Changing the Mind of Transformers for Topically-Controllable Language Generation

Haw-Shiuan Chang Jiaming Yuan Mohit Iyyer Andrew McCallum
CICS, University of Massachusetts Amherst
hschang@cs.umass.edu, jiamingyuan@umass.edu,
{mccallum,miyyer}@cs.umass.edu

Abstract

Large Transformer-based language models can aid human authors by suggesting plausible continuations of text written so far. However, current interactive writing assistants do not allow authors to guide text generation in desired topical directions. To address this limitation, we design a framework that displays multiple candidate upcoming topics, of which a user can select a subset to guide the generation. Our framework consists of two components: (1) a method that produces a set of candidate topics by predicting the centers of word clusters in the possible continuations, and (2) a text generation model whose output adheres to the chosen topics. The training of both components is self-supervised, using only unlabeled text. Our experiments demonstrate that our topic options are better than those of standard clustering approaches, and our framework often generates fluent sentences related to the chosen topics, as judged by automated metrics and crowdsourced workers.

1 Introduction

Recently, Transformer-based language models (LMs) have achieved impressive performance in language generation tasks (Radford et al., 2019; Dai et al., 2019) such as open-domain story generation (See et al., 2019a). When writing with the LM, users often desire an intuitive and effective way to control what a LM is going to generate (Keskar et al., 2019). To address this need, interactive writing assistants provide options to reveal possible developments of the story and generate continuations guided by the user-selected options.

Interactive writing assistants have wide applications in creative writing (Roemmele and Gordon, 2015; Clark et al., 2018; Akoury et al., 2020), education (Luo et al., 2015), and gaming (Walton, 2020). Nevertheless, the existing systems' options usually do not provide fine-grained control and/or require substantial human labor. In some prior work (Keskar et al., 2019; Tu et al., 2019), users choose among a static set of predefined attributes (e.g., sentiment) that only provide coarse-grained control. Other work (Roemmele and Gordon, 2015; Clark et al., 2018) presents users with multiple generated continuations, which requires substantial reading effort and might not contain topics that users want to see. Finally, options could be nodes in a plot graph that are handcrafted (Luo et al., 2015) or derived from a collaboration between humans and machine (Li et al., 2013), but such choices are usually limited due to the high cost of preparing the options.

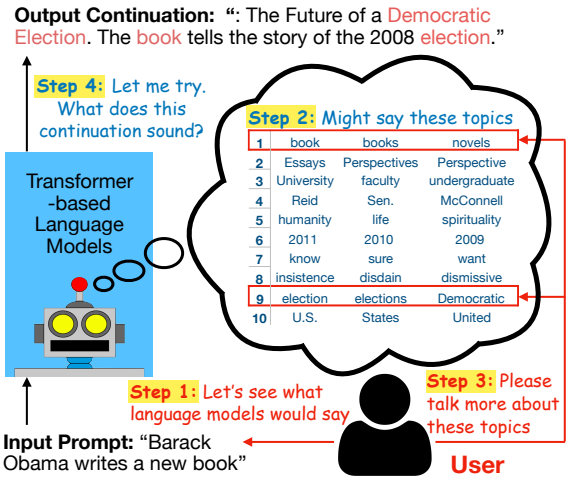

Figure 1: Given an input prompt, the Transformer-based LM provides $K = 10$ topics that might be mentioned next and each topic is represented by $M = 3$ words. The user could guide the generation process by choosing a subset of topics.

To address these limitations, we propose an interactive writing framework that provides a set of topics and guides the text generation by the user-chosen topics. The topic options are generated dynamically based on the input prompt to pro-

Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics, pages 2601–2611
April 19 - 23, 2021.

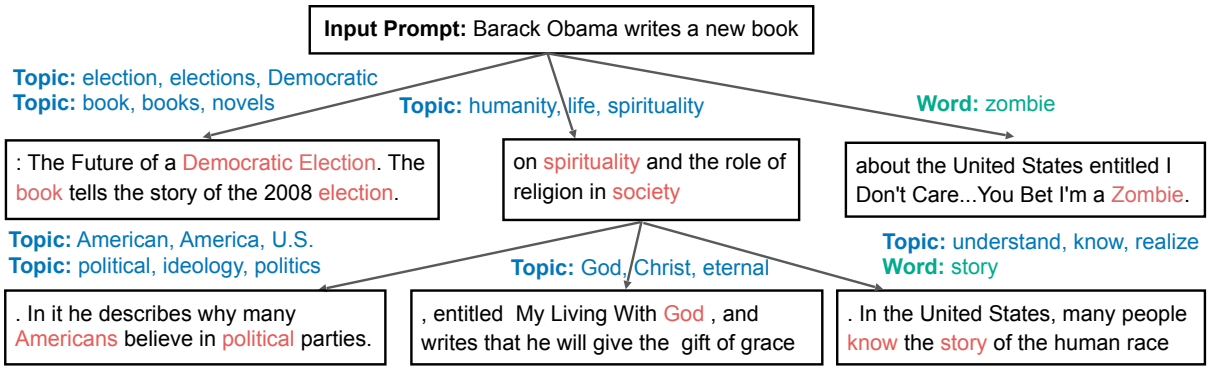

Figure 2: Examples of our generated options and continuations. We highlight the words in the continuation that are related to the chosen topics or to the specified word.

vide fine-grained control, and our models are self-supervised without the need to define the attributes or collect annotations. As depicted in Figure 1, a user can peek at the most probable K topics (shown as bags of words) appearing after the input prompt and control the generation by choosing the topics.

In Figure 2, we compare multiple generated sentences conditioned on different chosen topic(s) or specified word(s). For example, if the user chooses a topic about *humanity*, *life*, and *spirituality*, our system continues the input prompt "*Barack Obama writes a new book*" with "*on spirituality and the roles of religion in society*". Then, we can use the generated text as the new input prompt and update the set of topics to include other more relevant topics such as *God*, *Christ*, and *eternal*. The process can be repeated to create a plot tree.

A user can also control the generation by specifying word(s) if the user wants to see the words that are not in the topic list or seeks a transition to a word that is not directly related to the input prompt. For example, a user can ask our system to generate a sentence about *zombie*. Consequently, the continuation of *"Barack Obama writes a new book"* becomes *"about the United States entitled I Don't Care...You Bet I'm a Zombie"*.

The system is realized by two components: an option generator and a conditional text generator. Given a prompt, the option generator suggests a set of K topics. After a user chooses a subset of the topics and specifies some words, the embedding of every word or topic will guide the conditional text generator to produce the continuation that is both consistent with the existing prompt and relevant to the chosen topics and words.

Both components are self-supervised and use pretrained GPT2 models (Radford et al., 2019) to encode the input prompt. During training, the option generator predicts the cluster centers of future words, which are in the continuation of the prompt, based on the contextualized embeddings from GPT2. The conditional text generator fine-tunes GPT2 to predict the next words given the prompt and a few subsequent words. Since both components' input and output only come from the prompt and its continuation, training the system only requires a raw corpus, word tokenizers, and a list of stop words. This makes the proposed method suitable for open-domain story generation and easily being fine-tuned for a specific domain.

In experiments, we demonstrate that our system recommends high-quality topics and often generate sentences that follow the chosen topics. We compare our option generator with global topic models such as LDA (Blei et al., 2001) or local topic models such as clustering the words in the input prompt. The results show that the proposed method generates significantly more topics that are plausible and promote the narrative. Moreover, we compare our conditional text generator with PPLM (Plug and Play Language Models) (Dathathri et al., 2020) and demonstrate that our generation is more fluent and relevant to the chosen topics. Our code is available at https://github.com/iesl/interactive_LM.

2 Method

The proposed framework consists of two components: option generator and conditional text generator. In Figure 3, we illustrate the two components and their interaction. First, given the prompt $x_1, ..., x_I$ inputted by a user, the option generator at the bottom of the figure outputs K topics. After the user chooses two topics about *book* and *election* and specifies one extra word *story*, the topics

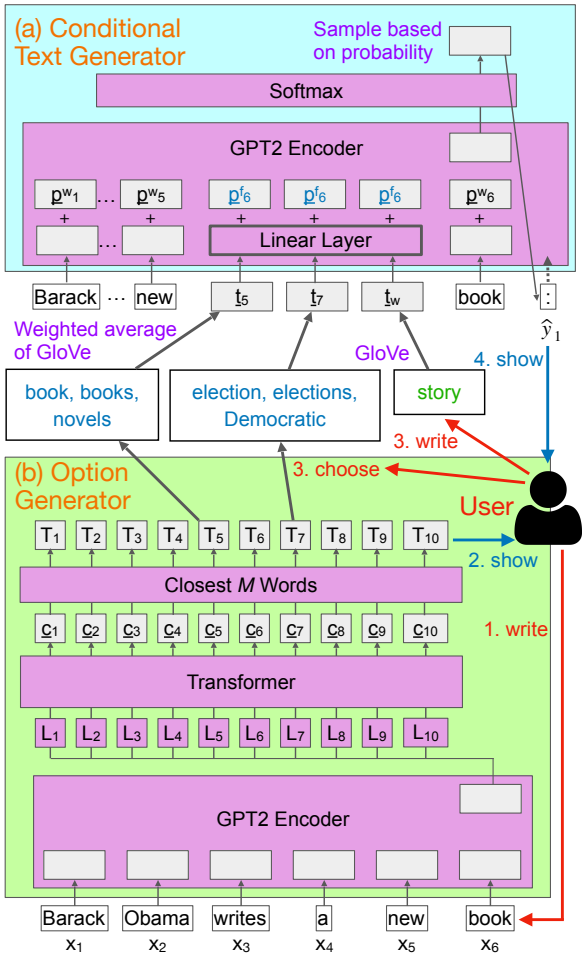

Figure 3: Our model architectures for (a) conditional text generator and (b) option generator. During testing, the information flows from the bottom to the top.

and word are passed to our text generator as the generation guidance. Accordingly, the generator continues to write the next token $\widehat{y}_1$.[1]

In the following subsections, we introduce our model designs and the way to train each component. More implementation details are described in the appendix.

2.1 Option Generator

When we do not have labeled attributes in a corpus, we can create options by clustering all the words in a corpus into topics (Tu et al., 2019). The clustering could be done by topic modeling approaches such as LDA (Blei et al., 2001). The resulting topics are static (i.e., the clustering is performed globally without considering the prompt). However, the prompt might have a narrow focus and the related words of interest are all clustered into a single topic.

A simple remedy is to cluster only the words in the prompt rather than all the words in the corpus. The topics are created dynamically and locally given a prompt and can capture more fine-grained aspects in the continuations. However, the topics derived from the prompt might provide less inspiration because the users have seen the prompt. Another major drawback of the approach is that the generated topics might encourage the LM to generate repetitive sentences or make a narrative circle inside a loop.

Motivated by the challenges, we propose an option generator that predicts the cluster centers based on the prompt instead of clustering the words in the prompt during testing.

2.1.1 Model Prediction

The goal of our option generator is to predict the K cluster centers of words in the possible continuations and use the cluster centers as the topics user could choose from. As in Figure 3 (b), the option generator uses GPT2 to encode the input prompt $x_1, ..., x_I$ and passes the output embedding to K different linear layers $L_1, ..., L_K$. To model the dependency of clusters, a Transformer (Vaswani et al., 2017) takes the K embeddings as input and predicts the cluster centers $\underline{c}_1, ...\underline{c}_K$ in GloVe (Pennington et al., 2014) space. During testing, each predicted cluster center is normalized by its L2 norm, and we use the M closest words in the normalized GloVe space to represent the topic T_i, which users can choose.

We choose to learn the cluster centers in GloVe space rather than GPT2 or BERT (Devlin et al., 2019) space because the non-contextualized word embeddings are easier to visualize. Users can easily understand the meaning of a cluster center by seeing nearby words. We normalize GloVe space in this work to make the squared L2 distance equal to twice the cosine distance between two embeddings.

Our architecture is similar to the one in Chang et al. (2021), but we use a pretrained GPT2 encoder rather than train a BERT-like Transformer from scratch. Another difference is that we ignore the connection between the second Transformer and the output of GPT2 to save GPU memory for handling a longer input prompt.

[1]The framework is flexible. For example, the GPT2 encoders in the two components could be shared. Besides topics, the option generator could be extended to predict likely attributes in the continuation such as positive sentiment and event frames (Tu et al., 2019) if the corresponding label data are available in the training corpus.

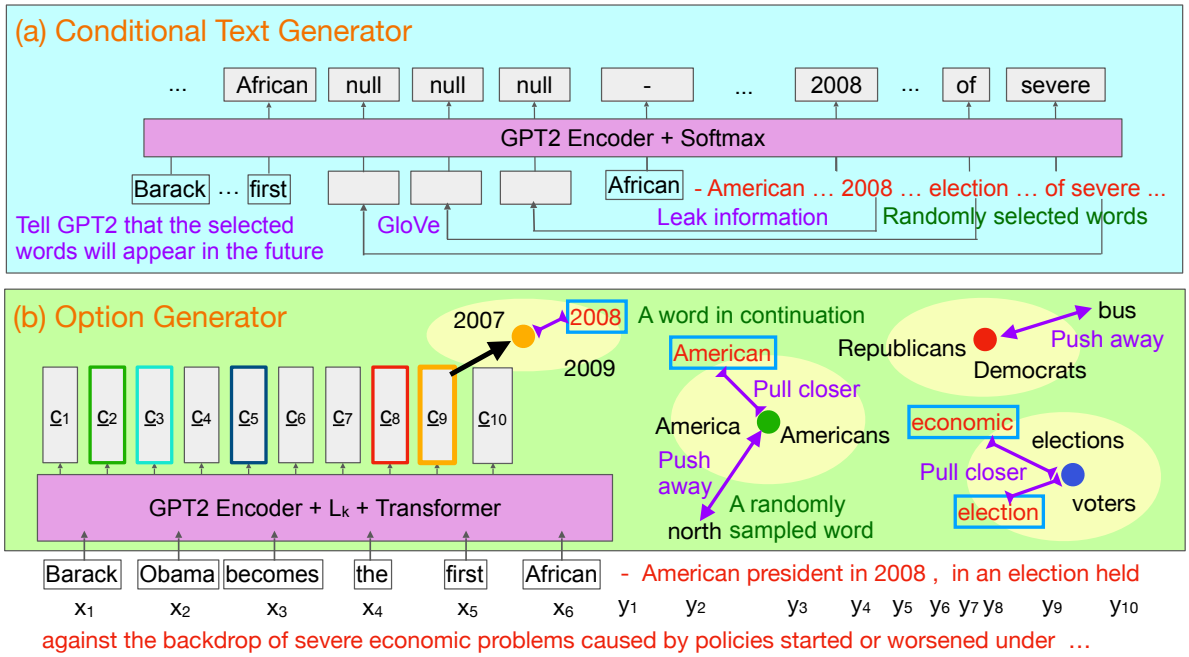

Figure 4: Training our two components using the same sentence. (a) We randomly pick $n = 3$ words in the actual continuation as our conditions for the text generator, and the null labels mean their predicted probabilities are ignored in our loss. (b) We visualize 5 out of $K = 10$ generated topics in a normalized GloVe space. Red words are the ones that appear in the continuation and pull the nearby cluster centers closer during training.

2.1.2 Model Training

In Figure 4 (b), we visualize our training procedure. For each input prompt in the training corpus, we run a forward pass through the Transformers and get predicted cluster centers $\underline{c}_1, ...\underline{c}_K$. Next, we collect 50 words in the continuation (except stop words) as positive examples and match the words with cluster centers as in the E-step of the EM algorithm (Dempster et al., 1977). We minimize the distances between the centers and their nearby positive examples by backpropagating the gradients through the matching and updating our Transformer models. Furthermore, we randomly sample some words as negative examples and maximize the distances between the cluster centers and their nearby embeddings from negative examples.

Using Figure 4 (b) as an example, the orange cluster center is pulled closer toward the embedding of *2008*, which appears in the continuation. The green cluster center is pushed away from the embedding of *north*, a randomly sampled word. Since each output embedding $\underline{c}_k$ is pulled by only the nearby embeddings of words in the continuation, the output embedding will naturally become the cluster center of the nearby continuation word embeddings. Notice that the related topics like *Democrats* and *Republicans* are not observed in the prompt and continuation, but our model can predict a red cluster center close to them because the model can learn from other similar input prompts whose continuation mentions words like *Democrats*.

Chang et al. (2021) discover that non-negative sparse coding (NNSC) (Hoyer, 2002) could encourage the Transformers to predict more diverse and relevant topics compared with Kmeans, so we adopt NNSC as our clustering loss, and its formulation could be found in Chang et al. (2021).

2.2 Conditional Text Generator

After the user chooses topic(s) or specifies word(s), each topic or word is converted to a GloVe embedding. The component aims to generate the text given the input prompt and the GloVe embeddings of the topics or words we prefer to see in the continuation.

Users only see the M words closest to the kth predicted cluster center $\underline{c}_k$ from our option generator, so we compute the kth topic embedding as

$$\underline{t}_k = \frac{\sum_{m=1}^{M} \cos(\underline{e}_m^w, \underline{c}_k)\underline{e}_m^w}{||\sum_{m=1}^{M} \cos(\underline{e}_m^w, \underline{c}_k)\underline{e}_m^w||}, \quad (1)$$

where $\underline{e}_m^w$ is the normalized GloVe embedding of the mth closet word and $\cos(\underline{e}_m^w, \underline{c}_k)$ is the cosine similarities between the mth word embedding and the embedding $\underline{c}_k$.

2.2.1 Model Prediction

During testing, the topic embeddings $\underline{t}_k$ or embedding of the specified words are inserted into GPT2 encoder before x_I, the last word piece in the prompt. The inserted embeddings nudge the GPT2 to generate the sentences containing the desired words with a higher probability.

As Figure 3 (a) shows, the GloVe embeddings are first passed through a linear layer to make their dimension become the same as the hidden state size of GPT2. Then, the transformed embeddings are added with special positional embeddings $\underline{p}_I^f$, which are different from those for the prompt $\underline{p}_i^w$. The special positional embedding tells GPT2 that the inserted embeddings have a different meaning and where the conditional generation starts.

The GPT2 encoder's output goes through a softmax layer, which computes the probability of each token being observed as the first word piece in the continuation y_1. We adopt top-k sampling (Fan et al., 2018), which reduces the chance of sampling words with low probability, to pick the next word, and autoregressively sample one token $\widehat{y}_o$ at a time to generate the continuation $\widehat{y}_1, ..., \widehat{y}_O$.

2.2.2 Model Training

We train the generator using the continuation of a prompt and some randomly selected non-stop words in the continuation as its generation conditions. Since the continuation contains the randomly-selected words, the generator would be heavily penalized if it ignores the conditions by assigning low probabilities to the selected words in all the continuation positions.

An example is illustrated in Figure 4 (a). Given an input prompt in the training set, we randomly pick a number n from 0 to K and sample n words from the next $O = 25$ words (except stop words). Next, the normalized GloVe embeddings of n words are inserted to the GPT2 encoder before the last word piece in the prompt, and we ignore the output probabilities corresponding to the inserted positions during training. To speed up the training, we conduct the future word insertion in multiple positions of each training text sequence.

We insert the future words just before the text that might contain the words rather than at the beginning as in the classic seq2seq model, because we do not want the model to learn to generate the continuation based on the future topics that have not yet be specified by the users (e.g., The GPT2 should not know that it will see *election* in the future when it learns to generate *Barack Obama ...* during training).

By allowing the LM to see the upcoming words earlier, we leak partial label information to the LM input. Consequently, GPT2 learns to utilize the information and generate the sentence containing the desired words to achieve a lower perplexity loss. Notice that the training method allows us to specify our topical preference without significantly scarifying generation efficiency and fluency, but it cannot guarantee to generate all the desired topics, especially when we specify multiple ones.

One concern of the method is that the LM cannot see all possible sets of topics or words users might specify during training. Besides, each GloVe embedding used to supervise LM comes from a single word, but we ask the LM to condition on average GloVe embedding of the top M words during testing. Nevertheless, we observe that the LM is often able to generalize well in our experiments because similar words have similar GloVe embeddings, lots of training instances could be easily prepared by the self-supervised method, and our option generator usually provides the topics mentioned in the continuation in our training corpus.

3 Experiments

We evaluate two components separately, and both evaluations include automated metrics and human judgment. Throughout the evaluation, the number of topics $K = 10$ and the length of generations is 50 word pieces. We find that fixing $K = 10$ works well in our experiments. If the possible continuations cover more than 10 topics, our option generator tends to output the important topics. If they cover fewer topics, our option generator tends to output the related topics that are not explicitly mentioned in the prompt or the duplicated topics. More experiment setup details could be found in the appendix.

3.1 Datasets

We use 90% of English Wikipedia 2016 as our training set for both components, 5% as our validation set to determine the hyperparameters such as the number of epochs, and the remaining 5% as our test set to perform the automated evaluation.

For human evaluation, we collect labels from Amazon Mechanical Turk (MTurk). We randomly sample sentences from the training set of STS benchmark (STSb) (Cer et al., 2017) as our input

prompts. Compared with Wikipedia, the sentences from STSb are easier to understand for annotators because a large portion of sentences in Wikipedia involves terminologies, depends on a longer context, or might even just be a list of names.

In STSb, we sample 24 sentences as our prompts, and each method generates one continuation for each input prompt. Each generated continuation or topics will be scored by three different workers.

3.2 Option Generator Evaluation

We evaluate the topics from different option generators by judging whether the topics will appear in the continuation and whether the topics would promote the narrative. The goal is to have topics that are relevant and provide new information. The topics that are too similar to the prompt words might be redundant and not helpful because the users have already seen the prompt.

3.2.1 Automatic Evaluation Metrics

- **Sim**: If the generated topics T can help users to write the continuation, the embedding of every non-stop word in the actual continuation should be similar to the embeddings of a generated topic. Thus, we compute

$$\text{Sim}(\bar{Y}, T) = \sum_{o=1}^{O'} \max_{k=1}^{K} (\underline{t}_k)^T \underline{e}_o^{\bar{y}}, \qquad (2)$$

  where $\bar{Y} = \{\bar{y}_o\}_{o=1}^{O'}$ is a set of non-stop words in the continuation and $O' = 25$. $\underline{t}_k$ is the normalized embedding of kth topic in T from equation 1 and $\underline{e}_o^{\bar{y}}$ is the oth word in $\bar{Y}$.
- **Sim Short**: When computing Sim, we use the input prompts containing around 180 words on average. To examine the topic quality at the start of writing, where the authors might need assistance the most, we also report $\text{Sim}(\bar{Y}, T)$ on short input prompts (with 35 words on average).
- **Sim Diff**: The options that are helpful to users should be sufficiently different from the words in the input prompt to promote the narrative and avoid generating repeated content. Thereby, we also evaluate methods using Sim Diff = $\text{Sim}(\bar{Y}, T)$ - $\text{Sim}(\bar{X}, T)$, where $\bar{X} = \{\bar{x}_i\}_{i=1}^{I'}$ are the non-stop words in the input prompt.

3.2.2 Human Evaluation

Our questionnaire shows the prompt and asks which generated topics are likely to appear in a reasonable continuation and which topics promote the narrative. For each method, we report the average number of its topics that are likely to appear (L), promote the topic (TP), and both (L&TP). For example, an MTurk worker is shown three topics generated by a method given a prompt: ABC. The worker thinks A is likely to appear in the continuation and AB promote the topic. Then, L=$|\{A\}|$=1, TP=$|\{AB\}|$=2, and L&TP=$|\{A\} \cap \{AB\}|$=$|\{A\}|$=1 for this prompt.

Scope	Method	Sim	Sim Short	Sim Diff
Global	Sample	14.63	14.42	**0.16**
	LDA	36.86	36.02	-2.82
	Kmeans	**40.65**	**39.91**	-3.40
Local	Sample	41.50	41.23	-12.51
	NNSC	46.70	42.80	-15.94
	Kmeans	47.94	43.89	-16.12
	Ours	**48.38**	**46.29**	**0.45**

Table 1: Comparison of the option generators using automatic metrics. The best numbers within each scope are highlighted.

Scope	Method	L	TP	L&TP
Global	LDA	5.76 ± 0.50	6.24 ± 0.33	5.26 ± 0.31
	Kmeans	6.94 ± 0.36	6.13 ± 0.30	5.96 ± 0.31
Local	Kmeans	$\mathbf{8.65} \pm 0.16$	5.31 ± 0.50	5.14 ± 0.50
	Ours	7.85 ± 0.25	$\mathbf{6.96} \pm 0.26$	$\mathbf{6.75} \pm 0.28$

Table 2: Comparison of option generators using human judgment (mean $\pm$ standard error). L and TP refer to likelihood and topic promotion, respectively.

3.2.3 Option Generator Baselines

We compare our generator with two types of methods.[2] The first type performs the clustering globally and selects the most relevant topics to the input prompt from the static set of clusters. We cluster all the words into $J = 150$ topics by LDA (Blei et al., 2001) (**LDA-global**) and into $J = 1000$ topics by Kmeans on the normalized GloVe embedding space (Tu et al., 2019) (**Kmeans-global**). We also randomly sample K words from the whole vocabulary as our cluster centers (**Sample-global**).

Similar to equation 1, we find the M words with the closest embeddings to each cluster center to represent the topic and compute the topic embedding $\underline{t}_j$ as the weighted average embedding of M words in the jth topic. Among all J cluster centers, we pick the K topics with the closest $\underline{t}_j$ to the

[2] Another alternative is to generate many continuations and cluster the words in the generation. However, the method takes time, which might be prohibited by limited computational resources and the real-time interaction requirement.

Input Prompt				The study also found that skin cancer nearly tripled in Norway and Sweden since the 1950s.							
LDA-global				Kmeans-local				Ours			
1	population, households	6	company, companies	1	Norway, Sweden	6	also, however	1	research, scientific	6	1980s, 1970s
2	patients, treatment	7	Norwegian, Norway	2	tripled, doubled	7	since, Since	2	tissues, tissue	7	even, though
3	psychology, research	8	story, book	3	nearly, almost	8	Sweden, Finland	3	patients, diagnosis	8	susceptibility, pathogenic
4	police, prison	9	hospital, Hospital	4	cancer, skin	9	study, studies	4	DNA, gene	9	decreased, increased
5	chemical, carbon	10	Icelandic, Iceland	5	1950s, 1940s	10	found, discovered	5	orange, purple	10	Sweden, Norway

Table 3: Comparison of all K topics for the input prompt using $M = 2$ words closest to each topic.

Input Prompt		The study also found that skin cancer nearly tripled in Norway and Sweden since the 1950s.
Generator		Generated Text
Option	Text	
LDA-global	Ours	A study of the Norwegian police has confirmed the cancer case. The law in Norway was the subject of the
Kmeans-local	Ours	The study also found that skin cancer nearly tripled in Norway and Sweden since the 1950s. As well, skin
Ours	PPLM	In this study, a study was conducted conducted in Italy and in Finland. From the 1990s to the 1970s, there
None	GPT2	The study also revealed that only 20% of the deaths in Norway were caused by a sudden cardiac response
Ours	Ours	Recent studies have shown that melanin causes a decrease in genetic susceptibility in people in Norway,

Table 4: The continuations that are generated by conditioning on all of K topics from different option generators. The input prompt comes from STSb.

prompt embedding, where the prompt embedding is the average embedding of all words in the input prompt.

The second type of methods discovers the K topics from the input prompt. We cluster non-stop words in the prompt using non-negative sparse coding (Hoyer, 2002) (**NNSC-local**) and Kmeans (**Kmeans-local**). We also sample K non-stop words from the prompt and call it **Sample-local**. Similar to equation 1, we represent each topic using M words and compute the weighted average of their embeddings $\underline{t}_k$ as the input of our text generator. Notice that the locally clustering methods produce similar results when the prompts come from STSb due to their short lengths, so we only test **Kmeans-local** in our human evaluation.

3.2.4 Results

In Table 1, we show that local methods generate the options more relevant to the input prompt than the global methods due to significantly higher Sim and Sim Short. Our method performs better compared to other local methods, especially in Sim Diff, which highlights the high novelty of our generated topics. The improvement on Sim Short is larger than that on Sim because our method could suggest the related topics that are not explicitly mentioned in the short prompt (e.g., *U.S.* in Figure 1).

The human evaluation results are presented in Table 2. Our method wins in terms of generating relevant topics that promote the narrative. The **Kmeans-local** performs better in L because most of the words in the input prompts could be mentioned again in the next sentence. However, it often leads to the redundant topics that are too similar to the prompt.

Table 3 compares the options generated by different methods while Table 4 compares the text generated using different option generators and text generators. More examples are presented in the appendix. In Table 3, we can see that most topics in **Kmeans-local** do not promote the narrative, which makes the generated continuation become a copy of the input prompt in Table 4. We will quantitatively evaluate the generated continuations using different option generators in the appendix. Notice that the high redundancy problem is hard to be solved by a conditional text generator because the relatedness between the prompt and the generated text is hard to be controlled (See et al., 2019b).

3.3 Conditional Text Generator Evaluation

To demonstrate our text generator's effectiveness, we use our option generator to prepare the topic embeddings and randomly select n topics as our conditions to simulate the user's choice, where n is a random number from 1 to K. The sentences generated by different methods are compared.

3.3.1 Automatic Evaluation Metrics

We match the union of $M \times K$ top words in the chosen topics with the words in the generated continuations and count the number of tokens that are matched exactly (token), the number of matched word types (word), and the number of topics that contain at least one matched word (topic) to measure the relevancy between the continuations and the chosen topics. Notice that the scores are underestimated because the generation might mention words in different morphological variations or other

Text Generation Method	Automatic Metrics						Inference Time	Human Judgement		
	Relevancy Hit			Quality				Relevancy		Fluency
	Token	Word	Topic	PPL (↓)	Dist-1	Dist-2	s (↓)	Recall	Precision	Score
PPLM	1.48	0.99	0.77	18.49	**40.29**	**80.83**	17.74	30.56 ± 2.96	56.01 ± 4.41	3.83 ± 0.13
Ours	**2.36**	**1.79**	**1.40**	**16.39**	37.98	79.65	**1.02**	**41.46** ± 3.47	**56.41** ± 4.41	**4.07** ± 0.10
GPT2	1.27	0.84	0.64	14.24	39.80	80.22	1.00	24.49 ± 2.77	48.69 ± 4.61	4.15 ± 0.11

Table 5: Comparison of conditional text generators. The numbers in Dist-1, Dist-2, Recall, and Precision are percentages. Lower perplexity (PPL) and inference time are better. The better performances between PPLM and our method are highlighted. In human evaluation, we report the mean ± standard error of each method.

words related to the topics.

The fluency of the generated text is measured using the perplexity (Serban et al., 2016) of the original GPT2 (with 345M parameters) without being fine-tuned on Wikipedia. Dist-n (Li et al., 2016) is the ratio between the number of unique n-grams and the number of all n-grams in the continuations, where n=1 or 2. Higher Dist-n implies more diverse generations. The average inference time per input prompt is also presented.

3.3.2 Human Evaluation

We present the prompt and the generated continuation and ask the worker to score the generation's fluency from 1 (not fluent at all) to 5 (very fluent). Next, we show K topics and ask which topics are mentioned in the generation. Treating the worker's choices as prediction and the topics our model conditions on as ground truth, we report the average precision and recall of the prediction.

3.3.3 Conditional Text Generator Baselines

We compare our method with **PPLM** (Plug and Play Language Models) (Dathathri et al., 2020) due to its strong performance against the weighted decoding approach from Ghazvininejad et al. (2017) when the condition is a bag of words.

The condition for **PPLM** is the union of the top M words in the chosen topics and each word's weight is neglected. We use our generation model without conditioning on any word (i.e., $n = 0$) during testing[3] as the base model of **PPLM**. We also present the performance of the base model itself as a reference to know the significance of our improvement (denoted as **GPT2**).

3.3.4 Results

Table 5 indicates that our model outperforms **PPLM** in all metrics except in Dist-1 and Dist-2. We suspect that our model generates slightly less diverse sentences in order to make the generation more relevant to the given topics.

The generation might mention a topic even if it is not chosen as a condition, so we achieve similar precision compared to **PPLM** in human evaluation. The recall of **PPLM** means that only around 30% of given topics are mentioned. The low recall indicates the difficulty of mentioning multiple randomly selected topics in the next 50 word pieces while keeping the sentence fluent. By contrast, achieving 40% on recall demonstrates the effectiveness of our conditional text generator.

Compared with **PPLM**, our model requires an additional training step but achieves low inference time and high relevancy to the given topics/words once the training is finished. The benefits make it preferable in our interactive writing application.

4 Related Work

Different interactive writing assistants provide different forms of options to let users express their preferences. The options could be manually defined classes (e.g., sentiment) (Keskar et al., 2019; Dathathri et al., 2020), semantic frames (Tu et al., 2019), or event structures such as (subject, verb, object, modifier) (Martin et al., 2018; Tambwekar et al., 2019; Ammanabrolu et al., 2020). The forms of options allow users to control the attributes of the generated text but require labels or classifiers that map the text to the attributes/options.

The options could also be a single query word at the beginning (Austin, 2019), the article title (Yan, 2016), politeness (Niu and Bansal, 2018) or specificity (See et al., 2019b) of the text, or the length of the generated sentence (Tu et al., 2019). However, the options cannot provide fine-grained control on topical directions of the generated contents.

A related research direction is the multi-stage story generation. To make a long story more coherent, recent work proposes to generate a skeleton and then generate the full text guided by

[3] We find the model performs similarly compared with the GPT2 with no condition during training.

the skeleton. The skeleton could be a sequence of SRL frames (Fan et al., 2019), a sequence of event structure (subject, verb, object, preposition, modifier) (Ammanabrolu et al., 2020), a story premise (Fan et al., 2018), or a story summary (Chen et al., 2019). Users can revise the skeleton to control the generated text, but the approaches assume the existence of the skeleton extractor or labels in the training corpus. Besides, the systems cannot suggest options given the partial text, which is one of the main focuses of our interactive writing assistant.

The skeleton could also be multiple keyphrases. The keyphrases are extracted based on word frequency (Ippolito et al., 2019; Tan et al., 2020; Wu et al., 2020), an off-the-shelf keyword extraction method (Peng et al., 2018; Goldfarb-Tarrant et al., 2019; Yao et al., 2019; Rashkin et al., 2020; Zhang et al., 2020), a sentence compression dataset and reinforcement learning (Xu et al., 2018), or image caption datasets and ConceptNet (Lin et al., 2020). Most of the studies focus on modeling the long-term dependency among the keyphrases and/or forcing the generation to contain the keyphrases. Instead, we focus on allowing users to determine the topical directions of the generation. Compared with conditioning on keyphrases, our interactive writing assistant is especially helpful when users do not know the exact phrases they want to see or when the given keyphrase extractor does not detect the desired topics.

5 Conclusion

We propose an interactive writing assistant that generates topic options given an input prompt and generates the continuation of the prompt given the topics chosen by a user. We decompose the framework into two components and propose a novel model for each component. The automated evaluation and human evaluation indicate that our system generates many topics that are related to but different from the prompt, and generates the sentences that are fluent and relevant to the chosen topics.

Acknowledgements

We thank Ao Liu for his preliminary exploration of this project and Nader Akoury for his helpful feedbacks. We also thank the anonymous reviewers for their constructive feedback.

This work was supported in part by the Center for Data Science and the Center for Intelligent Information Retrieval, in part by the Chan Zuckerberg Initiative under the project Scientific Knowledge Base Construction, in part using high performance computing equipment obtained under a grant from the Collaborative R&D Fund managed by the Massachusetts Technology Collaborative, in part by the National Science Foundation (NSF) grant numbers DMR-1534431 and IIS-1514053.

Any opinions, findings, conclusions, or recommendations expressed in this material are those of the authors and do not necessarily reflect those of the sponsor.

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.

Prithviraj Ammanabrolu, Ethan Tien, Wesley Cheung, Zhaochen Luo, William Ma, Lara J. Martin, and Mark O. Riedl. 2020. Story realization: Expanding plot events into sentences. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, New York, NY, USA, February 7-12, 2020*, pages 7375–7382. AAAI Press.

John Austin. 2019. The book of endless history: Authorial use of GPT2 for interactive storytelling. In *Interactive Storytelling - 12th International Conference on Interactive Digital Storytelling, ICIDS 2019, Little Cottonwood Canyon, UT, USA, November 19-22, 2019, Proceedings*, volume 11869, pages 429–432. Springer.

David M. Blei, Andrew Y. Ng, and Michael I. Jordan. 2001. Latent dirichlet allocation. In *Advances in Neural Information Processing Systems 14 [Neural Information Processing Systems: Natural and Synthetic, NIPS 2001, December 3-8, 2001, Vancouver, British Columbia, Canada]*, pages 601–608. MIT Press.

Daniel Cer, Mona Diab, Eneko Agirre, Iñigo Lopez-Gazpio, and Lucia Specia. 2017. SemEval-2017 task 1: Semantic textual similarity multilingual and crosslingual focused evaluation. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, pages 1–14, Vancouver, Canada. Association for Computational Linguistics.

Haw-Shiuan Chang, Amol Agrawal, and Andrew McCallum. 2021. Extending multi-sense word embedding to phrases and sentences for unsupervised semantic applications. In *Proceedings of the Twenty-Seventh AAAI Conference on Artificial Intelligence*.

Gang Chen, Yang Liu, Huanbo Luan, Meng Zhang, Qun Liu, and Maosong Sun. 2019. Learning to predict explainable plots for neural story generation. *arXiv preprint arXiv:1912.02395*.

Elizabeth Clark, Anne Spencer Ross, Chenhao Tan, Yangfeng Ji, and Noah A Smith. 2018. Creative writing with a machine in the loop: Case studies on slogans and stories. In *23rd International Conference on Intelligent User Interfaces*.

Zihang Dai, Zhilin Yang, Yiming Yang, Jaime Carbonell, Quoc Le, and Ruslan Salakhutdinov. 2019. Transformer-XL: Attentive language models beyond a fixed-length context. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2978–2988, Florence, Italy. Association for Computational Linguistics.

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. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net.

Arthur P Dempster, Nan M Laird, and Donald B Rubin. 1977. Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society: Series B (Methodological)*, 39(1):1–22.

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

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

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

Marjan Ghazvininejad, Xing Shi, Jay Priyadarshi, and Kevin Knight. 2017. Hafez: an interactive poetry generation system. In *Proceedings of ACL 2017, System Demonstrations*, pages 43–48, Vancouver, Canada. Association for Computational Linguistics.

Seraphina Goldfarb-Tarrant, Haining Feng, and Nanyun Peng. 2019. Plan, write, and revise: an interactive system for open-domain story generation. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics (Demonstrations)*, pages 89–97, Minneapolis, Minnesota. Association for Computational Linguistics.

Patrik O Hoyer. 2002. Non-negative sparse coding. In *Proceedings of the 12th IEEE Workshop on Neural Networks for Signal Processing*.

Daphne Ippolito, David Grangier, Chris Callison-Burch, and Douglas Eck. 2019. Unsupervised hierarchical story infilling. In *Proceedings of the First Workshop on Narrative Understanding*, pages 37–43, Minneapolis, Minnesota. Association for Computational Linguistics.

Nitish Shirish Keskar, Bryan McCann, Lav R Varshney, Caiming Xiong, and Richard Socher. 2019. CTRL: A conditional transformer language model for controllable generation. *arXiv preprint arXiv:1909.05858*.

Boyang Li, Stephen Lee-Urban, George Johnston, and Mark Riedl. 2013. Story generation with crowdsourced plot graphs. In *Proceedings of the Twenty-Seventh AAAI Conference on Artificial Intelligence, July 14-18, 2013, Bellevue, Washington, USA*. AAAI Press.

Jiwei Li, Michel Galley, Chris Brockett, Jianfeng Gao, and Bill Dolan. 2016. A diversity-promoting objective function for neural conversation models. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 110–119, San Diego, California. Association for Computational Linguistics.

Bill Yuchen Lin, Wangchunshu Zhou, Ming Shen, Pei Zhou, Chandra Bhagavatula, Yejin Choi, and Xiang Ren. 2020. CommonGen: A constrained text generation challenge for generative commonsense reasoning. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 1823–1840, Online. Association for Computational Linguistics.

Linbo Luo, Wentong Cai, Suiping Zhou, Michael Lees, and Haiyan Yin. 2015. A review of interactive narrative systems and technologies: a training perspective. *Simulation*, 91(2):126–147.

Lara J. Martin, Prithviraj Ammanabrolu, Xinyu Wang, William Hancock, Shruti Singh, Brent Harrison, and Mark O. Riedl. 2018. Event representations for automated story generation with deep neural nets. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI-18), New Orleans, Louisiana, USA, February 2-7, 2018*, pages 868–875. AAAI Press.

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

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

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. GloVe: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, Doha, Qatar. Association for Computational Linguistics.

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

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.

Melissa Roemmele and Andrew S. Gordon. 2015. Creative help: A story writing assistant. In *Interactive Storytelling - 8th International Conference on Interactive Digital Storytelling, ICIDS 2015, Copenhagen, Denmark, November 30 - December 4, 2015, Proceedings*, volume 9445, pages 81–92. Springer.

Abigail See, Aneesh Pappu, Rohun Saxena, Akhila Yerukola, and Christopher D. Manning. 2019a. Do massively pretrained language models make better storytellers? In *Proceedings of the 23rd Conference on Computational Natural Language Learning (CoNLL)*, pages 843–861, Hong Kong, China. Association for Computational Linguistics.

Abigail See, Stephen Roller, Douwe Kiela, and Jason Weston. 2019b. What makes a good conversation? how controllable attributes affect human judgments. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 1702–1723, Minneapolis, Minnesota. Association for Computational Linguistics.

Iulian Vlad Serban, Alessandro Sordoni, Yoshua Bengio, Aaron C. Courville, and Joelle Pineau. 2016. Building end-to-end dialogue systems using generative hierarchical neural network models. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence, February 12-17, 2016, Phoenix, Arizona, USA*, pages 3776–3784. AAAI Press.

Pradyumna Tambwekar, Murtaza Dhuliawala, Lara J. Martin, Animesh Mehta, Brent Harrison, and Mark O. Riedl. 2019. Controllable neural story plot generation via reward shaping. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI 2019, Macao, China, August 10-16, 2019*, pages 5982–5988. ijcai.org.

Bowen Tan, Zichao Yang, Maruan AI-Shedivat, Eric P Xing, and Zhiting Hu. 2020. Progressive generation of long text. *arXiv preprint arXiv:2006.15720*.

Lifu Tu, Xiaoan Ding, Dong Yu, and Kevin Gimpel. 2019. Generating diverse story continuations with controllable semantics. In *Proceedings of the 3rd Workshop on Neural Generation and Translation*, pages 44–58, Hong Kong. Association for Computational Linguistics.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pages 5998–6008.

Nick Walton. 2020. AI dungeon.

Zeqiu Wu, Michel Galley, Chris Brockett, Yizhe Zhang, Xiang Gao, Chris Quirk, Rik Koncel-Kedziorski, Jianfeng Gao, Hannaneh Hajishirzi, Mari Ostendorf, et al. 2020. A controllable model of grounded response generation. *arXiv preprint arXiv:2005.00613*.

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

Rui Yan. 2016. i, poet: Automatic poetry composition through recurrent neural networks with iterative polishing schema. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence, IJCAI 2016, New York, NY, USA, 9-15 July 2016*, pages 2238–2244. IJCAI/AAAI Press.

Lili Yao, Nanyun Peng, Ralph M. Weischedel, Kevin Knight, Dongyan Zhao, and Rui Yan. 2019. Plan-and-write: Towards better automatic storytelling. In *The Thirty-Third AAAI Conference on Artificial Intelligence, AAAI 2019, Honolulu, Hawaii, USA, January 27 - February 1, 2019*, pages 7378–7385. AAAI Press.

Yizhe Zhang, Guoyin Wang, Chunyuan Li, Zhe Gan, Chris Brockett, and Bill Dolan. 2020. POINTER: constrained progressive text generation via insertion-based generative pre-training. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing, EMNLP 2020, Online, November 16-20, 2020*, pages 8649–8670. Association for Computational Linguistics.