdots, x^i\}$ where $x^i = [R^i, s^i]$. The goal of this work is to generate x^i given X^{i-1}, that is to first predict the knowledge R^i contained in s^i and then predict s^i itself.

2.1 Keyword Predictor Model

To provide manual control to users, we first develop a keyword predictor model. Given the current story context X^{i-1}, the model predicts a set of keywords K^i for the next sentence yet to be generated. The prediction of keywords instead of directly predicting knowledge triples not only allows us to control the generation in an interpretable manner, but it also helps to greatly reduce the search space for the knowledge triples. We formulate this keyword prediction problem similar to a left-to-right language model where the goal is to predict the string of concatenated keywords:

$$p(K^i|X^{i-1}) = \prod_{j=1}^{q} p(k^i_j|X^{i-1}, K^i_{<j}), \quad (1)$$

where $K_{<j}$ denotes all the predicted keywords up to the jth keyword and p is the probability distribution. We use a GPT-2 (Radford et al., 2019) transformer to model this probability distribution. We optimize the keyword predictor with maximum likelihood training and a next token prediction loss. Following Yao et al. (2019), we provide the labels for K^i by extracting keywords from a ground truth training sentence s^i using the RAKE algorithm (Rose et al., 2010) to train our keyword predictor. Note that our model allows generation of multiple keywords and thus provides the flexibility to choose a subset of them as the control signal to fit in the generation.

2.2 Knowledge Retrieval

In this step, we use the generated keywords K^i in Section 2.1 and retrieve all the related knowledge triples from our knowledge base G. This is simply done by converting all knowledge triples into knowledge sentences using predefined templates and then matching keywords against the knowledge sentences. This results in the knowledge set $\hat{R}^i = \{\hat{r}^i_1, \cdots, \hat{r}^i_z\}$ with size z. Future work will

focus on replacing this simple retrieval with a learnable module similar to Guu et al. (2020).

Algorithm 1 Building Pseudo Label of R^i

Input: Story sentence s^i and its preceding sentence s^{i-1}, USE encoder U, RAKE keywords extractor, and knowledge base G
Output: Pseudo Label of R^i

1: Extract keywords K^i from s^i using RAKE
2: Find $\bar{R} = \{T(t)|t \in G \text{ and } \exists k^i_j \in K^i, \text{s.t. } k^i_j \in t\}$
3: Encode each $\bar{r}_j \in \bar{R}$ to U^r_j using USE
4: Encode $[s_{i-1}, s_i]$ to U^s
5: Compute cosine similarity $score$ between each U^r_j and U^s
6: **return** $\bar{r}_j$s with the top N highest $score$

2.3 Building Pseudo Label of R^i

The main challenge for controlling generation with knowledge is that we have no explicit access to the hidden, latent controlling knowledge humans use to supervise their story writing. That means R^i, the knowledge associated with s^i is not available. We, therefore, propose to use a weakly supervised signal to build the pseudo labels of R^i from s^i. We hypothesize that R^i should 1) overlap with s^i in terms of keywords and 2) have strong connections to both the preceding sentence s^{i-1} and s^i. We include s^{i-1} along with s^i because it is hard to retrieve appropriate knowledge using only s^i due to the ambiguity of natural language. We also did not include other previous context beyond s^{i-1} as additional context overwhelms the information contained in s^i.

Following our hypothesis, we first extract keywords K^i from s^i using RAKE (Rose et al., 2010) and then match K^i with all knowledge triples in G. Transforming the retrieved triples into knowledge sentences gives us our set of $\bar{R}^i$. We then take the sentence s^i and s^{i-1}, concatenate them, and encode them using the Universal Sentence Encoder (USE) (Cer et al., 2018), a widely-used toolkit for semantic similarity, $U^s = U([s^{i-1}, s^i])$, where we denote the encoder of USE as U. For each $\bar{r}^i_j \in \bar{R}^i$, we then calculate the cosine similarity between U^s and $U^r_j = U(\bar{r}_j)$ and sort $\bar{R}^i$ based on this score. We take the top N highest scores $\bar{r}^i_j$ as a pseudo label of R^i. Algorithm 1 describes this process. During the training phase of each following model, we use this pseudo label of R^i to represent R^i.

2.4 Contextual Knowledge Ranker

While knowledge retrieval with keywords reduces the controlling knowledge candidate space from the knowledge base G to the subset $\hat{R}^i$, this set is still large and noisy since words are ambiguous and can have multiple senses. We, therefore, contextualize the knowledge sentences in $\hat{R}^i$ to obtain relevant and useful ones under X^{i-1}. To do this, we develop a contextual knowledge ranker. The model is trained with pseudo-labels extracted with access to the future sentence s^i as described in Sec. 2.3.

We use a BERT model to encode both the context X^{i-1} and each knowledge sentence $\hat{r}^i_j \in \hat{R}^i$. To adapt to the format of BERT, we append a [SEP] token to each R^j and s^j inside the context X^{i-1}. A [CLS] token is then added to the beginning of X^{i-1}. For segment ids, we mark the tokens from the knowledge base as 0 and those from the story as 1. The representation of X^{i-1} and $\hat{r}^i_j$ are then obtained after applying a linear layer on top of the embedding of the [CLS] token:

$$V_x = W_1 \ \text{BERT}_{\text{CLS}}(X^{i-1}),$$
$$V_j = W_2 \ \text{BERT}_{\text{CLS}}(\hat{r}^i_j),$$

where W_1 and W_2 are learnable weights. We then calculate the relevance *score* C between X^{i-1} and $\hat{r}^i_j$ using the inner product between V_x and V_j as :

$$C^i_j = C(X^{i-1}, \hat{r}^i_j) = V_x V_j \quad (2)$$

We take R^i (Sec. 2.3) as positive samples and $\hat{R}^i \backslash R^i$ as negative samples to train our ranker. Given a positive and a negative knowledge sentence r_p and r_n, we define the ranking loss as

$$L = \max\{0, M - C(X^{i-1}, r_p) + C(X^{i-1}, r_n)\} \quad (3)$$

where M is a margin and determined empirically. Algorithms 2 describe the ranker training process.

At inference time, we simply calculate C_j for all $\hat{r}^i_j \in \hat{R}^i$, sort them based on C^i_j score and pick the top N most relevant knowledge sentences as R^i.

2.5 Conditional Generator

The conditional generator is a language model that incorporates the controlling knowledge and generates the following sentences. It concatenates the story context X^{i-1} and controlling knowledge R^i as input and generates s^i. A GPT-2 transformer is used to model this conditional probability distribution. We describe the concatenated input representation in the Appendix A.5.

Algorithm 2 Knowledge Ranker Training

Parameters: BERT model parameters Θ and ranker model parameters W_1 and W_2
Input: A story S^l with l sentences and a knowledge base G

Initialize Θ using a pre-trained BERT model and W_1, W_2 randomly.
Dataset $D = \emptyset$
Call Algorithm 1 to retrieve R^1 from G using s^1.
for $i \in \{2, \ldots, l\}$ **do**
    Call Algorithm 1 to retrieve R^i using s^i.
    Get $\hat{R}^i$ using knowledge retrieval (Section 2.2)
    for $j \in \{1, \ldots, N\}$ **do**
        Sample r_p from R^i and r_n from $\hat{R}^i \backslash R^i$
        $D = D \cup (X^{i-1}, r_p, r_n)$
    end for
end for
for $(X, r_p, r_n) \in D$ **do**
    Calculate loss L using Equation 3
    Optimize BERT, W_1, W_2
end for
return BERT, W_1, W_2

3 Experimental Setup

3.1 Datasets

We use the ROC story dataset (Mostafazadeh et al., 2016) for our experiments. It consists of 98,161 stories, where each story contains five sentences. 88,344/4,908/4,909 stories are used for train/validation/test sets, respectively. Following Guan et al. (2020), for each sentence, delexicalization is performed by replacing all the names and entities in stories with special placeholders, [*MALE*], [*FEMALE*], and [*NEUTRAL*] for male, female and unknown names and entities, respectively. Given the first sentence of each story, our model's task is to generate the rest of the story. For our external knowledge base, we use ConceptNet (Speer and Havasi, 2012), consists of 600k knowledge triples.

3.2 Models

We used Megatron-LM (Shoeybi et al., 2019) for pre-trained BERT and GPT-2 models to initialize our contextual knowledge ranker and generative models, respectively. For the model configurations, hidden size, number of layers, and attention heads, we used the configurations of BERT and GPT-2 as in Megatron-LM. For generation with our GPT-2 models, we used a top-k sampling scheme (Fan et al., 2018) with $k = 40$ and a softmax temperature of 0.7. We detail the training hyperparameters and the input representations for GPT-2 and BERT in Appendix A.1 & A.2 . Both the keyword predictor and the conditional sentence generator follow the same settings.

To train our contextual knowledge ranker, we set the margin to 5.0. We set the number of knowledge sentences in R^i to 10. Therefore, for a given story context, the top 10 retrieved knowledge sentences from ConceptNet according to USE are chosen as the positive samples. We further select 40 negative samples to compute our margin loss. We then randomly sample 50 (positive, negative) pairs for each story context to train our contextual knowledge ranker. In total, we used $\sim$15 million pairs for training and $\sim$1 million pairs for validation. After training our ranker, we achieve a validation accuracy of 0.9.

3.3 Controllability Experiment Setup

To test the controllability of our model, we perform experiments where we change keywords to their antonyms. With antonyms, we expect maximal change to the story generation. To do that, we first use MEGATRON-CNTRL-124M to generate keywords K and corresponding full story S. Then we identify the first keyword $k_a^i \in K^i$ from K whose antonym is available at WordNet (Miller, 1995). If multiple antonyms for k_a^i are available we sample one with a uniform random probability. Afterwards, we provide the start of story $\{s^1, s^2, \cdots, s^{i-1}\}$, the keywords shared with our original story $\{K^1, K^2, \cdots, K^{i-1}\}$, and the antonym of k_a^i to either MEGATRON-CNTRL-124M or larger models (e.g. MEGATRON-CNTRL-355M). We then let the model finish the generation. We refer to these generations as MEGATRON-CNTRL-ANT, for example, we call the antonym generations from MEGATRON-CNTRL-355M model as MEGATRON-CNTRL-355M-ANT.

3.4 Baselines

We compare our model with the following state-of-the-art story generation models. (1) **Plan and write (Yao et al., 2019):** The authors use an LSTM-based model to first generate a sequence of keywords for planning the story. These keywords are then used to condition the generation. (2) **Knowledge enhanced GPT-2 (Guan et al., 2020):** This work is currently the SOTA for ROC story generation. It finetunes a pre-trained GPT-2 model with knowledge triples from commonsense datasets. Similar to our method, the knowledge triples are converted to sentences with templates. A multitask learning framework is then developed to further finetune the story generation task and

classify corrupted stories from real ones. We do not compare to Fan et al. (2019) because Guan et al. (2020) has already shown their model significantly outperforms Fan et al. (2019) and in this work, we compare to Guan et al. (2020). (3) **GPT-2-124M:** This baseline finetunes a GPT-2 model with a next token prediction loss on the story.

3.5 Evaluation

We conduct both automatic as well as human evaluations to assess our generation.

3.5.1 Automatic Evaluation

We use the following metrics to compare different models: **Repeat:** measures the redundancy of the generated story by reporting the percentage of the stories that contain at least one repeated 4-gram (Shao et al., 2019). **Distinct:** measures the diversity of generated stories by reporting the ratio between distinct 4-grams to all generated 4-grams. **Perplexity:** In the inference phase, our models involve two steps of generation: (i) generate set of knowledge sentences, R^i from story context X^{i-1}, (ii) generate story sentence, s^i from X^{i-1} and R^i. To report the perplexity of the conditional generator we sample R^i sequentially before generating each story sentence s^i and report the total perplexity of all sentences s^i for $i \in [2, l]$ where l is the number of sentences in the story.

3.5.2 Human Evaluation on Quality

We conduct human evaluations on Amazon Mechanical Turk[1] (AMT) to analyze the quality of our generations on three aspects: **Fluency**, **Coherence**, and **Consistency**. To evaluate fluency, we show the annotators a pair of generated stories from two models. We ask them to evaluate each sentence independently and choose the story with better overall fluency. Fluency of a story is defined as a measure of intra-sentence linguistic quality and grammatical correctness taken over all sentences of the story. For coherence, we provide the same stories as in fluency but ask to choose the one with better inter-sentence causal and temporal dependencies. We let the annotators choose *tie* for both fluency and coherence.

Different from the settings of fluency and coherence, we only show one generated story to annotators to evaluate consistency. They are required to choose whether the story is logically consistent, based on whether the story self contradicts or not.

[1]https://www.mturk.com/

We set up these three evaluations as independent AMT tasks to make sure the tasks do not influence each other and introduce spurious correlations between labels. To reduce noise in our labeling process, we only accepted workers with an approval rating over 90% and have over 1k accepted jobs. We further limited the location of the annotators to the United States. For each example, we explicitly ask them to spend at least 15 seconds to evaluate coherency and 10 seconds to evaluate the other two properties. In total, we randomly sample 200 stories and assign five annotators for each story. We adopted majority voting to make final decisions among the five annotators.

3.5.3 Human Evaluation on Controllability

To evaluate how controllable our model is, we conduct another human evaluation just for controllability. We show the annotators the start of a story, original keywords, and the corresponding generation. We then show the antonyms of the keywords, along with the corresponding generated story, and ask the annotators if the new story has changed compared to the original story in accordance with the meaning of the keyword's antonyms. The rest of the AMT settings for these experiments are the same as our consistency experiments.

4 Results

In this section, we first perform automatic and human evaluations with different model sizes and compare our framework to the existing baselines. We then evaluate the controllability of our model and finally show ablation study varying GPT-2 and BERT model sizes. The detailed configuration of the model sizes are shown in Table 2. We provide several generated stories in Appendix A.7 varying the length of the given context. We use M-CNTRL to denote MEGATRON-CNTRL in the tables due to the limited space.

Model Name	Conditional Generator (GPT-2)	Keyword Generator (GPT-2)	Knowledge Ranker (BERT)
M-CNTRL-124M	124M	124M	336M
M-CNTRL-355M	355M	355M	336M
M-CNTRL-774M	774M	774M	336M
M-CNTRL-2B	2.5B	2.5B	336M
M-CNTRL-8B	8.3B	2.5B	336M

Table 2: Number of parameters of our models (M-CNTRL is the short form for MEGATRON-CNTRL).

Source A	Coherence ↑	Fluency ↑	Source B
M-CNTRL-124M	**78.5%** - 13.0%	**66.5%** - 22.5%	Yao et al. (2018)
M-CNTRL-124M	**46.0%** - 39.0%	**44.5%** - 43.5%	Guan et al. (2020)
M-CNTRL-355M	**56.0%** - 30.5%	**46.5%** - 30.5%	Guan et al. (2020)
M-CNTRL-355M	**52.0%** - 31.5%	**46.5%** - 39.0%	M-CNTRL-124M
M-CNTRL-774M	**44.5%** - 41.5%	**56.0%** - 33.5%	M-CNTRL-355M
M-CNTRL-2B	**50.5%** - 30.5%	**53.0%** - 39.0%	M-CNTRL-774M
M-CNTRL-8B	**46.0%** - 39.5%	46.5% - 46.5%	M-CNTRL-2B

Table 3: Pairwise comparison between our models and baselines. Percentages in the format "A% - B%" indicate how often annotators rank the samples from source A better than from source B for a given category, and vice versa. Percentage pairs do not sum to 100% as the annotators were allowed to choose "tie" as being of equal quality. MEGATRON-CNTRL-124M achieves better results than all baselines. Scaling the models shows better coherence and fluency.

Name	PPL ↓	Repeat ↓	Distinct ↑	Consistency ↑ (Human Eval)
GPT-2-124M	6.98	27.2	74.1	69.5
Yao et al. (2018)	NA	**13.3**	63.7	49.0
Guan et al. (2020)	7.04	22.1	77.1	67.0
M-CNTRL-124M	9.37	20.0	80.1	74.5
M-CNTRL-355M	8.02	19.9	81.6	75.5
M-CNTRL-774M	6.58	21.3	81.6	80.5
M-CNTRL-2B	6.31	21.2	82.6	89.0
M-CNTRL-8B	**6.21**	21.2	**82.8**	**93.0**

Table 4: Evaluation results for the previous state-of-the-art models as well as our algorithm at different sizes. Perplexity, repeat, and distinct are evaluated automatically whereas consistency is obtained using human evaluations. Our smallest model with 124M parameters achieves better distinct and consistency score compared to prior work. Increasing model size up to 8B improves perplexity, distinct, and consistency scores. For reference, the ground truth human writing gives 7.6 score for repeat and 88.9 for distinct.

4.1 Automatic and Human Evaluations

Table 4 shows that our smallest model, MEGATRON-CNTRL-124M achieves better distinct and consistency scores compared to previous work. For repetition, our model is worse than Yao et al. (2019) which was also observed in Guan et al. (2020). The reason could be their small 8M model is better at learning short term statistics (e.g. 4-grams), while large models are better at learning long term dependencies. Compared to other GPT-2 based models (i.e. GPT-2-124M and Guan et al. (2020)), MEGATRON-CNTRL-124M achieves lower repeat and higher distinct scores, hence our model generates less repetitive stories.

We notice from Table 4 that our perplexity (PPL) score is much higher than other GPT-2-based models. Our hypothesis for why this occurs is that other GPT-2-based methods directly model and report $P(s^i|s^1, s^2, \cdots, s^{i-1})$ while our conditional generator models and reports $P(s^i|X^{i-1}, R^i)$. When computing perplexity, $[s^1, s^2, \cdots, s^{i-1}]$ are given ground truth tokens, but R^i and all R in X^{i-1} must be sampled from a distribution that is learned with weak supervision. This sampling introduces noise and non-determinism that results in higher perplexity. This discrepancy is not an issue when analyzing automatic evaluation metrics within our model family. When scaling our model from 124M up to 8B parameters we see a consistent drop in PPL and increase in distinct. This shows larger models can generate better stories with more diversity.

Human evaluation results are presented in last column of Table 4 (consistency) and in Table 3. Comparing MEGATRON-CNTRL-124M to Yao et al. (2019), we achieve much better coherence, fluency, and consistency scores, which shows the benefit of large pre-trained transformer models. Comparing MEGATRON-CNTRL-124M to Guan et al. (2020) which uses a similar base model, we find that fluency is similar, however we should note that Guan et al. (2020) is not controllable and our model has significantly better coherence (+7.0%) in Table 3 and consistency (+7.5%) in Table 4. We attribute this to the use of the retrieved knowledge, R^i. By explicitly providing facts pertinent to the next sentence, the conditional generative model can focus on just generating text. By comparison, a standard autoregressive GPT-2 model is tasked with predicting both the topics and the text of the next sentence.

Scaling this up, and comparing MEGATRON-CNTRL-355M to Guan et al. (2020), our model significantly outperforms in all aspects. Furthermore, a thorough comparison among MEGATRON-CNTRL-355M, MEGATRON-CNTRL-774M, MEGATRON-CNTRL-2B, MEGATRON-CNTRL-8B shows that scaling the model size further almost always improves the quality of genera-

tion in terms of fluency, coherence and consistency. For consistency, our best model at 8B parameters achieves a score of 93%.

4.2 Controllability Evaluation

We evaluate the controllability by changing keywords to their antonyms as detailed in Section 3.3 & 3.5. Table 5 shows repeat and distinct for MEGATRON-CNTRL-124M as well as the controlled versions at three different sizes. Altering control with antonym keywords gives lower repeat and higher distinct scores than the original generation. As the model size increases, the repeat stays almost constant while distinct improves. These results show that changing keywords manually results in distinct and not repeated text.

Name	Repeat ↓	Distinct ↑
M-CNTRL-124M	20.0	80.1
M-CNTRL-124M-ANT	17.8	80.9
M-CNTRL-355M-ANT	18.0	81.6
M-CNTRL-8B-ANT	18.5	82.8

Table 5: Comparing controllability of the models by changing the keywords to their antonyms. Controlled generations show less repetition and higher diversity compared to the original one.

Further supporting this hypothesis, evaluation of controllability in Table 6 shows that MEGATRON-CNTRL-124M-ANT achieves a high controllability score of 77.5%. This means that by changing the keywords to their antonym, 77.5% of newly generated stories change their semantic content to follow the new antonym keywords. We also show that larger models are better able to leverage keyword control. Scaling the model size from 124M to 355M and 8B model further improves the controllability score to 84.5% and 91.5%, respectively. We again observe the quality (e.g. coherence) of our controlled generation improves as the model size scales to 8B.

Name	Controllability ↑
M-CNTRL-124M-ANT	77.5%
M-CNTRL-355M-ANT	84.5%
M-CNTRL-8B-ANT	**91.5%**

Table 6: Human evaluation for controllability by changing keywords to their antonyms. Over 77% of our generation changes according to the keywords.

4.3 Ablation Studies

In this section, we conduct the ablation study on the planning strategy and external knowledge. The study of model size can be found in the Appendix A.3.

Name	Repeat ↓	Distinct ↑
M-CNTRL-124M (D)	20.04	80.14
M-CNTRL-124M w/o knowledge (D)	23.59	79.39
M-CNTRL-124M (S)	23.87	79.45
M-CNTRL-124M w/o knowledge (S)	23.98	79.61

Table 7: Ablation studies of static (S) vs dynamic (D) planning strategy, with and without knowledge.

4.3.1 Planning Strategy

In this section, we investigate the effects of planning strategy in our framework. Yao et al. (2019) showed that static planning works better than dynamic planning for LSTM-based models. To introduce the static planning in our model, we predicted all the keywords and relevant knowledge sentences from the starting sentence and generated the entire stories. When we compare these generations with the stories generated by dynamic planning, we see in Table 7 (first and third rows) that dynamic planning outperforms the static planning strategy with higher distinction (+0.7%) and lower repetition (-3.8%) scores. This is due to direct guidance over each sentence provided by the retrieved knowledge from dynamic planning . In contrast, in static planning, the retrieved knowledge sentences are all predicted together at the beginning using only the starting sentence, which makes the supervision for each story sentence weaker and noisier.

4.3.2 External Knowledge

In this section, we investigate the importance of retrieved knowledge. Table 7 (first and second rows) shows that, when excluding the knowledge from our framework (i.e. MEGATRON-CNTRL-124M w/o knowledge), distinction score decreases by 0.8% and repetition increases by 3.6%, highlighting the importance of external knowledge in our approach. Unlike dynamic planning, we observe that in static planning, the external knowledge does not play an important role in the quality of the generations and using or not using the knowledge leads to similar qualities. This observation also confirms that knowledge needs to be planned dynamically.

5 Future Work

Short story sentences in ROC story dataset limits our exploration from several potential research directions. For example, how long the control signal

would propagate for longer generations? Investigating this issue using longer story datasets (e.g. WRITINGPROMPTS (Fan et al., 2018)) is a subject for future work. Other interesting direction may include incorporating the structure-level controllability by adding it as either an extra input for the conditional generator or a multitask learning supervision for each sequence.

We also observed that in some cases during the generation, our model simply mentions the given word in the sentence, and talks about things that are not strictly related to or restricted by the given word. For example, the generated story of MEGATRON-CNTRL-8B in Table 15 only mentions the keyword "realize" instead of centering around it. This is caused by the RAKE keywords extractor, which does not always extract the keywords that represent the sentence well. One way to mitigate this issue is to leverage longer context information to identify better keywords which is subject of the future work.

6 Related Work

Knowledge Incorporation of knowledge into language models has shown promising results for downstream tasks, such as factual correct generation (Logan et al., 2019) , commonsense knowledge graph construction (Bosselut et al., 2019), entity typing (Zhang et al., 2019) and etc. More recently, several works have shown that inclusion of learned mechanisms for explicit or implicit knowledge can lead to the state-of-the-art results in Question Answering (Guu et al., 2020; Karpukhin et al., 2020; Lee et al., 2019; Lewis et al., 2020) and dialogue modeling (Roller et al., 2020).

Storytelling There are several different storytelling tasks described throughout the literature. Storytelling can be classified into story completion (Chen et al., 2019), story ending generation (Guan et al., 2019), story generation from prompts (Fan et al., 2018) or titles (Yao et al., 2019), and story generation from a given sentence (Guan et al., 2020). Different approaches have been developed to model the structure of stories with storylines (Yao et al., 2019), skeletons (Xu et al., 2018), Conditional Variational AutoEncoders (Wang and Wan, 2019) and a coarse-to-fine framework (Fan et al., 2019). Other works focus on incorporating commonsense knowledge into story generation with attention-based models (Guan et al., 2019; Chen et al., 2019). Recently, pre-trained language models have been used to finetune on both story completion datasets and commonsense knowledge to further improve the quality of story completion (Guan et al., 2020). However, few works concern the controllability of language model generation, especially for the large pre-trained models that are common in today's literature.

Controllable Generation Controllable text generation has a wide range of applications, including controlling through persona (Zhang et al., 2018; Boyd et al., 2020), politeness (Niu and Bansal, 2018), etc. Wiseman et al. (2018) presented controlling generations by learning latent, discrete templates from data. Fu et al. (2019) discovered the importance of pivot words that determines the sentence attributes and presented a lexical analysis framework. To control large pre-trained models, Keskar et al. (2019) demonstrated the ability to control text generation through a wide range of aspects, such as domains and links. Plug-and-play language models Dathathri et al. (2019) also address whole document controllability by adding a linear classifier on top of GPT-2 to predict whether generated text observes a particular style or property. Prabhumoye et al. (2020) provides a good survey of five modules for control. Differing from these works, we control the generation through keywords backed by external knowledge.

7 Conclusion

In this paper, we proposed a novel framework that adds control to text generation with external knowledge. Our model first generates a set of keywords and a knowledge retriever then queries an external knowledge base for triples related to the keywords. Based on the relevance to the story context, a contextual knowledge ranker ranks the retrieved knowledge sentences and feeds the top ones to a conditional generator to generate the next story sentence. Experimental results on the ROC story dataset showed that our model outperforms state-of-the-art models by generating less repetitive, more diverse and logically consistent stories. Human evaluation of the controllability of our model shows that 91.5% of generated stories are successfully controlled by changing keywords to their antonym. In line with current trends, we also demonstrate that using larger pre-trained language models consistently improves both the quality of the generated stories and controllability.

References

Antoine Bosselut, Hannah Rashkin, Maarten Sap, Chaitanya Malaviya, Asli Celikyilmaz, and Yejin Choi. 2019. Comet: Commonsense transformers for automatic knowledge graph construction. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4762–4779.

Alex Boyd, Raul Puri, Mohammad Shoeybi, Mostofa Patwary, and Bryan Catanzaro. 2020. Large scale multi-actor generative dialog modeling. *arXiv preprint arXiv:2005.06114*.

Daniel Cer, Yinfei Yang, Sheng-yi Kong, Nan Hua, Nicole Limtiaco, Rhomni St John, Noah Constant, Mario Guajardo-Cespedes, Steve Yuan, Chris Tar, et al. 2018. Universal sentence encoder. *arXiv preprint arXiv:1803.11175*.

Jiaao Chen, Jianshu Chen, and Zhou Yu. 2019. Incorporating structured commonsense knowledge in story completion. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 6244–6251.

Sumanth Dathathri, Andrea Madotto, Janice Lan, Jane Hung, Eric Frank, Piero Molino, Jason Yosinski, and Rosanne Liu. 2019. Plug and play language models: a simple approach to controlled text generation. *arXiv preprint arXiv:1912.02164*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Angela Fan, Mike Lewis, and Yann Dauphin. 2018. Hierarchical neural story generation. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 889–898.

Angela Fan, Mike Lewis, and Yann Dauphin. 2019. Strategies for structuring story generation. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2650–2660.

Yao Fu, Hao Zhou, Jiaze Chen, and Lei Li. 2019. Rethinking text attribute transfer: A lexical analysis. In *Proceedings of the 12th International Conference on Natural Language Generation*, pages 24–33.

Aaron Gokaslan and Vanya Cohen. 2019. Openwebtext corpus.

Jian Guan, Fei Huang, Zhihao Zhao, Xiaoyan Zhu, and Minlie Huang. 2020. A knowledge-enhanced pretraining model for commonsense story generation. *arXiv preprint arXiv:2001.05139*.

Jian Guan, Yansen Wang, and Minlie Huang. 2019. Story ending generation with incremental encoding and commonsense knowledge. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 6473–6480.

Kelvin Guu, Kenton Lee, Zora Tung, Panupong Pasupat, and Ming-Wei Chang. 2020. Realm: Retrieval-augmented language model pre-training. *arXiv preprint arXiv:2002.08909*.

Vladimir Karpukhin, Barlas Oğuz, Sewon Min, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. 2020. Dense passage retrieval for open-domain question answering. *arXiv preprint arXiv:2004.04906*.

Nitish Shirish Keskar, Bryan McCann, Lav R. Varshney, Caiming Xiong, and Richard Socher. 2019. Ctrl: A conditional transformer language model for controllable generation.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Kenton Lee, Ming-Wei Chang, and Kristina Toutanova. 2019. Latent retrieval for weakly supervised open domain question answering. *arXiv preprint arXiv:1906.00300*.

Patrick Lewis, Ethan Perez, Aleksandara Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, et al. 2020. Retrieval-augmented generation for knowledge-intensive nlp tasks. *arXiv preprint arXiv:2005.11401*.

Robert Logan, Nelson F Liu, Matthew E Peters, Matt Gardner, and Sameer Singh. 2019. Barack's wife hillary: Using knowledge graphs for fact-aware language modeling. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5962–5971.

George A Miller. 1995. Wordnet: a lexical database for english. *Communications of the ACM*, 38(11):39–41.

Nasrin Mostafazadeh, Lucy Vanderwende, Wen-tau Yih, Pushmeet Kohli, and James Allen. 2016. Story cloze evaluator: Vector space representation evaluation by predicting what happens next. In *Proceedings of the 1st Workshop on Evaluating Vector-Space Representations for NLP*, pages 24–29.

Tong Niu and Mohit Bansal. 2018. Polite dialogue generation without parallel data. *Transactions of the Association for Computational Linguistics*, 6:373–389.

Shrimai Prabhumoye, Alan W Black, and Ruslan Salakhutdinov. 2020. Exploring controllable text generation techniques. *arXiv preprint arXiv:2005.01822*.

Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. 2018. Improving language understanding by generative pre-training.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners.

Stephen Roller, Emily Dinan, Naman Goyal, Da Ju, Mary Williamson, Yinhan Liu, Jing Xu, Myle Ott, Kurt Shuster, Eric M Smith, et al. 2020. Recipes for building an open-domain chatbot. *arXiv preprint arXiv:2004.13637*.

Stuart Rose, Dave Engel, Nick Cramer, and Wendy Cowley. 2010. Automatic keyword extraction from individual documents. *Text mining: applications and theory*, 1:1–20.

Zhihong Shao, Minlie Huang, Jiangtao Wen, Wenfei Xu, et al. 2019. Long and diverse text generation with planning-based hierarchical variational model. 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 3248–3259.

Mohammad Shoeybi, Mostofa Patwary, Raul Puri, Patrick LeGresley, Jared Casper, and Bryan Catanzaro. 2019. Megatron-lm: Training multi-billion parameter language models using gpu model parallelism. *arXiv preprint arXiv:1909.08053*.

Robert Speer and Catherine Havasi. 2012. Representing general relational knowledge in conceptnet 5. In *LREC*, pages 3679–3686.

Trieu H Trinh and Quoc V Le. 2018. A simple method for commonsense reasoning. *arXiv preprint arXiv:1806.02847*.

Tianming Wang and Xiaojun Wan. 2019. T-cvae: Transformer-based conditioned variational autoencoder for story completion. In *Proceedings of the 28th International Joint Conference on Artificial Intelligence*, pages 5233–5239. AAAI Press.

Sam Wiseman, Stuart M Shieber, and Alexander M Rush. 2018. Learning neural templates for text generation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3174–3187.

Thomas Wolf, Victor Sanh, Julien Chaumond, and Clement Delangue. 2019. Transfertransfo: A transfer learning approach for neural network based conversational agents. *arXiv preprint arXiv:1901.08149*.

Jingjing Xu, Xuancheng Ren, Yi Zhang, Qi Zeng, Xiaoyan Cai, and Xu Sun. 2018. A skeleton-based model for promoting coherence among sentences in narrative story generation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 4306–4315.

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.

Lili Yao, Nanyun Peng, Ralph M. Weischedel, Kevin Knight, Dongyan Zhao, and Rui Yan. 2018. Plan-and-write: Towards better automatic storytelling. *CoRR*, abs/1811.05701.

Rowan Zellers, Ari Holtzman, Hannah Rashkin, Yonatan Bisk, Ali Farhadi, Franziska Roesner, and Yejin Choi. 2019. Defending against neural fake news. In *Advances in Neural Information Processing Systems*, pages 9051–9062.

Saizheng Zhang, Emily Dinan, Jack Urbanek, Arthur Szlam, Douwe Kiela, and Jason Weston. 2018. Personalizing dialogue agents: I have a dog, do you have pets too? In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2204–2213.

Zhengyan Zhang, Xu Han, Zhiyuan Liu, Xin Jiang, Maosong Sun, and Qun Liu. 2019. Ernie: Enhanced language representation with informative entities. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 1441–1451.

Yukun Zhu, Ryan Kiros, Richard S. Zemel, Ruslan Salakhutdinov, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. 2015. Aligning books and movies: Towards story-like visual explanations by watching movies and reading books. *CoRR*, abs/1506.06724.

A Appendices

A.1 GPT-2 Hyperparameters:

We used the BPE subword tokenizer from Radford et al. (2019) to tokenize each sentence of the ROC story dataset. The maximum sequence length is set to 1024 learned positional embeddings. An Adam optimizer (Kingma and Ba, 2014) with a learning rate of 0.0001 is utilized. We added dropout to both the embedding layer and multi-head attention layers with 0.1 probability. Gradients were clipped to a global gradient norm of 5.0. We finetuned the GPT-2 models for 5 epochs and selected the best one with the lowest perplexity on the validation set.

A.2 BERT Hyperparameters:

We set the maximum sequence length to 512 learned positional embeddings. We used a WordPiece tokenizer with the `bert-large-uncased` vocabulary for tokenization. The model was also optimized with an Adam optimizer with a learning rate of 0.0001, but it used a weight decay of 0.01. Gradients are clipped to a global norm of 1.0. We also added dropout to embedding layer and multi-head attention layers with 0.1 probability. For the selection of margin, we tried 0.1, 0.5, 1.0, and 5.0. The choice of 5.0 gives the best result.

A.3 Model Size

In addition to analyzing the effect of scale on our conditional generative model, we also performed an ablation study on the model size of our GPT-2-based keyword predictor and BERT-based ranker models. The results in Table 8 show that increasing the keyword model size from 774M to 2B reduces the repetition while keeping the other scores similar. Increasing the size of our contextual ranker from 336M to 1.2B reduces the repetition while also decreasing the diversity. It is not clear which one is better. We conjecture that as the positive samples, R^i, we used to train our contextual ranker are weakly supervised, and the fact that we used templates to synthetically convert knowledge triples to knowledge sentences, scaling the model size might be overfitting to noise. We, therefore, use the smaller, more computationally efficient model with 336M parameters for ranker models in all our experiments.

Name (a-b-c)	PPL ↓	Repeat ↓	Distinct ↑
2B-2B-336M	**6.31**	21.2	**82.6**
2B-2B-1.2B	6.35	**19.7**	81.2
2B-774M-1.2B	6.33	20.4	81.4

Table 8: Ablation studies varying keyword prediction model (b) and ranking model (c) keeping the conditional generator fixed (a). Increasing keyword prediction model reduces repetition. Larger ranking models does not give consistently better scores as it may overfit to noise due to the weakly supervised labels.

A.4 Datasets Used for pre-trained Models

The pre-trained GPT-2 models were trained on a 174GB corpora including: Wikipedia (Devlin et al., 2018), OpenWebText (Gokaslan and Cohen, 2019), RealNews (Zellers et al., 2019), and CC-Stories (Trinh and Le, 2018). For BERT models we include BooksCorpus (Zhu et al., 2015) in the training dataset.

A.5 Input Format

For the format of R^j, we add a " SEP " string to separate different knowledge sentences r^j_k in R^j. We add a " EOK " string to denote the end of knowledge sentences.

For the story context $X^{i-1} = \{x^1, x^2, \cdots, x^{i-1}\}$ where $x^j = [R^j, s^j]$, we add a "OS" token to the end of each s^j to denote the end of sentence. At the end of the story, a " |endoftext| " token is appended. We then concatenated X^{i-1} with R^i as the input to conditional generator.

A.6 Preprocessing

We follow the implementation of Guan et al. (2020), to replace " ." with ". ". More details have already been provided in Section 3.1.

A.6.1 Computation Resources

We used up to 20 DGX-1V servers (a total of 160 Tesla V100 32GB GPUs) for our experiments. Our infrastructure is highly optimized for large scale deep learning applications. The servers are connected via NVSwitch with 300 GB/sec bandwidth between GPUs inside a server and 100 GB/sec of interconnect bandwidth between servers using 8 InfiniBand adapters per server. As our story dataset is small, the training of GPT-2 models usually takes less than three hours. For the BERT model, it took eight hours as we have $\sim$ 15 million samples to train.

A.7 Generation Examples

A.7.1 Single Sentence Context

We present five generation examples keeping the context fixed to one sentence.

Context: [MALE] and his friends are bored on a weekend day.
Guan et al. (2019) they decided to sneak into a movie . they got a bunch of popcorn and a movie . then , they made it to a movie . now , they have a really good time .
MEGATRON-CNTRL-124M keyword: go, decides it is going to be a big night. [MALE] and his friends decide to go out. they get drunk and have a great time. when they are done they feel happy.
MEGATRON-CNTRL-8B keyword: go, decides they decided to go to the video game store. [MALE] picks out a new video game. [MALE] and his friends have fun playing the new video game. [MALE] is no longer bored.
MEGATRON-CNTRL-8B-ANT keyword: come a new tv show begins to come on the local channel. [MALE] is excited to watch his favorite tv show. he watches his show and they all laugh together. [MALE] and his friends have a fun time watching their favorite show.

Table 9: Example 1

Context: [MALE] had a heavy duty backpack.
Guan et al. (2019) he took it to the airport . he checked his pockets . [MALE] did not have a bag . [MALE] was sad .
MEGATRON-CNTRL-124M keyword: took he took it to the store one morning. he put it on the shelf at the store. a coworker looked at it. then he told [MALE] he hated lugging around.
MEGATRON-CNTRL-8B keyword: took he took it to the army base camp. he packed it with items like food , ammo , and tools. he took it to the jungle every day. one day , he took it out and his backpack exploded.
MEGATRON-CNTRL-8B-ANT keyword: abstain he abstained from buying another backpack. [MALE] used the old backpack for years. his old backpack was a big burden. [MALE] bought a new backpack with greater weight capacity.

Table 10: Example 2

Context: [NEUTRAL] 's blind date insisted on going to the new sushi place.
Guan et al. (2019) she accepted without saying a word . later , he picked out a sushi roll . [NEUTRAL] was so happy with her new sushi rolls . she had to wait for him to come back .
MEGATRON-CNTRL-124M keyword: sure, date she decided to go to the restaurant. she ordered a blowfish. they gave her a small grin. she had a great time.
MEGATRON-CNTRL-8B keyword: thought, rude [NEUTRAL] was nervous because he was worried she would be rude. but at the table , she was sweet and friendly. and she was very nice , even though she ate little sushi. [NEUTRAL] wondered if maybe blind dates were actually very wonderful !
MEGATRON-CNTRL-8B-ANT keyword: uncertain [NEUTRAL] was uncertain. she was not a sushi person but agreed to try it. [NEUTRAL] and her blind date ordered salmon sushi. the salmon sushi was delicious.

Table 11: Example 3

Context: [MALE] hiked up the local hill.
Guan et al. (2019) he stopped and picked up his brush . he used that brush to brush his teeth . afterwards , he was ready to ride back home . he went on his way home .
MEGATRON-CNTRL-124M keyword: top he got a great view from the top. his legs felt great. he climbed up the mountain and felt much better. he went back to the hill and reached the top.
MEGATRON-CNTRL-8B keyword: top he was at the top of the hill looking down. the weather looked beautiful. the clouds suddenly passed. the sun began to shine over the land again.
MEGATRON-CNTRL-8B-ANT keyword: bottom he saw a huge boulder at the bottom. he could n't believe its size. he had to climb to the top of the hill. he was afraid he would fall if he stood on top.

Table 12: Example 4

Context: her favorite glasses were ruined.
Guan et al. (2019) she was really upset . she had no choice but to buy new ones . she went to the mall to shop and found a great pair . she bought them and was so happy .
MEGATRON-CNTRL-124M keyword: wore she wore them every day. then she had a horrible accident. her sister had broken them. she had to buy new ones.
MEGATRON-CNTRL-8B keyword: wore she wore them daily. she got them at the shoe store. she found them and they were her favorite. after a while they got broken.
MEGATRON-CNTRL-8B-ANT keyword: refresh she was desperate to refresh them. she found a recipe online. she poured bleach in them. she was glad to have functional glasses.

Table 13: Example 5

A.7.2 Varying Number of Sentences in the Context

We present three generation examples varying the number of sentences in the context.

Context: [MALE] and his friends are bored on a weekend day. it is going to be a big night.
MEGATRON-CNTRL-124M keyword: go, decides [MALE] and his friends decide to go out. they get drunk and have a great time. when they are done they feel happy.
MEGATRON-CNTRL-8B keyword: decides [MALE] decides to watch horror movies. his friends join him. they watch horror movies all night.
MEGATRON-CNTRL-8B-ANT keyword: stop they stop by a local arcade after the baseball game. [MALE] turns on his favorite arcade game. [MALE] 's favorite game makes him feel happy.

Table 14: The context contains two sentences.

Context: [MALE] liked trading commodities. one day [MALE] was trading several products. [MALE] noticed that he gained a lot of money from the new trades.
MEGATRON-CNTRL-124M keyword: lost, money [MALE] invested the money into stocks. [MALE] lost all of his trading profits.
MEGATRON-CNTRL-8B keyword: realize [MALE] realized that the new trades were his best trading idea. [MALE] quit his old job and went full time on his new trading idea.
MEGATRON-CNTRL-8B-ANT keyword: keep [MALE] decided to keep doing this. now [MALE] had a successful trading career.

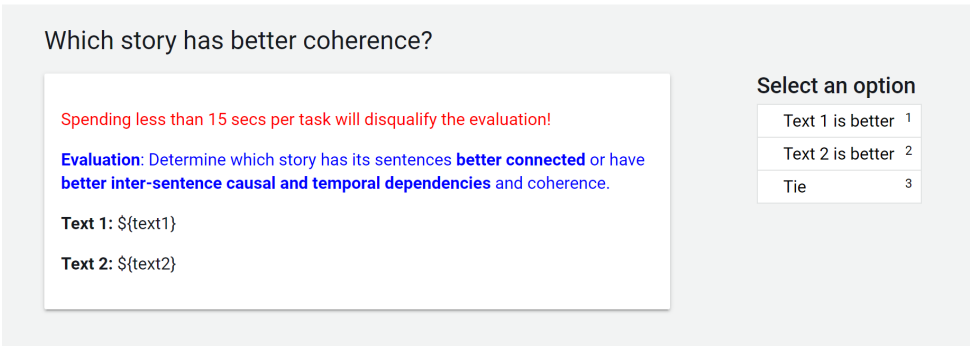
Table 15: The context contains three sentences.

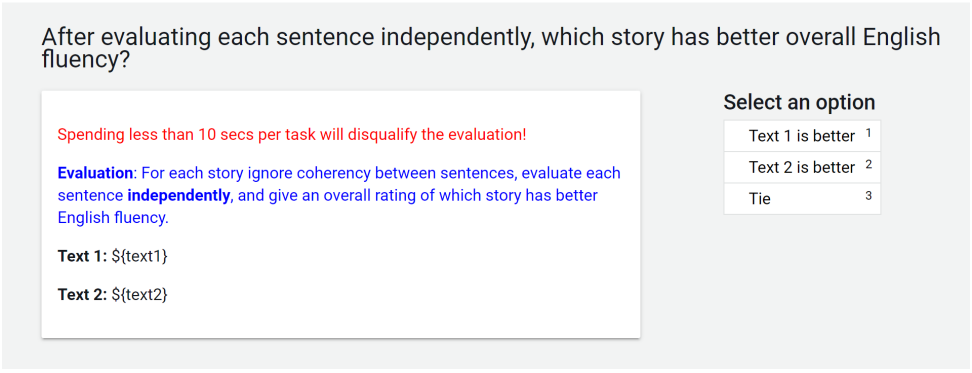
Context: we went to a dinner and show one night. the first guy we saw was really nice. after that we asked him out. he agreed.
MEGATRON-CNTRL-124M keyword: got we got into a bad argument.
MEGATRON-CNTRL-8B keyword: NONE (no keywords) we had a really good time.
MEGATRON-CNTRL-8B-ANT keyword: end we ended up dating exclusively.

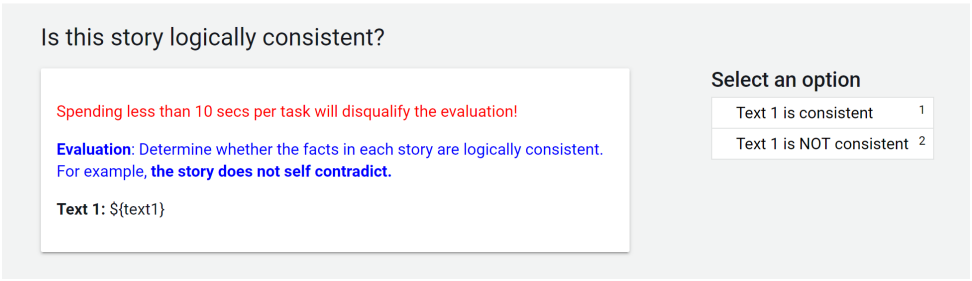
Table 16: The context contains four sentences.

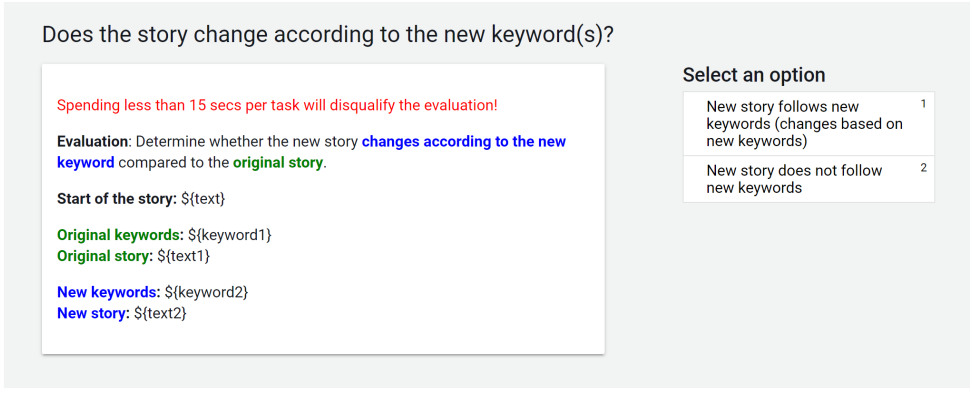
A.8 Human Evaluation Interface for Annotators

Below we provide the interfaces used for human evaluations (coherence, fluency, logical consistency, and controllability).


Which story has better coherence?
Spending less than 15 secs per task will disqualify the evaluation!
Evaluation: Determine which story has its sentences better connected or have better inter-sentence causal and temporal dependencies and coherence.
Text 1: ${text1}
Text 2: ${text2}
Select an option
Text 1 is better 1
Text 2 is better 2
Tie 3


After evaluating each sentence independently, which story has better overall English fluency?
Spending less than 10 secs per task will disqualify the evaluation!
Evaluation: For each story ignore coherency between sentences, evaluate each sentence independently, and give an overall rating of which story has better English fluency.
Text 1: ${text1}
Text 2: ${text2}
Select an option
Text 1 is better 1
Text 2 is better 2
Tie 3


Is this story logically consistent?
Spending less than 10 secs per task will disqualify the evaluation!
Evaluation: Determine whether the facts in each story are logically consistent. For example, the story does not self contradict.
Text 1: ${text1}
Select an option
Text 1 is consistent 1
Text 1 is NOT consistent 2


Does the story change according to the new keyword(s)?
Spending less than 15 secs per task will disqualify the evaluation!
Evaluation: Determine whether the new story changes according to the new keyword compared to the original story.
Start of the story: ${text}
Original keywords: ${keyword1}
Original story: ${text1}
New keywords: ${keyword2}
New story: ${text2}
Select an option
New story follows new keywords (changes based on new keywords) 1
New story does not follow new keywords 2