Sequently Controlled Text Generation

Alexander Spangher
University of Southern California
spangher@usc.edu

Xinyu Hua, Yao Ming
Bloomberg
{xhua22, yming7} @bloomberg.net

Nanyun Peng
University of California Los Angeles
violetpeng@cs.ucla.edu

Abstract

While GPT-2 generates sentences that are remarkably human-like, longer documents can ramble and do not follow human-like writing structure. We study the problem of imposing structure on long-range text. We propose a novel controlled text generation task, sequentially controlled text generation, and identify a dataset, NewsDiscourse as a starting point for this task. We develop a sequential controlled text generation pipeline with generation and editing. We test different degrees of structural awareness and show that, in general, more structural awareness results in higher control-accuracy, grammaticality, coherency and topicality, approaching human-level writing performance.

1 Introduction

Imagine that you are tasked with: Write a “Related Works” section. Would it help to know the past structure of the article (e.g. it is coming after the “Discussion” section)? How about the full structure of the article (e.g. after the “Introduction” but before the “Problem Statement”)?

The macro-structure of text (i.e. its discourse structure (Po’ttker, 2003)) impacts both human and machine comprehension (Emde et al., 2016; Sternadori and Wise, 2010; Lu et al., 2019; Zhou et al., 2020). Although naive language models have made impressive advancements and generate fluent text (Radford et al., 2019; Brown et al., 2020; Beltagy et al., 2020), the text is structurally dissimilar to human-written text (Figure 2, Section 7). Even the well-known Ovid’s Unicorn generation, which seems like a natural news article, exhibits unnatural structure (see Appendix F).

On the other hand, although numerous works have focused on content planning using keywords (Yao et al., 2019), plot-design (Rashkin et al., 2020) and entity tracking (Peng et al., 2021), macro-structural control has been relatively understudied.

So, in this work, we study (1) how to impose macro-structural control on narrative text generation and (2) how much structural awareness during generation contributes to well-structured and fluent text. We propose a novel task, sequentially controlled text generation. In this task, the user provides a prompt as well as a sequence of local control codes, each of which guides the generation of a single sentence. (In our experiments, we use headlines as prompts and Van Dijk (2013) discourse tags as control codes (Figure 1).)

We develop methods to address this task, expanding prior work focused on single control code generation (Keskar et al., 2019; Dathathri et al., 2019; Yang and Klein, 2021). As in prior work, the controlled generation problem is decomposed into a discriminator and a generator. However, in this work, the discriminator learns to incorporate an entire sequence of control codes. We hypothesize that information about structural intention can positively impact generative output (intuition for this is given in the hypothetical at beginning of this introduction). We show that our methods improve structural cohesion and certain aspects of
coherence over naive GPT-2 output.

Next, we hypothesize that more structural awareness improves generation. Again, we refer to the introduction hypothetical: humans craft text according to how it fits into a document’s full structure (Chenlo et al., 2014), so a generative model should similarly benefit from having such information. We test this hypothesis by varying the discriminator’s conditional independence assumptions. We experiment with three different degrees of control: local-only (where the discriminator is only aware of the current sentences’ control code), past-aware (where the discriminator is aware of the current sentences’ control code and all previous control codes), and full-sequence (where the discriminator is aware of the entire document’s sequence of control codes). We show that more structural awareness, especially of past structure, helps generate the highest-quality text. Finally, we show how to re-introduce a degree of local control by combining structurally-aware generation methods with a local sentence-level editing technique.

In summary, our novel contributions are:

- We propose a novel task, sequentially controlled text generation and identify a discourse schema (Van Dijk, 2013) and dataset (Choubey et al., 2020) to explore this task (Sections 2, 4).

- We combine two different approaches in controlled text generation: generation and editing, and show that the highest-quality text is generated when both of these approaches are used (Section 3).

- We use our methods to study the degree of structural control that yields the highest-quality text: local, past-aware and full-sequence control. We show that overall, full-sequence produces optimal text over an array of metrics (Section 7).

We see this work opening the door to a variety of follow-on directions: giving users control over the macro-structure of their generated output can allow users to quickly prototype different structures for their work. It can allow them to work in tandem with a generative algorithm to infill missing structural components in a piece of writing\(^1\). It might even allow them produce different versions of the same story for readers with different reading preferences. Finally, we also see macro-structural control providing a natural complement to, and being used in tandem with, other forms of controlled generation, like fact-aware generation (Logan IV et al., 2019) or creative generation (Goldfarb-Tarrant et al., 2020; Tian and Peng, 2022; Peng, 2022) to yield more engaging and useful generative content.

2 Problem Statement

We assume, as input, a headline sentence, \(X_0\), and a sequence of control codes \(\vec{c} = c_1, \ldots, c_S\) of length \(S\) (i.e., one for each sentence we wish to generate in the document. \textit{Adjacent codes can be of the same type.}) We wish to produce, as output, a document \(X\) of length \(S\) as a sequence of sentences \(X = X_1, \ldots, X_S\), each composed of a sequence of words \(X_k = x_1, \ldots, x_{n_k}\) of length \(n_k\).

We define the sequentially controlled text generation objective as:

\[
p(x|\vec{c}) = \prod_{k=1}^{S} \prod_{i=1}^{n_k} p(x_i|x_{<i}, X_{<k}, \vec{c}) \quad (1)
\]

\(^1\)Perhaps aiding in human-in-the-loop computational journalism (Cohen et al., 2011)
where $x_i$ is a word in sentence $k$, $x_{<i}$ are the preceding words, $X_{<k}$ are the preceding sentences (including the headline, $X_0$). $c_k$ is the control code for $k$. We assume that $\vec{c}$, the entire sequence of control-codes for a document, is given.

We use Bayes rule to factorize $t_1$ into:

$$t_2 \propto p(x_1|x_{<1}, X_{<k}) \frac{p(\vec{c}|x_1, x_{<1}, X_{<k})}{t_2; \text{naive word likelihood}} \frac{p(c_j|x_1, x_{<1}, X_{<k})}{t_3; \text{class likelihood}}$$

(2)

t_2$ is calculated using a standard pretrained language model (PTLM) and $t_3$ is calculated by a trained discriminator. This allows us to maximally re-use naively trained language models and, as we show, is far more resource efficient than fine-tuning a prompt-based model.

Three approximations for $t_3$ are:

**Local-Only**

$$t_3 \approx p(c_k|x_1, x_{<1}, X_{<k})$$

(3)

In the local-only model, we assume each control code $c_k$ is conditionally independent of other control codes given $x_i$. Thus, our generator model $t_1$ is made aware only of local structure: the control code $c_k$ pertaining to the current sentence, $k$. Because of this conditional independence assumption, local-only control is similar to prior work that used only single-control codes, where the goal was to generate a single sentence $p(x|e) = \prod_{i=1}^{n} p(x_i|e)$ (Keskar et al., 2019). However, we show that we can remove these independence assumptions and study more complicated structural control which, as we show later, produces more coherent output.

**Past-Aware**

$$t_3 \approx \prod_{j=1}^{k} p(c_j|x_i, x_{<1}, X_{<k}, c_{<j})$$

(4)

In the past-aware model, we assume autoregressive dependence between control codes, conditioned on $x$. Control codes for future sentences, $c_{>k}$, are conditionally independent. In Equation 1, this results in $x_i$ being dependent on $c_k$ and the sequence of control codes, $c_{<k}$.

**Full-Sequence**

$$t_3 = \prod_{j=1}^{S} p(c_j|x_i, x_{<1}, X_{<k}, c_{<j})$$

(5)

In the full-sequence model, we make no conditional independence assumptions.

We can restrict both the past-aware and the full-sequence approximations to a sliding window around sentence $s^2$. We can also add a prior on $p(\vec{c})$ to induce a discount factor$^3$. This focuses the generator on control code $c_k$ and down-weights surrounding control codes.

In the next section, we show how to model these objectives. We first describe the discriminator we use as our control-code model, the controlled generation techniques and the editing techniques we adopt.

### 3 Methodology

As described in Section 2, we can efficiently do generation by combining a naively-trained language model with a discriminator. Hence, the discriminator is the main architectural component that allows us to incorporate inter-dependencies between control code sequences. We start by describing how our discriminator models different degrees of structural awareness (Equations 3, 4 and 5) in Section 3.1.

We design a generation pipeline to balance structural and local awareness. The flow we use to accomplish this is depicted in Figure 3. The first step is Generation. Here, we sample each word, $x_i$, using techniques described in Section 3.2 which allow us to leverage our discriminator to impose structural control. When we have completed a sentence, we move to Editing. Here, we edit the sentence to further impose local control on each sentence, updating $x$ to optimize a variation of Equation 1: $p(x_i|x_{<1}, c_k)$, discussed in Section 3.3.

#### 3.1 Discriminator

The discriminator we construct takes as input a sequence of sentences ($X$) and a sequence of local control tags ($\vec{c}$). Our architecture combines a sentence-classification model, similar to that used in (Spangher et al., 2021a), with a separate label embedding architecture to incorporate knowledge of $c_{<j}$. Hence, we can make predictions for $c_j$ based not only on $x$, but prior tags, $c_{<j}$, allowing us to model structural dependencies (Equation 2). For a full description of architecture, see Appendix A.

---

$^2$i.e. $t_3$ ranges only from $j = k - w...k + w$ instead of the full sequence of sentences. In practice, we use $w = 3$.

$^3$The form of our prior is: $t_3 = \prod_{j=1}^{S} m(i, j)p(c_j|x_i, x_{<1}, X_{<k}, c_{<j})$, where $m(i, j) = b^{i-j}$. We experiment with $b = [.33, .66, 1]$. 

6850
We train it to model local-only, past-aware and full-sequence control variants expressed in Section 2: we train separate prediction heads to make predictions on \( c_{k-w}, ..., c_k, ..., c_{k+w} \), i.e. labels from \(-w, ..., +w\) steps away from current sentence \( k \).

For local-only control (Equation 3) we only use predicted probabilities from the main head, \( k \). In past-aware control (Equation 4), we multiply predicted probabilities from heads prior to the current sentence \( < k \), and for full-sequence control, we multiply predicted probabilities from all heads.\(^5\)

We now describe how we use these predictions.

### 3.2 Generation

We combine our discriminator’s predictions with a naive PTLM to solve Equation 2 in two different ways: **Hidden-State Control**, based on (Dathathri et al., 2019) and **Direct Probability**, based on (Yang and Klein, 2021).

**Hidden-State Control (HSC):** Wolf et al. (2019)’s GPT-2 implementation caches hidden states \( H \) to produce logits approximating \( p(x_i | x_{<i}) \). We perturb these hidden states \( H \), resulting in \( H' \) that produce logits approximating Equation 1 instead. We generate \( H \) from a naive PTLM and use this to make a prediction \( \hat{c} \) using our discriminator. We then calculate the loss \( L(\hat{c}, c) \) and backpropagate to \( H \) to derive \( \hat{H} \).

**Direct-Probability Control (DPC):** We calculate \( p(x_i | x_{<i}, X_{<s}) \) to identify the 200 most likely \( x_i \) under the naive language model, \( |x_{i,j}|^{200}_{j=0} \). Then we calculate \( p(c_i | x_{i,j}, x_{<i}, X_{<s}, C_{<s}) \) for each \( x_{i,j} \) using our discriminator. We directly multiply these probabilities to calculate Equation 1\(^6\).

Note that the HSC and DPC algorithms are extensions of previous work: the difference is that here they are used to model control code *sequences* rather than *single* tags. *The key components that allow this is our discriminator, which makes predictions based on label sequences, and our algorithm which, as shown in Figure 3, increments codes each time an \(<eos> \) token is generated.*

### 3.3 Editing

After we have finished generating a sentence, we edit it to introduce more discourse markers of the local control code.

We identify words in our input sequence that have the most impact on control-code prediction by using the gradient on our input sentence of the discriminator’s loss onto tokens and masking full words, following Ross et al. (2021). We use only the current sentence prediction made by our discriminator (i.e. Equation 3), so that we impose local control on the sequence even in settings where the generator imposes structural control.

\(^5\)Note: we still factor label-sequences autoregressively, as in Equations 4 and 5 and learn each prediction head separately. However, keeping separate heads allows the model more flexibility in predicting how attributes of a sentence might predict future or past tags. Preliminary experiments show that this approach outperforms learning a single head for all labels.

\(^6\)Note that DPC has the advantage of being simpler to implement and batch-parallelizable. However, the restriction to the top \( k = 200 \) words selected according to \( p(x_i | x_{<i}, X_{<s}) \) means that we might be limiting discriminator perturbation of word-selection.
We cull the high-gradient words based on heuristics7 to encourage the editor to introduce explicit discourse markers. We fine-tune a label-aware infilling model (Raffel et al., 2019) to generate candidate edits8 given the masked input. We mask and infill until we have generated a sentence that has an increased likelihood \( p(c_k|x_k) > p(c_k|x_k) \), and generate edit candidates (\( n = 10 \)). We select edits on the basis of class likelihood and perplexity9.

For more comparison and distinction from previous work for both Generation and Editing, see Appendix D.1, E.

### 4 Datasets and Schema

The form of sequential control we study is discourse: i.e. the functional role sentences play in a document’s larger argumentative purpose. We use a news discourse schema proposed by Van Dijk (2013), Choubey et al. (2020) to apply this schema and annotate a dataset, NewsDiscourse, consisting of 802 articles from 3 outlets10, tagged on the sentence level. Their schema consists of 9 classes: {Main Event, Consequence, Current Context, Previous Event, Historical Event, Anecdotal Event, Evaluation, Expectation}.11 Although each sentence is tagged with a code, codes often repeat. For example, an entire paragraph can be tagged with Main Event sentences. We show a partial sample in Figure 1. We adopt this schema to describe each news article’s structure.

We also use a dataset of unlabeled news articles12 to fine-tune a GPT-2 model for news. We sample 30,000 documents from this dataset in a manner so that the distribution of sentence-lengths matches the distribution of sentence lengths in the Choubey et al. (2020) dataset.

### 5 Implementation Details

We fine-tune a GPT2-base model on a large news corpus with a max word-piece length=204813. We use this to generate naive PTLM language-modeling as well as sentence-embeddings in our Discrimination model. Further implementation details are discussed in Appendix A.

We discuss the discriminator results here briefly. As shown in Figure 4, the primary head, \( p \), has a Micro F1-score of .65, which approaches state-of-the-art on this dataset14. However, performance degrades rapidly for heads farther from \( p \). For more

---

7Words that are not proper nouns, named entities (except the DATE class) or adjectives, as we find these categories are more likely to be topic words spuriously correlated with control-codes.

8A T5 model trained using a specific input template incorporating the label. E.g. label: Background. text: The senator <MASK> to the courtroom to <MASK>.

9Perplexity of the entire generated document so far is used as a selection criteria, \( PPL(x_t \oplus X_{<k}) \), to encourage edits preserving the logical flow of the document.

10nytimes.com, reuters.com and xinhuanet.com

11For a detailed class description, see Appendix F.1

12kaggle.com/snapcrack/all-the-news. Dataset originally collected from archive.org. We filter to articles from nytimes.com and reuters.com.

13Rather than 1024 in (Radford et al., 2019). We observe that >99% of human-generated news articles were shorter than 2048 word pieces.

14.71 Micro-F1 in Spangher et al. (2021a), which used auxiliary datasets.
results on discriminator performance, including experimental variations, see Appendix A.1.

6 Experiments

We sample 10 documents from the test set of our discourse dataset (n = 200) to test different pipeline settings. The input to our models is a headline (as a prompt) and the full sequence of gold-truth discourse labels of that document.

Baselines

We compare our experimental pipelines (Section 3) with the following baselines: (1) Naive GPT-2 generation given only the headline as input (i.e. no control codes), (2) a fine-tuned Prompting approach and (3) the original Human-written articles.

For (2), we directly train a class-conditional language model to generate text by including labels in the prompt, as in (Keskar et al., 2019). Local-only prompting is achieved by only including the local control code (and prior generated sentences) in the prompt, and updating the prompt to generate a new sentence. For past-aware prompting, we include all control codes prior to our current sentence in the prompt, and update on every new sentence. Finally, for full-sequence prompting, we including the full sequence of control codes in the prompt. (See Appendix C for more details and examples of prompt design.)

For each of these baselines, we test with and without editing (with the human-written text being edited by our algorithm in Human and with the generated text in all other trials being edited).

Evaluation

For all pipelines, we select the best hyperparameter configurations based on perplexity and model-assigned class likelihood. Then, we manually annotate each generated document for 4 metrics: Accuracy (0-1)\(^{15}\), Grammar (1-5)\(^{15}\), Logical Flow (1-5)\(^{15}\), and Topicality (1-5)\(^{15}\). We recruit two expert annotators with journalism experience to perform annotations blindly without awareness to which generation pipeline was used, and find moderate agreement \(\kappa \in [.36, .55]\) across all categories. For more details, see Appendix G. We

\(^{15}\)Accuracy: how close a generated sentence matches the discourse function of the gold-truth label for that sentence.

\(^{16}\)Grammar: how grammatical and locally coherent a sentence is.

\(^{17}\)Logical Flow: how well a sentence functions in the flow of the story.

\(^{18}\)How well each sentence corresponds to the original headline of the article.

7 Results

Best Overall Trial

We show automatic and human metrics for the subset of pipelines with top-performing hyperparameters in Table 2. In general, the highest-performing generation pipelines are all variations of DPC with either past-aware, or full-sequence structural control.

We observe that DPC with past-aware control and editing has the highest class-label accuracy, nearly approaching the human trials. The top performing pipelines for grammar and topicality are DPC with full-Sequence control and without editing. GPT-2 performed best only for Logical Flow, which was surprising but could perhaps be because the unconstrained nature of GPT-2’s generation allowed it to hallucinate a flow that seemed consistent even if it was poorly structured.

Effect of Different Pipeline Components

We show the distributional shifts in performance across all trials, in Figures 5, 6\(^{19}\). Structural control has a largely positive effect on generated text. In Figure 5, we find that Full-Sequence models are, on average, able to generate the most label-accurate sentences with the best grammar, logical flow and topicality. Finally, editing improves accuracy, grammar and logical flow (Figure 6.).

The original human-generated text is our gold-standard, and it is highly class-accurate, grammatical, coherent and topical. Interestingly, as seen in Table 2, editing can also be applied to human-written text to boost label accuracy, but at the expense of coherence.

8 Discussion

We set out to answer two questions in this research: (1) whether we could impose structural control over generated documents and (2) what kinds of structural control (local-only, past-aware, or full-sequence) had the greatest effect on discourse, flow, topicality and grammaticality. Our novel pipelines, which extend various discriminator-based approaches for generation and editing, approach human-level performance. However, a gap between our model’s output and human-generated

\(^{19}\)And 10, in Appendix E.
Table 2: Metrics on different trial runs. Each cell shows Unedited/Edited variants. (Hyperparams = $\gamma = .75$, $b = .33$). ** Optimal sentence length is determined relative human generation, i.e. $\min |x - 37.9|$. 

Figure 5: Comparison of different structural control methods across different pipelines and hyperparameters.

Figure 6: The effect of editing, across different pipelines and hyperparameters.

Insight #1: Some structural information improves all metrics of quality. Our structural exploration suggests that, for the best-performing pipelines, past structural information (along with editing) boosts class accuracy the most, but knowledge of the full-sequence does not. In the analogy given in the Introduction, this equates to: to write a “Related Works” section, it helps to know that it comes after the “Introduction” vs. the “Discussion”, but not information of what sections come after. This is perhaps because enough signal is already given by the past sequence and the full sequence just adds more noise. However, full-sequence information does yield the best grammar and topicality. This might indicate a regularizing role played by the full-sequence. In general, we suspect that past-aware modeling and editing both push the model more towards the class label at the expense of topicality, flow and grammar, while full-sequence does the opposite. In practice, some combination of these pipeline components might be desired.

Insight #2: Weak discriminators can still impose accurate control. At .61 macro F1, our discriminator is a relatively weak classifier. Previous work in classifier-based controlled text generation used large training datasets and classifiers that routinely scored above .8 F1 (Dathathri et al., 2019; Yang and Klein, 2021). The weakness of our discriminator is one reason why HSC may have performed poorly. However, in other trials we see strong accuracy. Thus, even with a weak classifier, we can control generation. This might be because even a weak discriminator can still give relative differ-
ences between generation that does or does match the control code.

**Insight #3:** Evaluating text candidates using multiple model’s perplexity might result in better selections. Just as surprisingly, editing also has an overall average positive effect on generation accuracy and generation quality (Figure 6). We had hypothesized that, because the editor makes locally-aware infilling decisions, it would improve class-accuracy but hurt other metrics of document quality, like topicality and flow. Indeed, for the top-performing trials, like DPC and Human, Editing only improves class accuracy. However, grammar and flow improves in other trials. This could be because, as mentioned in Section 3.3, we selected candidates based on how well they make sense in the document. This also suggests that using multiple PTLMs to select for better quality combines different virtues of each model.

**Error Analysis:** We observed that sentence tokenizing remained a huge challenge. Many of the grammar errors that our annotators observed were from sentences that ended early, i.e. after decimal points. Indeed, the correlation between sentence-length and grammar is relatively high ($r = .34$). One reason for this could be that error-prone sentence tokenizing models provided faulty training data during pretraining of LMs. This will continue to hinder document-level structural work, which often relies on a model accurately ending a sentence. Another observation, in Table 2, is that perplexity doesn’t necessarily correlate with human judgments of quality, especially for more complex writing like Financial news reporting.

9 Related Work

**Discourse-Aware Narrative Text Generation.** Generating narrative text, such as news articles and scientific reports, has been a long standing problem in NLP. Early work relies on template (Xu et al., 2018; Wiseman et al., 2018), rules (Ahn et al., 2016; Leppänen and Toivonen, 2021), or specialized architectures (Fan et al., 2018; Bosselut et al., 2018) that are hard to generalize.

Recently, pre-trained Transformers have shown impressive capabilities to produce fluent text, and there are few works that seek to adapt them to document-level generation with appropriate discourse structures. One work, DiscoDVT (Ji and Huang, 2021), uses a discrete variational auto-encoder (VAE) with a latent space guided by explicit Penn Discourse Treebank (PDTB) relations (Prasad et al., 2008). We are excited by this work, which shows strong improvements in coherence. While our work is able to learn from more abstract structural tags rather than low-level PDTB relations, our approach is fully supervised. We are excited by the semi-supervised nature of Ji and Huang (2021)’s approach, which may allow it to learn discourse structures that are less well annotated. Also, possible extensions of this approach using hierarchical discrete VAEs (Razavi et al., 2019) or diffusion models (Li et al., 2022) might provide users’ control over higher-order macro-structures in text, such as those explored in (Spangher et al., 2021b).

**Controlled Generation** The black-box nature of neural generation models poses challenges for many real-world applications (Wiseman et al., 2017; Holtzman et al., 2019). Researchers have designed various techniques to control the syntactic structure (Goyal and Durrett, 2020), sentiment (Hu et al., 2017; Fu et al., 2018; Luo et al., 2019), and language style (Niu and Bansal, 2018; Cao and Wang, 2021). Most notably, the CTRL model (Keskar et al., 2019) conditions the output by incorporating textual control codes during the pretraining stage. However, such training is resource-intensive and requires large datasets. Alternatively, PPLM (Dathathri et al., 2019), FUDGE (Yang and Klein, 2021), GeDI (Krause et al., 2021), and NADO (Meng et al., 2022) achieve inference-time control through either directly manipulating the generator’s hidden states, or adjusting the probabilistic distribution over the output vocabulary. Our work differs from prior work in that we tackle structured control instead of a single attribute. Our task, though, can be relatively easily addressed by extensions to these frameworks, and we look forward to future work that might improve further on the results we showed.

**Sequentially Controlled Generation** Sequential control for text generation has been explored from many angles, from symbolic planning approaches (Meehan, 1976; Lebowitz, 1987), to keyword-based approaches (Yao et al., 2019) and concept, event and entity driven planning approaches (Rashkin et al., 2020; Peng et al., 2021; Alabdulkarim et al., 2021; Han et al., 2022). We are the first, to our knowledge, to utilize a purely latent control
structure based off of discourse structures. There is increasing interest in exploring how discourse can be used to guide generation (Ghazvininejad et al., 2021; Cohan et al., 2018), from early works developing discourse schemas for generation (Mann, 1984; Stede and Umbach, 1998) to evaluating creative generation pipelines (Hua and Wang, 2020). However, neither direction allows discourse structures to be explicitly controlled in generation.

**Editing.** Most existing neural models generate text in one-shot, from left to right. Recently, an emerging line of research (Guu et al., 2018; Malmi et al., 2019; Kasner and Dušek, 2020) has explored editing as part of the generation pipeline to further improve the output quality, or satisfy certain desired constraints. Our work builds off of the MiCE framework (Ross et al., 2021), which was originally designed for generating contrastive explanations. Our observation that editing increased both the discourse adherence and the coherence of the generated text adds to a growing body of evidence that editing can play an important and modularized role in larger creative generation pipelines. We see editors such as simplification editors (Dong et al., 2019), factual correction editors (Cao et al., 2020) and stylistic editors (Kabbara and Cheung, 2021) each aimed at different desired attributes of text can possibly play a role in multi-attribute control (Li et al., 2022).

Finally, we see overlaps as well to an earlier paradigm of generative modeling: Bayesian models for text like Latent Dirichlet Allocation (LDA) (Blei et al., 2003) and, more interestingly, sequential variants (Du et al., 2012). There is recent work marrying PPLM-style controlled text generation with topic modeling (Carbone and Sarti, 2020). Such directions might lead to more hierarchical, structural control.

### 10 Conclusion

We have formalized a novel direction in controlled text generation: sequentially controlled text generation. We extended different techniques in controlled text generation to fit this direction, and have shown how a news discourse dataset can be used to produce news articles exhibiting human-like structure. We have explored what degrees of structural awareness yield the most human-like output: more structural control yields higher-quality output. And, we showed how to combine structural control with local editing. We have probed different parts of our pipeline to show the effects of each part.

### 11 Acknowledgements

Alexander Spangher acknowledges support from Bloomberg’s Data Science Ph.D. Fellowship. This work was conducted while Alexander was an intern at Bloomberg. We also acknowledge Jonathan May for helpful early conversations about this work. We acknowledge Vivienne Litzke for annotation effort and further conversations.

### 12 Ethics Statement

#### 12.1 Limitations

A central limitation to our work is that the datasets we used to train our models are all in English. As mentioned previously, we used Choubey et al. (2020)’s NewsDiscourse dataset, which consists of the sources: nytimes.com, reuters.com and xinhuanet.com. Although xinhuanet.com is a Chinese source, they used English-language articles. Additionally, we used an unlabeled news dataset from Kaggle for fine-tuning GPT2-base and for calculating some automatic metrics like % Unseen Words. We filtered this dataset down to two English-language, Western domains: nytimes.com and reuters.com in order to match the domains as closely as possible to the NewsDiscourse dataset.

Thus, we must view our work in discourse generation with the important caveat that non-Western news outlets may not follow the same discourse structures in writing their news articles. We are not aware of existing Van Dijk-style (Van Dijk, 2013) datasets towards which we could provide an exact comparison. But, we hope in future work to look at other kinds of discourse structures that might exist in other languages.

#### 12.2 Risks

There is a risk that the work will be used for misinformation or disinformation. This risk is acute in the news domain, where fake news outlets peddle false stories that attempt to look true (Boyd et al.; Spangher et al., 2020). Along this vein, there is the aforementioned work using discourse-structure to identify misinformation (Abbas, 2020; Zhou et al., 2020), and the risk in developing better discourse-aware generation tools is that these misinformation detectors might lose their effectiveness.

There is also a non-malicious misinformation risk, as large language models have been known to

---

20kaggle.com/snapcrack/all-the-news
generate hallucinated information (Choubey et al., 2021). The more such threads of research are pursued without an accompanying focus on factuality and truth, the more risk we run of polluting the information ecosystem. However, like others (Dathathri et al., 2019), we see a value in continuing this direction of research, even if this current work is not the final output we wish to see being used by non-researchers in the world. It is one step along the way.

There is also a risk that news articles in either of our datasets contain potentially libelious or defamatory information that had been removed from the publishers’ website after the dataset was collected. However, we do not release either of the datasets we use, so we do not see our actions as privacy-violating.

12.3 Licensing

Of the two datasets we used, NewsDiscourse (Choubey et al., 2020) is published as a dataset resource in ACL 2020. They collected reuters.com and xinhua.net via crawling, and the nytimes.com from existing academically licensed datasets (Bhatia et al., 2015; Sandhaus, 2008).

We were unable to ascertain the license for the Kaggle dataset. It has been widely used in the academic literature, including in papers published in ACL venues (Pathak and Srihari, 2019) and others (Alhuqail, 2021). We corresponded with the authors and opened a discussion question [URL withheld to preserve anonymity] seeking more information about the license. The authors are public about their desire to have their dataset used and we have had independent lawyers at a major media company ascertain that this dataset was low risk for copyright infringement.

12.4 Computational Resources

The experiments in our paper required computational resources. We used 8 30GB NVIDIA GPUs, AWS storage and CPU capabilities. We designed all our models to run on 1 GPU, so they did not need to utilize model or data-parallelism. However, we still need to recognize that not all researchers have access to this type of equipment. We used Huggingface GPT2-base models for our predictive tasks, and will release the code of all the custom architectures that we constructed. Our models do not exceed 300 million parameters.

12.5 Annotators

We recruited annotators from professional networks. Both consented to annotate as part of the experiment in exchange for acknowledgement. One is a graduate student studying in Europe, and the other is a former journalist. One annotator is female, and the other is male. One is half-Asian and half-white identifying, the other is white. Both identify as cis-gender. This work passed IRB.

References


Ryan L Boyd, Alexander Spangher, Adam Fourney, Bemira Nushi, Gireeja Ranade, James Pennebaker, and Eric Horvitz. Characterizing the internet research...
agency’s social media operations during the 2016 US presidential election using linguistic analyses.


Rujun Han, Hong Chen, Yufei Tian, and Nanyun Peng. 2022. Go back in time: Generating flashbacks in stories with event temporal prompts. In 2022 Annual Conference of the North American Chapter of the Association for Computational Linguistics (NAACL).


Felix Stahlberg, James Cross, and Veselin Stoyanov. Simple fusion: Return of the language model.


A Further Implementation Details

A.1 Discriminator Implementation

We tested 122 different discriminator variations. A summary of the major architectural iterations is shown in Table 3. We describe each variation as follows; the top-performing variation, with a subset of input sentences and labels, is shown in Figure 7.

Contextualized word vectors (\vec{w}) from a PTLM (we experimented with either GPT-2 or a RoBERTa as in Spangher et al. (2021a)) are obtained for each sentence, and are combined using self-attention. Switching to GPT-2 yielded a 16-point F1-macro score drop. Hidden-State Control, based on (Dathathri et al., 2019), relies on perturbations to the state variable \( H \) from the naive language model to generate word-probabilities \( p(x_i|X_{<k},x_{<i},\vec{c}) = p(x_i|H,\vec{c}) = p(\vec{c}|H,x_i)p(x_i|H) \). So, we need to use the same PTLM for the language model as we do for the discriminator. We do not have the same restriction on Direct Probability Control (Yang and Klein, 2021), as the probabilities are directly multiplied and thus do not need to share any architectural components. For the sake of an apples-to-apples comparison on the mechanism of control, though, we use a GPT-2 model for the PTLM layer in our discriminator.

Next, we tested either embedding each sentence separately in batch, or embedding the entire document (+Flattened Sentences). Embedding the entire document yielded a 3 point F1-macro increase.

These sentence vectors are then contextualized: we tested an LSTM layer (+LSTM) to contextualize these vectors and an autoregressive transformer layer (+Transformer). Using transformer yielded a 6 point F1-macro increase. We next fine-tuned the GPT-2 LM using its LM head on an unlabeled, 30K article news corpus. This yielded a 3 point F1-macro increase.

To incorporate label information as input to the model (as in the Past and Full variants) we embed each label using a learned embeddings layer, and then we combine these embeddings using self-attention. Experimenting with a different window size yielded a 5 point F1-macro increase. We find that a window of 3 yields the best-performing discriminator.

Table 3: F1 Macro on main prediction head, \( p_k \), for different discriminator variations. RoBERTa baseline is from Spangher et al. (2021a). GPT-2 variations described in body.

Finally, a feed forward classifier combines the sentence vector with the label vector. We find that sharing the PTLM improves accuracy, but not other layers.

A.2 Details on Hyperparameters

A.2.1 Discount Factor, \( b \)

To impose further structural control, we impose a prior on \( t_3 \) that acts as a discount factor. In words, we downweight the discriminator probabilities for control codes that are farther away from the current sentence being generated. The form of our prior is: \( t_3 = \prod_{j=1}^{n} m(i,j)p(c_j|x_i,x_{<i},X_{<k},c_{<j}), \) where \( m(i,j) = b^{||i-j||} \). We experiment with \( b = [0.33, 0.66, 1] \). So, the lower the discount factor, \( b \), the more the current, local control code matters. When \( b = 0 \), the local-only variant of our discriminator, Equation 3, is expressed by default.

Figure 7: Sentence classification model for \( k = 3 \) of a 3 sentence document. Word embeddings \( (\vec{w}_{ki}) \) for each sentence \( (X_k) \) are combined with self attention \( s_k \). A transformer contextualizes \( s_k \) with \( s_{<k} \). Labels \( \vec{c} \) are embedded \( (\vec{e}) \) and self-attention generates label vectors \( (h_k) \). \( a_k, h_k \) are combined for predictions \( (\vec{p}) \).
We see in Figure 8 that discount factor $b$ has a non-linear effect on the output. In accordance with our prior results, $b = 0$ is the lowest-performing variant across all four human-quality metrics. $b = .33$ seems to be the most effective discount factor overall, and yields the best output for accuracy and logical flow, while $b = 1$ yield the best-performing output for grammar and topicality. We conclude that a finer-grained balance of local control and structural control might be important overall, but in some cases more structural control might help as noted previously.

### A.2.2 Hidden-State Control (HS)

In Dathathri et al. (2019), authors find anywhere between 3 and 10 backpropagation steps is acceptable. In this work, we use 10 steps with a small step size. We also test different regularizations, also explored in (Dathathri et al., 2019), on the output logits generated from $\hat{H}$. We experiment with different hyperparameters for one of the regularizations: $l = \gamma l + (1 - \gamma) l^0$ where $l^0$ is the naive, unperturbed logits. We experiment with different values of $\gamma$ from 0 (fully unperturbed) to 1 (fully perturbed).

### A.2.3 Direct-Probability Control (DPC)

Authors in (Yang and Klein, 2021) offer an innovation by training their classifier $p(c|x)$ to consider subsequences $p(c|x_1, ... x_i)$ for all $i$, ostensibly improving the accuracy of their joint probability calculation while midsequence. This is in contrast to Dathathri et al. (2019)’s training regime, which only considers full sequences $p(c|x_1, ... x_n)$. However, Yang and Klein (2021) do not provide ablations to show whether it is this training regime, or their direct calculation of $p(x)p(c|x)$, which is responsible for the improvements they observe. In this work, we perform this ablation and find that it has negligible difference, according to automatic evaluation metrics. We also introduce a mean fusion (Stahlberg et al.) into the $p(x)p(c|x)$ joint likelihood: $\gamma p(c|x) + (1 - \gamma)p(x)$ and test different values of $\gamma$.

### B Automatic Metrics List

Here, we discuss the automated metrics reported in Table 2. They are largely based off metrics proposed in See et al. (2019).

#### B.1 Metrics Reported in Paper

**Label Probability** : We measure the label probability assigned to the gold-truth class label given in our input sequence: $p(c|x_{<s}, x_{<i}, X_{<s})$. We use head $p$, or the current head, in the discriminator shown in Figure 7.

**Perplexity** : Perplexity is calculated using the fine-tuned GPT-2 model, which we fine-tuned on 30,000 news articles.

**Diverse N-grams** : We measure the likelihood that an n-gram in one sentence will be unique compared with the entire document. In other words:

$$
\text{Diverse N-Grams}(s, d) = \frac{\# \text{ unique n-grams in sentence } s}{\# \text{n-grams in document } d}
$$

We calculate the set of n-grams per document as the total number of 1,2,3-grams in that document. We calculate one measurement per sentence in the document, and average these scores together.

**Sentence Length** : We measure the total number of words in the sentence, based on word-level tokenization using https://spacy.io/.

**Unseen Words** : We use an external corpus of 30,000 news articles to determine a typical, large news vocabulary. Any words that are outside of this vocabulary are considered “Unseen Words”. For our purposes, we are most interested in exploring malformed words, which are sometimes generated by the language model. However, unseen words might also be proper nouns.

### C Generation-Baseline #1: Prompting. Further Details

As a baseline, we train a language model to directly calculate $p(x_i|x_{<i}, X_{<s}, \hat{c})$, following (Keskar...
et al., 2019). We design the following prompt structure to simulate baseline, past-aware and full-sequence control variants.

**Baseline:**
- **Headline:** <Headline> Labels: <Current Label> Sentences: <Sentence 1> <Sentence 2> ... <Sentence s>
- **Sentence:**<Label 1> <Label 2> ... <Label s>

**Past-Aware:**
- **Headline:** <Headline> Labels: <Label 1>, <Label 2> ... <Label k> Sentences: <Sentence 1> <Sentence 2> ... <Sentence s>

**Full-Sequence:**
- **Headline:** <Headline> Labels: <Label 1>, <Label 2> ... <Label s> Current Position: <i> Sentences: <Sentence 1> <Sentence 2> ... <Sentence s>

The prompts are specific to current sentence being generated. We first start by generating sentence 1, whereby the prompt for Baseline and Past-Aware is both:
- **Headline:** <Headline> Labels: <Label 1> Sentences: <Sentence 1>

Then, we let the model generate the first sentence and stop when we generate the <EOS> character. We then regenerate the prompt to include the previously generated sentence and update the tags, so Baseline becomes:
- **Headline:** <Headline> Labels: <Label 2> Sentences: <Sentence 1>

and Past-Aware becomes:
- **Headline:** <Headline> Labels: <Label 1> <Label 2> Sentences: <Sentence 1>

We continue in this fashion, resetting the prompt each time, until we have finished generating sentences for all the tags in our input data.

The Full-Sequence process is very similar, except we do not need to update the label-space, since by default the model is exposed to the full sequence of tags before generation.

### D Editing

In this section, we describe the various components of the editing model. First, we note the differences in our approach and Ross et al. (2021)”s method. Then, we discuss the infilling model and the discriminator.

#### D.1 Key Differences

Ross et al. (2021) designed their editor to flip classifier predictions. So, they edited input $x \rightarrow \hat{x}$ until $c(p(\hat{x}|x)) \neq c(p(x)|x)$. Then, $\Delta(x, \hat{x})$ was given as the explanation for the flip. We are not concerned with flipping predictions so much as maximizing the probability of the ground truth label. So, we design our objective to be $x \rightarrow \hat{x}$ until $p(c|\hat{x}) > p(c|x)$.

To understand why the loss-gradient on the input can provide feature importance, consider the first-order Taylor approximation of the loss, $l(x) \approx l(a) + l'(a)(x-a)$. Here, the gradient of the loss at $a$, $l'(a)$, can be seen as a set of linear weights similar to logistic regression coefficients, which are commonly used for feature importance.

We also wished to restrict editing to **explicit discourse markers**, spuriously correlated words, so we heuristically excluded all Proper Nouns, Named Entities (except DATE) or adjectives from the edit candidate set. Table 6 shows explicit discourse markers in the news discourse context. Here, we show the top words associated with each discourse class$^{24}$. Some words effect the tense of the sentence$^{25}$, others inject epistemological uncertainty$^{26}$, still others time-peg events to certain days$^{27}$.

#### D.2 Infilling Model

We train a label-aware infilling model in a similar method as Ross et al. (2021). Our prompt is:

- **label:** <label> text: Lorem Ipsum <mask> Lorem <mask> Ipsum.

Where the masks replace high-salience words, which we discovered as described above. We format samples using sentences in our training dataset, and train a T5 model as described by the authors.

#### D.3 Possible Improvements

We note that this infilling method directly models $p(\hat{x}|M(x), c)$, i.e., the likelihood of infilled words given a label and a masked sentence. Another possible approach to this problem would be to use a naive infiller and Bayes rule as done in the generation phase of this paper to generate logits $p(\hat{x}|M(x))p(c|x, M(x))$. This could possibly improve the editor for the same reasons Dathathri

---

$^{24}$Most positive coefficients of a Logistic Regression Classifier that takes as input a sentence and predicts its discourse class

$^{25}$Top verbs in Expectation are almost all present-tense, while top verbs in Previous Event are almost all past-tense

$^{26}$Top verbs in Evaluation are all “say” verbs, while verbs in Current Context are based on observable events

$^{27}$Top Main Event nouns are nearly all weekday names
Table 4: A selection of sentences and the edit operations performed on them. The editor focuses on (a) temporal relations, (b) conditional statements (c) explicit discourse markers (e.g.
“expect”) and correct grammar.

![Diagram](image_url)

Figure 9: Transition Probability Matrix (log likelihood) for tag sequences.

et al. (2019) and Yang and Klein (2021) observed an improvement over CTRL (Keskar et al., 2019).

Another aspect of the editor that we noticed was that it could sometimes degrade the coherency and topicality of the document. This is especially evident in the Human trials. We partially addressed this by selecting candidate edits based off the perplexity of the whole document. We could have mitigated this further by giving our infiller the entire document as context.

E Further Methods Comparison

The standard controlled text generation setup is typically expressed as follows:

\[ p(x|c) = \prod_{i=1}^{n} p(x_i|x<i, c) \quad (7) \]

where \( x \) is the output sequence and \( c \) is a single control code (for example: sentiment (Dathathri et al., 2019)). Here, \( x \) is a single sentence (or paragraph) of \( n \) words, factorized autoregressively into words \( x_i \) and previous words \( x_{<i} \).

Previous approaches to controlled text generation (Dathathri et al., 2019; Yang and Klein, 2021) factorize the right term of Equation 7 as follows:

\[ p(x_i|x<i, c) \propto p(x_i|x<i)p(c|x_i, x_{<i}) \quad (8) \]

As in Equation 7, this factorization decomposes our sequentially controlled text generation model into an uncontrolled language model and a control code model. The key difference between Equation 8 and 2 is in the second term, i.e. how we choose to model the control codes (the difference in the first term is simply a rather trivial extension of a naive

\[^{28}\text{i.e. we could have trained a model based on } p(\hat{z}|M(x), X_{<i}, c), \text{ instead of } p(\hat{z}|M(x), c)\]

language from a sentence-to-paragraph generation to a document-generation context).

We show a direct comparison of all of our generation approaches in Figure 5. Here, we show that Direct Probability Control has the best effect over Naive GPT-2 for class-accuracy and, surprisingly, perhaps, Grammar and Topicality as well.

E Ovid’s Unicorn Is Not Structural

We annotate of the famous Ovid’s Unicorn news article generated and presented by the original GPT-2 authors (Radford et al., 2019).

We analyse this article as we have analyzed our generation models Section 7. One of our annotators gave each sentence the Van Dijk discourse label that best fits (Van Dijk, 2013), and the other assessed whether it actually fit. This is not an apples-to-apples comparison with the Label Acc. column in Table 2, because we are assessing the accuracy of the label that we chose after reading the text.

We next measured the likelihood that an article with the discourse structure of Ovid’s Unicorn would exist naturally. We build a simple bigram model for tags, \( p(c_{t+1}|c) \), to calculate the total...
Table 5: Log-Likelihood of Tag-Sequence, according to simple bi-gram model \( p(c_{t+1}|c_t) \), trained by counting tag sequences in the training dataset. 5th/50th/95th percentiles shown for test set.

<table>
<thead>
<tr>
<th>Discourse Label</th>
<th>Top words</th>
</tr>
</thead>
<tbody>
<tr>
<td>Main Event</td>
<td>monday cooperation shot</td>
</tr>
<tr>
<td>Consequence</td>
<td>closed showed issued</td>
</tr>
<tr>
<td>Previous Event</td>
<td>comment declined agency</td>
</tr>
<tr>
<td>Current Context</td>
<td>shot prime groups</td>
</tr>
<tr>
<td>Historical Event</td>
<td>2015 2016 2017</td>
</tr>
<tr>
<td>Anecdotal Event</td>
<td>want told old</td>
</tr>
<tr>
<td>Evaluation</td>
<td>say think told</td>
</tr>
<tr>
<td>Expectation</td>
<td>expected likely continue</td>
</tr>
</tbody>
</table>

Table 6: Top predictive words for each discourse type (top positive \( \beta \) coefficients for a Logistic Regression trained to predict \( y = \) news discourse tag per sentence using and \( X = \) a bag of words representation of each sentence).

**Previous Event**: A specific event that occurred shortly before the main event. It either directly caused the main event, or provides context and understanding for the main event.

**Current Context**: The general context or world-state immediately preceding the main event, to help the readers better understand and contextualize the main event. Similar to **Previous Event**, but not necessarily tied to a specific event.

**Historical Event**: An event occurring more than 2 weeks prior to the main event. Might still impact or cause the main event, but is more distal.

**Expectation**: An analytical insight into future consequences or projections made by the journalist.

**Evaluation**: A summary, opinion or comment made by the journalist on any of the other discourse components.

**Anecdotal Event**: Sentences describing events that are anecdotal, such events may happen before or after main events. Anecdotal events are specific events with specific participants. They may be uncertain and can’t be verified. A primary purpose of this discourse role is to provide more emotional resonance to the main event.

In Table 6 we attempt to provide more insight into different News Discourse elements by modeling using Logistic Regression.

**G Annotation**

We recruit two manual annotators, one with \( > 1 \) year and the other with \( > 4 \) years of journalism experience. Both annotators offered to perform these tasks voluntarily in exchange for acknowledgement.
For their reference, we showed the annotators the label definitions (shown in Section F.1) and a decision-tree (shown in Figure 11). The decision-tree breaks down key components of discourse reasoning.

Additionally, we gave them training annotation questions for practice. For the training task, they were asked to view human-written sentences from 10 articles and go through the step-by-step question process based on the decision tree. These labels were checked with the gold labels from the training dataset, and they trained until they were answering questions with >80% accuracy.

The interface we used to collect annotations is shown in Figure 12. Annotators were blind to the method that generated the text but were shown the desired true labels and simply had agree Y/N if the label fit.

For Grammar, we asked them to count the number of grammar mistakes per sentence (1: >6, 3: 2-4, 5: 0). For Logical Flow, we used a qualitative metric (1: “Poor”, 3: “OK”, 5: “Great”). For Topicality, we also used a qualitative metric (1: “Not at all”, 3: “OK”, 5: “Great”)

An earlier interface that asked annotators to assign their own tags was too difficult.