Plug-and-Blend: A Framework for Controllable Story Generation with Blended Control Codes

Zhiyu Lin
Georgia Institute of Technology
North Ave NW, Atlanta, GA 30332
zhiyulin@gatech.edu

Mark O. Riedl
Georgia Institute of Technology
North Ave NW, Atlanta, GA 30332
riedl@cc.gatech.edu

Abstract

We describe a Plug-and-Play controllable language generation framework, Plug-and-Blend, that allows a human user to input multiple control codes (topics). In the context of automated story generation, this allows a human user loose or fine grained control of the topics that will appear in the generated story, and can even allow for overlapping, blended topics. We show that our framework, working with different generation models, controls the generation towards given continuous-weighted control codes while keeping the generated sentences fluent, demonstrating strong blending capability.

1 Introduction

Recent advancement in very large pre-trained neural language models (e.g. (Radford et al., 2019; Brown et al., 2020)) have enabled a new generation of applications that make use of the text generation capability they provide, ranging from autocompletion of e-mails to solving complicated math equations. However these very large pre-trained neural language models are also difficult to **control** beyond providing a prompt for a generated continuation. This makes very large language models ill-suited for *co-creative* tasks wherein a human works with a language model in an iterative fashion to produce novel content, such as stories or poems. Co-creative tasks require an ability to not only prompt the language model but to guide the generation with, for example, style, context, or topic constraints.

Conditional generation is a family of text generation methods that attempt to provide controllability by either directly modifying the model to accept control signals or posing constraints in the generation process. Conditional text generation techniques add an extra input feature (Ficler and Goldberg, 2017) and fine-tuning with additional information embedded (Fang et al., 2021; Hosseini-Asl et al., 2020; Keskar et al., 2019; Khalifa et al., 2020; Hu et al., 2017; Wu et al., 2020; Ficler and Goldberg, 2017; Chan et al., 2020), or by sideloading additional discriminators along with a pre-trained model, without changing base model parameters holisticly (Dathathri et al., 2020; Madotto et al., 2020; Duan et al., 2020; Mai et al., 2020).

We seek "plug-and-play" approaches to controllable text generation wherein new language models can be slotted into existing generative systems; new language models are being developed and it becomes intractable to update and retrain controlled generation architectures. Plug-and-play techniques such as (Krause et al., 2020; Pascual et al., 2020) aim to only intervene with the outputs—a vector of logits—of a generative language model. This becomes especially important as the latest iteration of very large pre-trained language models such as GPT-3 (Brown et al., 2020) restrict access to the hidden states and layer weights of models. As language models improve, they can be easily incorporated into existing, controllable generation frameworks.

We present *Plug-and-Blend* [1], an efficient plug-and-play generative framework for controllable text generation that (a) works with the logit outputs of any language model; (b) facilitates fine control of generated sentences by allowing continuous bias towards specific control codes; and (c) allows multiple control codes representing style and topic constraints to be provided in overlapping contexts. These control codes can be blended together to generate content that meets multiple style or topic constraints. We describe that these key capabilities empower latent space walking in the hyperspace of generated sentences, and show a simple content planning technique that utilizes this feature to generate paragraphs regarding user intentions in a co-authoring. We present our work in the context

[1] Code available at https://github.com/xxbidiao/plug-and-blend

Proceedings of the 3rd Workshop on Narrative Understanding, pages 62–71
June 11, 2021. ©2021 Association for Computational Linguistics

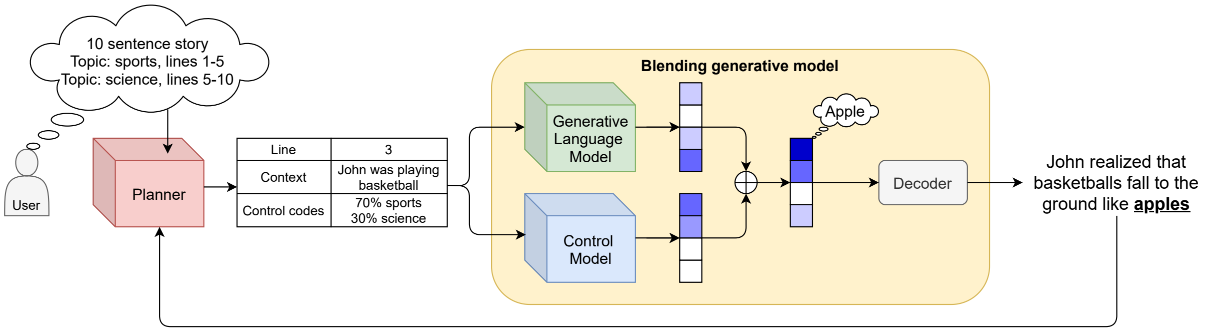

10 sentence story
Topic: sports, lines 1-5
Topic: science, lines 5-10
User
Planner
Line
3
Context
John was playing basketball
Control codes
70% sports
30% science
Blending generative model
Generative Language Model
Control Model
Apple
Decoder
John realized that basketballs fall to the ground like apples

Figure 1: Illustration of overall architecture of our framework

of automated story generation wherein a human author provides a prompt as well as a high-level control specification for topics.

2 Related Work

2.1 Plug-and-Play Conditional Generation

Researchers aim for "plug-and-play" (PnP) frameworks (Dathathri et al., 2020) which can be used along an existing generative LM (referred to as the “base LM”) with minimum or no interference between the PnP components and the base LM.

Comparing to non-plug-and-play methods ("white-box" approaches), these frameworks can be roughly classified into three categories. *Gray-box* approaches access and modify some non-input-output layer computations, usually the hidden representation, hence “plugging” an additional model in the middle of the base LM (Dathathri et al., 2020; Madotto et al., 2020; Duan et al., 2020; Mai et al., 2020). *Black-box* approaches including “Prompt Engineering” that aim to change the prompts fed into the base LM at inference time (Wallace et al., 2019; Li and Liang, 2021). *Guided generation* targets at building a controllable “guiding” model that shifts the output from base LM at inference time (Krause et al., 2020; Pascual et al., 2020).

The generation model we propose is an extension of GeDi (Krause et al., 2020). Adding to the complete decoupling of generation and controlling, we enhanced it with additional capabilities to support multi-topic generation with continuous weighting, supporting the downstreaming applications while keeping its capability to transfer to different base LMs.

2.2 Controllable Story Generation

Neural story generation systems train or fine-tune a language model on story data. Sampling from a language model trained on story data tends to result in text output that looks like stories as well. However, sampling from $P_\theta(x_t|x_{<t})$ (See Section 3) is uncontrolled in the sense that one does not have any influence over the output after the initial context prompt.

A number of story generation systems have attempted to condition the generation with some form of high-level plan. Storytelling systems such as (Akoury et al., 2020; Yao et al., 2019) embeds topic constraints directly into the model. These system extract a set of topics from a dataset that must be incorporated into the story. PlotMachines (Rashkin et al., 2020) allows a human user to specify topics that can be incorporated into a story in any order. Wang et al. (2020) generate a story by interpolating between a start event and an end event in a slot filling fashion, targeted the same goal. Our work differs in two ways. First, we allow blending of topics such that a single line in a story can meet more than one topic provided by a human user. Second, we have developed a black-box plug-and-play system that works with different LMs.

3 Preliminaries

Generative Language Models (LMs), specifically continuation models, take a context (“prompt”) and generate a continuation by predicting the next tokens. This is achieved by optimizing the model parameters θ that best estimates the probability density of a sequence of word tokens $x_{1:T} = \{x_1, \ldots, x_T\}$ represented as an auto-regressive factorization

$$P_\theta\left(x_{1:T}\right)=\prod_{t=1}^{T} P_\theta\left(x_t \mid x_{<t}\right). \qquad (1)$$

By iteratively predicting a distribution on the next token given the previous tokens, a continuation can be generated by repeatedly sampling $P_\theta\left(x_t \mid x_{<t}\right)$

and attach the selected token back to the "previous" tokens for the next step.

Sequences generated this way are not controlled; To control the generated sequence, an **attribute** represented as a class variable (Keskar et al., 2019) that could describe sentiment or topics can be introduced to equation (1) to form a Class-Conditional Language Model (CC-LM):

$$P_\theta\left(x_{1:T} \mid c\right)=\prod_{t=1}^{T} P_\theta\left(x_t \mid x_{<t}, c\right) \qquad (2)$$

where c represents the class variable, or "control code", that describes an **attribute** of the sequence $x_{1:T}$. However, since c and $x_{1:T}$ are entangled in equation (2), naively optimizing P_θ requires a new CC-LM to be trained.

To decouple the conditional generation component, c, from the unconditional part, $P_{LM}\left(x_{1:T}\right)$, (Krause et al., 2020) proposed the GeDi framework and an algorithm to enable a separate controlling model to guide the generation process of a base language model. Instead of tackling $P_\theta\left(x_{1:T} \mid c\right)$ directly, they train a contrastive discriminator model on the side to estimate

$$P_\theta\left(c \mid x_{1:t}\right)=\alpha P(c) \prod_{j=1}^{t} P_\theta\left(x_j \mid x_{<j}, c\right) \qquad (3)$$

where α is the normalization constant $\alpha = 1/(\sum_{c' \in \{c, \bar{c}\}} \prod_{j=1}^{t} P\left(c'\right) P_\theta\left(x_j \mid x_{<j}, c'\right))$, and c and c' are contrastive control codes (c and not-c). At the decoding stage of the generation process, one can guide the generation by using $P_\theta\left(c \mid x_{1:t}\right)$ as a posterior to the output probability distribution of the base LM:

$$\begin{aligned} &P\left(x_t \mid x_{<t}, c\right) \propto \\ &\quad P_{LM}\left(x_t \mid x_{<t}\right) P_\theta\left(c \mid x_t, x_{<t}\right)^{\omega} \end{aligned} \qquad (4)$$

where ω is a parameter for control strength, with larger values biasing generation more strongly towards c. CC-LMs trained this way do not require access to any internal data of the base LM, and works independently of it.

4 The Plug-and-Blend Framework

Our *Plug-and-Blend* framework consists of two components (See figure 1): (1) a *blending generative Model* that is responsible for plug-and-play controlled continuations using the control specifications; and (2) a *planner* that plans and assigns control specifications based on control sketches.

A *control sketch* is a high-level specification of what topics should be present in the story and what portions of the story each topic should approximately appear in. This provides a human co-creator the ability to guide the generator loosely, with a broad range per topic, or tightly, with a narrow range per topic. We envision a co-creative loop wherein the human user provides a control sketch and iteratively updates the control sketch based on generation results, refining the topics and refining the ranges for the topics. The user interface for eliciting control sketches from a human is outside the scope of this paper and experiments about the co-creative loop are left for future work. The next sections provide the algorithmic support for control sketches.

4.1 Blending Generative Model

The blending generative model generates the sentence continuation. It consists of two parts, a (1) plug-and-play language model and (2) a control model. Given a prompt $x_{<t}$, the plug-and-play language model produces a vector of logits $P_{LM}\left(x_t \mid x_{<t}\right)$. The control model biases the output of the language model toward particular tokens associated with the topics of the control codes $c \in C$ based on the desired strengths of each topic $\omega^*_{c \in C} \in \Omega$. Together the two models iteratively find the best token x_t that reflects both natural language composition and control bias presented by c and ω. A larger ω^*_c means more steering towards the topic represented by control code c.

Inspired by the application of generative adversarial networks to latent space walking, we treat $P_\theta\left(c \mid x_t, x_{<t}\right)$ (described in section 3) as a heuristic of **direction** that increases $P\left(x_t \mid x_{<t}, c\right)$ in a $|V|$-dimensional latent space, where V is the language model's vocabulary. For example, consider two different control codes c_1 and c_2 instantiating equation (4). To apply both control codes in the generation process, we use the heuristic

$$\begin{aligned} &P\left(x_t \mid x_{<t}, c_1, c_2\right) \propto P_{LM}\left(x_t \mid x_{<t}\right) \times \\ &\quad P_\theta\left(c_1 \mid x_t, x_{<t}\right)^{\omega_1} P_\theta\left(c_2 \mid x_t, x_{<t}\right)^{\omega_2} \end{aligned} \qquad (5)$$

to combine the effect of both posterior distributions into one universal posterior. ω_1 and ω_2 in this case represents control strength for each control code, c_1 and c_2 respectively, and can be different, enabling continuous blending between topics. This process can be repeated with a set of control codes $C = \{c_1, \ldots, c_n\}$ with weights $\Omega = \{\omega_1, \ldots, \omega_n\}$.

Formally, at the decoding stage of the generation process, a control model compute controlled probability using the following equation:

$$P\left(x_t \mid x_{<t}, C\right) = \\ P_{LM}\left(x_t \mid x_{<t}\right) \prod_{c^* \in C} P_\theta\left(c^* \mid x_t, x_{<t}\right)^{\omega_c^*} \quad (6)$$

where the control strengths of individual control codes are normalized with $\sum_c \omega_c^* = \omega$, where ω is total control strength.[2] This can be efficiently computed by batching input sequences appended by different control codes, with little overhead compared to the original GeDi (Krause et al., 2020) framework.

4.2 Planner

The human user provides a high-level control sketch of the story, consisting of the number of sentences, N, a set of topics, C, and a range of lines to which to apply the topic, $r := (s, e)$ where $s \leq e$. See figure 2 for example sketches. Sketches can have their range r overlap such that multiple topics can be applied to the same lines of the story.

Given the control sketch, the planner produces a control configuration C_n, Ω_n for each sentence position $n = \{0, \ldots, N-1\}$. The control configuration for each sentence is passed to the blending generative model along with previous generated sentences as prompt.

We interpret a control sketch as story arc on a specific topic, which typically contains a transition, an engagement and a phase-out, the planner should give highest control strength to the midpoint of the area, $m := (s+e)/2$, and lower strength towards the start and end of the span of the area; We capture this as a Gaussian distribution.

Formally, the following equation translates the sketch into a control configuration for each position $n \in N$:

$$\omega_{c,n}^{+} = f(\mathcal{N}(m, (\sigma/(e-s+\epsilon)^2))(n-m) \quad (7)$$

where $f(\cdot)$ indicates probability density function, ϵ is an infinitesimal, and σ is a tunable parameter representing overall transition smoothness, where higher σ grants smoother transitions in the cost of reduced topic engagement for midpoint. Since there can be multiple control sketches and they can be of the same control code, we apply each individual sketch in the order they are presented and normalize after each application so that $\Sigma_n \omega_{c,n} = 1$.

5 Experiments

For our experiments, we use the GPT2-large model fine-tuned on ROCStories (Mostafazadeh et al., 2016) as our base language model. Fine-tuning GPT2 on ROCStories results in a model that generates short stories about common everyday situations. We pair the language model with a pre-trained GeDi (which in turn is based on GPT-2-medium) trained on AG-news[3] as the guiding model. Across all setups, at generation time, we use greedy decoding with repetition penalty described in Keskar et al. (2019), and only use the first sentence generated as the output, discarding every token after it if any.

Since there is no ground truth for any generated sequence, metrics such as BLEU and other n-gram-based metrics are not applicable. This poses a unique challenge in evaluating our system, limiting us to unsupervised metrics. In this section, we report evaluation of our blending generative model from two aspects:

- Fluency: measuring how our generated sequence forms natural language; and
- Control fidelity: measuring how our generated sequence respects the requested control codes and strength.

5.1 Blending Fluency

To evaluate fluency of sequences generated by our blending generation model, we use perplexity of *base* language model. The intuition is that if generated sentences have low average perplexity when evaluated by the base LM then they are consistent with sentences we would find in the English language, as represented by the data used to train the base LM. This in turn results in fluent-appearing sentences.

To generate sequences from our model, we used 100 sentences from a held-out evaluation set of ROCStories not seen at fine-tuning time. ROCStories contains five-sentence stories; we always pick the first sentence. That sentence becomes our prompt and is paired with all possible combinations of two topic choices chosen from “Business”, “Science”, “Sports”, or “World”. These are the topics that the GeDi model are optimized for. Our control sketch gives equal blending weighting for all topics. We vary the control strength using the following

[2] This is not the only way to formalize this heuristic; We found this to be effective and efficient.

[3] http://groups.di.unipi.it/~gulli/AG_corpus_of_news_articles.html

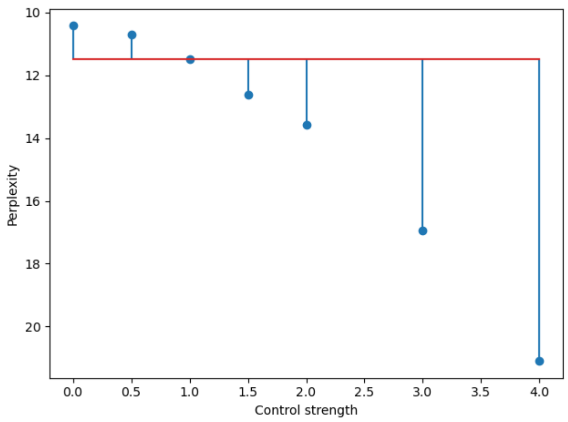


Figure 2: Perplexity (lower is better) of generated sequences with 2 topics. Baseline performance set at $1x$ of (Krause et al., 2020)-suggested control strength.

increments: $[0, 0.5, 1, 1.5, 2, 3, 4]x$, where 0 represents an uncontrolled base LM and $4x$ represents 400% of the control strength hyperparameter used by Krause et al. (2020).

Figure 2 shows the average perplexity of generated sequences, measured by the Base LM. We observe that average perplexity increases with stronger control, signaling a departure of generated sequences from what the base LM would generate, and a potential decrease in fluency. This is to be expected as the control is biasing the generated text more and more toward the use of words that are consistent with a particular topic and away from general word frequency. While perplexity increase is more or less linear in the range of 0 to 2x strength, once above 2x strength, it can be better described as exponential, hinting a stabler capability to generate fluent sentences in the region of 0 to 2x control strength.

5.2 Control Fidelity

Control fidelity is how well the generator responds to multiple control codes applied at once (see Krause et al. (2020) for experiments applying one control code at a time; we do not replicate them in this paper). For story generation, multiple control codes can be applied to the same sentence in a story at different weights. We perform experiments in a latent space walking setting, to measure content changes of generated sentences under the same prompt, same control codes but different relative control strength, in an unsupervised way.

Given a particular prompt line in a story and two control topics c_1 and c_2, we re-generate the same line multiple times under different control strengths for each topic. Specifically we set ω_{c_1} to 0%, 25%, 50%, 75% or 100% and $\omega_{c_2} = 1 - \omega_{c_1}$ to represent a range of different possible blends of topics in the same line. See table 1 for an example. Since we know the control parameters used to generate these sentences, in which c_1 receives more and more control strength relative to c_2, we expect to see sentences that are increasingly about topic c_1 and decreasingly about topic c_2. These sentences do not comprise a story sequence, but are different alternative sentences for the same line in a story under different topic control specifications.

To determine whether a given generated sentence was representative of a topic, we score each generated sentence with an off-the-shelf BART-based zero-shot classifier (Wolf et al., 2020)[4] with c_1 and c_2, in raw text form, as possible classes. We then compare the order of the sentences as determined by the classifier to the ground truth order of increasing control strength of c_1. We report the correlation of order between these two sequences using Kendall's τ-a metric. A perfectly strictly increasing classifier score will grant a τ-a score of 1 for a sequence. If the sentences have some reordering based on classification score, τ-a is reduced. A score of 0 indicates a random ordering and and a score of -1 indicates a sequence that is exactly in opposite order. Table 1 shows the classifier scores for the possible next sentences under different control strengths; the classifier scores are not monotonically decreasing, resulting in a τ-a score of 0.8.

Figure 3 shows a heat-map of the average τ-a score of sequences of sentences generated with different control code pairs and different total control strength (percentages). For each combination of parameters, 100 sequences of 5 sentences are generated and evaluated. Comparing to the baseline, which is the evaluation metric applied to order-randomized stories in ROCStories dataset, we observe universal statistical significance ($p < .01$) in improvement in τ-a metric. That is, without a control bias, rank ordering is random. As we increase the total control strength, the rank order of generated sentences more closely matches the ground truth order.

Some topic combinations (For example, Science-Sports) work better than others (For example, Science-World); the "World" category appears to include a lot of overlapping vocabulary usage with

[4] pipeline("zero-shot-classifier")

Prompt: The people gathered to protest the court's ruling last week.

$c_1 =$ **Sports**	$c_2 =$ **Business**	**Generated Sentence**	**Classifier score**	
ω_{c_1}	ω_{c_2}		c_1	c_2
100%	0%	Coach Leeman was in a wheelchair and had been taken to hospital for treatment.	86%	14%
75%	25%	Coach Reebok was one of them.	65%	35%
50%	50%	The players were joined by a few of them.	84%	16%
25%	75%	The company that owns the team was fined $1,000 for violating a rule prohibiting employees from using their own equipment.	37%	63%
0%	100%	Bankruptcy Judge William H. said that the bank had failed to pay its creditors and was in default on $1 billion of loans it owed them.	24%	76%

Comparing column 1 with column 4, Kendall's τ-a $= 0.8$ for this generated sequence.

Table 1: An example sequence of sentences generated for evaluation of control fidelity. The first two columns indicate the requested control strengths for two topics, sports and business. The generated sentence results from the prompt and the control weights (all numbers are $2x$ the default control strength). The last two columns indicate the probability that each line is either Sports or Business based on a BART-based topic classifier. We expect to see the classifier score for c_1 decrease as the classifier score for c_2 increases.

the other categories. Note that a perfect Kendall's τ-a of 1.0 is likely impossible because our zero-shot topic classifier will introduce some noise to the ranking. However, the results show us that the plug-and-blend technique (a) significantly increases the likelihood that topics will be incorporated into sentences, and (b) is sensitive to blended topics.

Figure 4 shows the same experiment as above, but with a non-fine-tuned version of GPT2-large. This shows that the plug-and-blend technique works on language models that haven't been fine-tuned on ROCStories. The prompts are still selected from ROCStories, however, for comparison, but are not as representative of the untuned model. In this condition, the text generated will not read as sentences in stories. We observe similar improvements over the baseline, demonstrating the ability of our method in keeping the strong adaptation capability.

5.3 Planner Experiments

In this section, we qualitatively demonstrate the capability of our pipeline by analyzing the generated paragraphs using simulated user inputs described as sets of control sketches.

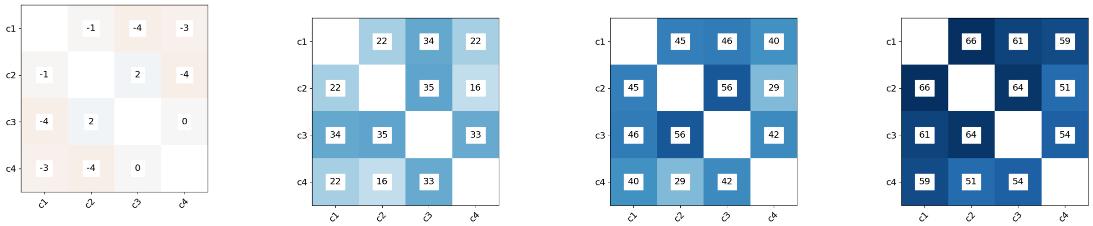
Table 2 (left column) shows three sets of control sketches with overlapping topic ranges. For example, sketch 1 requests a 10-line story that covers the topic of sports for the first 6 lines and covers the topic of science for the last 6 lines (topics overlap in the middle). For each control sketch we generate 10-line stories ($N = 10$) using the hyper-parameter $\sigma = 1$ (see Equation 7). We use a neutral prompt consisting of only the word "Recently" as the context to generate the first line or if the generator ever generates an empty line. The remainder of lines use up to 2 sentences generated for the previous context.

Table 2 (right column) shows the generated stories for each control sketch. We bold the sentence where it is most clear that the topic has changed. Figure 5 shows how the heuristic transforms each control sketch into bias weights. The figure shows ω_{c_1} for $c_1 = \text{Sports}$ showing how the planner decreases the probability density bias for the topic (the probability density for the second topic, ω_{c_2}, is the mirror image).

With slight differences in the input control sketches, we observe very different generated stories, with the transition between sports and science happening later. One can see from Figure 5 why this would be the case: the probability density for the first topic becomes increasingly stronger for the first lines of the story as the control sketch requests the second topic later.

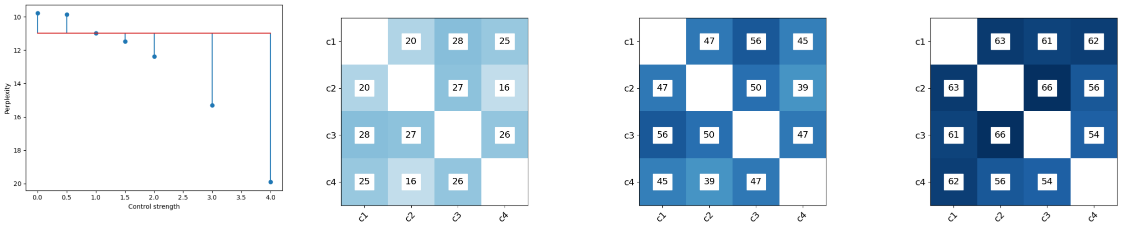
Because each sentence is biased by the previous sentences in addition to the control sketch, the sentence where the topic appears to switch often comes later than the point of earliest topic overlap. The requirement that each sentence continue the previous context creates a sense of momentum from the previous context and thus from the previous topic.

Incoherent transitions may still happen. In the story in Table 2 for sketch 3 shows one such incoherent transition due to the generation of an end-of-text token. Our implementation uses the initial prompt in this case, causing a portion of the story to not be contextualized by the earlier story sentences. Our ROCStories-tuned language model, based on



(a) Baseline on order-shuffled stories in ROCStories dataset. (b) Total control strength $1x$. (c) Total control strength $2x$. (d) Total control strength $4x$.

Figure 3: average τ-a (higher meaning better control fidelity) under different Total control strength for the tuned model with topics: (c1) Business, (c2) Science, (c3) Sports, (c4) World, comparing to uncontrolled baseline. Heat map strength is given as percentages ($-100\% \ldots 100\%$).



(a) Perplexity of generated sequences. (b) Total control strength 1x. (c) Total control strength 2x. (d) Total control strength 4x.

Figure 4: Experiment results for the untuned model. Refer to Figure 3a for baseline comparison.

5-sentence stories, tends to predict end-of-text earlier than models trained on longer stories.

6 Discussion

Our experiments suggest that there is a trade-off between control fidelity and fluency. As Figures 2 and 3 show, a higher total control strength results in overall better τ-a scores, meaning more sensitivity and ability to correctly differentiate between topic blends, but worse perplexity, risking less fluent language. In practice, an iterative deepening algorithm where multiple control strengths are used to generate multiple candidate sentences per line, can be used. Control strength modifiers of $1x$, $2x$, $3x$, $4x$, etc. can be tried and the best generated sentence, as measured by perplexity (or any other task-specific metric), is selected. This can, just like how multiple control codes are handled, be implemented very efficiently.

The current planner is heuristic. Empirically we find the heuristic to create good blends. We envision a planner that can be parameterized and learn from demonstrations. Reinforcement learning, in which the context and control sketches work as world states, can choose control configurations as actions. Feedback (reward) from the user would be necessary. This would incorporate the plug-and-blend technique into a human-in-the-loop creative process wherein the generator learns blending preferences from the human creator (Guzdial et al., 2018).

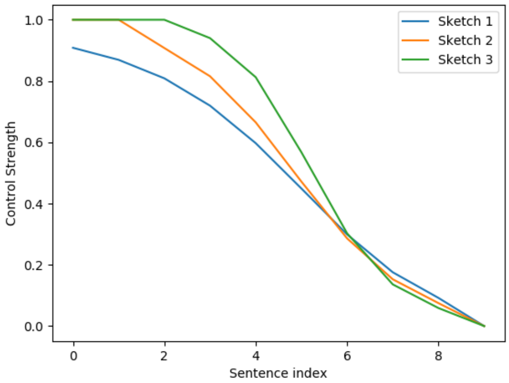


Figure 5: Control strength generated by the planner for the first control code used for each control sketch in Table 2. The control strength for the second control code is the mirror of each.

In this paper we have made no assumptions about how control sketches are acquired from the human user. We envision a processes whereby the human co-creator has an idea of what the final product—in this case a story—should be and can express that vision at the level of how topics manifest through the story. Supporting a co-creative

Control Sketches	Generated Paragraph
Sketch 1 $c_1 =$ Sports, $\{0 \rightarrow 5\}$ $c_2 =$ Science, $\{\mathbf{4} \rightarrow 10\}$	Jackie Robinson was playing in the NBA. He had just been drafted by the Cleveland Cavaliers. He decided to play for a team that was trying to win a championship. "I'm gonna go out there and beat them," he said "They're going to be great." "Yeah, they're good," he said "they'll be great." **He added that his wife had been working on the project for about a year.** **He was happy to see her progress.** She was able to get some work done and she started getting more comfortable with it. She had a lot of fun doing it! She also got her own computer, which is now in the living room.
Sketch 2 $c_1 =$ Sports, $0 \rightarrow 5$ $c_2 =$ Science, $\mathbf{5} \rightarrow 10$	Jackie Robinson was playing in the NBA. He had just played basketball for a few years. He was looking forward to his first game of the season. He decided to play with his friends and play against them in the court. He got a lot of feedback from everyone who played against him, including some that were very excited about it! I was really happy when I saw how he played. I also had to admit that my favorite player was the guy who beat me in the finals. **The computer game Super Mario 64 is a great game, but it's not perfect.** **I played it on my laptop and found that I couldn't play it properly because of some bugs.** The problem was that the graphics were bad, so I had to use an emulator instead of playing the game.
Sketch 3 $c_1 =$ Sports, $0 \rightarrow 5$ $c_2 =$ Science, $\mathbf{6} \rightarrow 10$	Jackie Robinson was playing in the NBA. He had just played basketball for a few years. He was looking forward to his first game of the season. He decided to play with his friends and play against them in the court. He had a lot of fun playing against them, but he didn't want to lose any time. So he played with his friends for about an hour before going home and playing again. He was very happy when they got home and started playing again! I think it's a good idea to have some fun with your kids, especially if you're not too busy. I'm sure that you'll enjoy this post as much as I did! **my daughter was diagnosed with a rare form of cancer.**

Table 2: Generated Examples with different Control-Sketches. Sentences in **bold** show a topic transition.

human-AI interaction, the human user can update the control sketch and re-generate parts (or all) of the story by changing the range of topics or choosing different topics. The control model will need to support different topics at different levels of granularity; currently the control model only supports four topics, which is sufficient for conducting experiments to characterize the plug-and-blend technique but not for full co-creativity.

7 Conclusions

In this paper, we present Plug-and-Blend, a plug-and-play framework that enhances a base LM, enables controllable generation with continuous-weighted control codes, along with capability of generating paragraphs based on control sketches, all without access to internal knowledge of this base LM. These capabilities will fuel a new generation of controllable generation applications with the key assets of decoupling between the controllable component and the generative component, and easiness of adapting to new advancements in the field of generative LMs.

8 Acknowledgment

This material is based upon work supported by the Office of Naval Research (ONR) under Grant #N00014-14-1-0003.

References

Nader Akoury, Shufan Wang, Josh Whiting, Stephen Hood, Nanyun Peng, and Mohit Iyyer. 2020. STORIUM: A Dataset and Evaluation Platform for Machine-in-the-Loop Story Generation. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6470–6484, Online. Association for Computational Linguistics.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language Models are Few-Shot Learners.

Alvin Chan, Yew-Soon Ong, Bill Pung, Aston Zhang, and Jie Fu. 2020. CoCon: A Self-Supervised Approach for Controlled Text Generation. *arXiv:2006.03535 [cs]*. ArXiv: 2006.03535.

Sumanth Dathathri, Andrea Madotto, Janice Lan, Jane Hung, Eric Frank, Piero Molino, Jason Yosinski, and Rosanne Liu. 2020. Plug and Play Language Models: A Simple Approach to Controlled Text Generation. *International Conference on Learning Representations*, (2020). ArXiv: 1912.02164.

Yu Duan, Canwen Xu, Jiaxin Pei, Jialong Han, and Chenliang Li. 2020. Pre-train and Plug-in: Flexible Conditional Text Generation with Variational Auto-Encoders. *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, (2020):253–262. ArXiv: 1911.03882.

Le Fang, Tao Zeng, Chaochun Liu, Liefeng Bo, Wen Dong, and Changyou Chen. 2021. Transformer-based Conditional Variational Autoencoder for Controllable Story Generation. *arXiv:2101.00828 [cs]*. ArXiv: 2101.00828.

Jessica Ficler and Yoav Goldberg. 2017. Controlling Linguistic Style Aspects in Neural Language Generation. *Proceedings of the Workshop on Stylistic Variation*, (2017):94–104. ArXiv: 1707.02633.

Matthew Guzdial, Nicholas Liao, and Mark Riedl. 2018. Co-Creative Level Design via Machine Learning. *Fifth Experimental AI in Games Workshop*. ArXiv: 1809.09420.

Ehsan Hosseini-Asl, Bryan McCann, Chien-Sheng Wu, Semih Yavuz, and Richard Socher. 2020. A Simple Language Model for Task-Oriented Dialogue. *Advances in Neural Information Processing Systems*, 33.

Zhiting Hu, Zichao Yang, Xiaodan Liang, Ruslan Salakhutdinov, and Eric P. Xing. 2017. Toward Controlled Generation of Text. In *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 1587–1596, International Convention Centre, Sydney, Australia. PMLR.

Nitish Shirish Keskar, Bryan McCann, Lav R. Varshney, Caiming Xiong, and Richard Socher. 2019. CTRL: A Conditional Transformer Language Model for Controllable Generation. *arXiv:1909.05858 [cs]*. ArXiv: 1909.05858.

Muhammad Khalifa, Hady Elsahar, and Marc Dymetman. 2020. A Distributional Approach to Controlled Text Generation. *arXiv:2012.11635 [cs]*. ArXiv: 2012.11635.

Ben Krause, Akhilesh Deepak Gotmare, Bryan McCann, Nitish Shirish Keskar, Shafiq Joty, Richard Socher, and Nazneen Fatema Rajani. 2020. GeDi: Generative Discriminator Guided Sequence Generation. *arXiv:2009.06367 [cs]*. ArXiv: 2009.06367.

Xiang Lisa Li and Percy Liang. 2021. Prefix-Tuning: Optimizing Continuous Prompts for Generation. *arXiv:2101.00190 [cs]*. ArXiv: 2101.00190.

Andrea Madotto, Etsuko Ishii, Zhaojiang Lin, Sumanth Dathathri, and Pascale Fung. 2020. Plug-and-Play Conversational Models. *arXiv:2010.04344 [cs]*. ArXiv: 2010.04344.

Florian Mai, Nikolaos Pappas, Ivan Montero, Noah A. Smith, and James Henderson. 2020. Plug and Play Autoencoders for Conditional Text Generation. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6076–6092, Online. Association for Computational Linguistics.

Nasrin Mostafazadeh, Nathanael Chambers, Xiaodong He, Devi Parikh, Dhruv Batra, Lucy Vanderwende, Pushmeet Kohli, and James Allen. 2016. A Corpus and Evaluation Framework for Deeper Understanding of Commonsense Stories. *Proceedings of the 2016 Conference of the North {A}merican Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 839–849. ArXiv: 1604.01696.

Damian Pascual, Beni Egressy, Florian Bolli, and Roger Wattenhofer. 2020. Directed Beam Search: Plug-and-Play Lexically Constrained Language Generation. *arXiv:2012.15416 [cs]*. ArXiv: 2012.15416.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language Models are Unsupervised Multitask Learners. page 24.

Hannah Rashkin, Asli Celikyilmaz, Yejin Choi, and Jianfeng Gao. 2020. PlotMachines: Outline-conditioned generation with dynamic plot state tracking. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 4274–4295, Online. Association for Computational Linguistics.

Eric Wallace, Shi Feng, Nikhil Kandpal, Matt Gardner, and Sameer Singh. 2019. Universal Adversarial Triggers for Attacking and Analyzing NLP. *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, (2019):2153–2162. ArXiv: 1908.07125.

Su Wang, Greg Durrett, and Katrin Erk. 2020. Narrative Interpolation for Generating and Understanding Stories. *arXiv:2008.07466 [cs]*. ArXiv: 2008.07466.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. 2020. HuggingFace's Transformers: State-of-the-art Natural Language Processing. *arXiv:1910.03771 [cs]*. ArXiv: 1910.03771.

Zeqiu Wu, Michel Galley, Chris Brockett, Yizhe Zhang, Xiang Gao, Chris Quirk, Rik Koncel-Kedziorski, Jianfeng Gao, Hannaneh Hajishirzi, Mari Ostendorf, and Bill Dolan. 2020. A Controllable Model of Grounded Response Generation. *arXiv:2005.00613 [cs]*. ArXiv: 2005.00613.

Lili Yao, Nanyun Peng, Ralph Weischedel, Kevin Knight, Dongyan Zhao, and Rui Yan. 2019. Plan-And-Write: Towards Better Automatic Storytelling. *Proceedings of the AAAI Conference on Artificial Intelligence*, 33(1):7378–7385. ArXiv: 1811.05701.