A-TIP: Attribute-aware Text Infilling via Pre-trained Language Model

Dongyuan Li, Jingyi You, Kotaro Funakoshi, Manabu Okumura
Tokyo Institute of Technology
{lidy,youjy,funakoshi,oku}@lr.pi.titech.ac.jp

Abstract

Text infilling aims to restore incomplete texts by filling in blanks, which has attracted more attention recently because of its wide application in ancient text restoration and text rewriting. However, attribute-aware text infilling is yet to be explored, and existing methods seldom focus on the infilling length of each blank or the number/location of blanks. In this paper, we propose an **A**ttribute-aware **T**ext **I**nfilling method via a **P**re-trained language model (A-TIP), which contains a text infilling component and a plug-and-play discriminator. Specifically, we first design a unified text infilling component with modified attention mechanisms and intra- and inter-blank positional encoding to better perceive the number of blanks and the infilling length for each blank. Then, we propose a plug-and-play discriminator to guide generation towards the direction of improving attribute relevance without decreasing text fluency. Finally, automatic and human evaluations on three open-source datasets indicate that A-TIP achieves state-of-the-art performance compared with all baselines.

1 Introduction

Originating from Cloze tests (Taylor, 1953), text infilling aims to fill in missing blanks in a sentence or paragraph by making use of the preceding and subsequent texts. For example, given two infilling tasks E1 and E2 in Fig.1, text infilling models are supposed to provide fine-grained control over the location of any number of blanks and infill a variable number of missing tokens for each blank. Text infilling has been gaining increasing attention in a number of prevailing research fields, including ancient text restoration (Lazar et al., 2021), text editing and rewriting (Su et al., 2021), and conversation generation (Ou et al., 2021).

However, current text infilling methods are based only on bidirectional semantic constraints (Ou et al., 2021), and other abundant attribute-based constraints, e.g., sentiment and topics, remain to be studied. In reality, infilling attribute-aware content can better satisfy human needs and introduce more diversity. For instance, as shown in Fig.1, A-TIP can fill in blanks under the guidance of an attribute to satisfy sentiment or expert knowledge infilling, while current text infilling models mainly focus on fluency, which leads to meaningless and monotonous infilling contents (Sun et al., 2021).

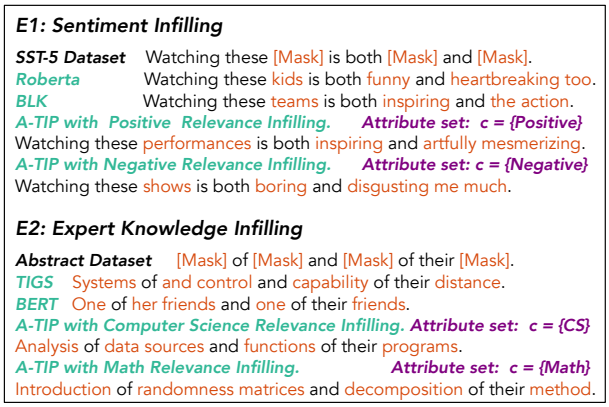

Figure 1: A-TIP can generate more fluent, diverse and attribute relevant infilling content in two examples.

Designing a simple but efficient attribute-aware text infilling model is a challenging task. First, to achieve attribute awareness, simply modifying a text infilling model architecture or fine-tuning with attribute-specific data will destroy the model's ability to infill blanks or require a significant cost for re-training (Dathathri et al., 2020). Second, if the model infills blanks towards the direction of improving text attributes, avoiding ill-formedness between infilling content and its bidirectional context becomes a challenge. For instance, "The movie interesting and perfect us" with _ as blanks. Finally, current methods lack fine-grained control over automatic determination of the number/location of blanks or the infilling length for each blank. For example, Markov assumption-based models (Liu et al., 2019; Zaidi

Proceedings of the 29th International Conference on Computational Linguistics, pages 5857–5869
October 12–17, 2022.

et al., 2020) hardly adapt to variable infilling lengths, while masked language model (MLM)-based methods (Devlin et al., 2019; Liu et al., 2020) are incapable of generating more than one word per blank, and generative LM-based methods (Donahue et al., 2020) cannot guarantee the output will match the number of missing blanks in the input.

To circumvent the above dilemma, in this paper, we propose an **A**ttribute-aware **T**ext **I**nfilling method based on a **P**re-trained LM (A-TIP), in which a plug-and-play discriminator provides fine-grained control over bidirectional well-formed fluency and attribute relevance.[1] Specifically, 1) we first propose a general text filling framework that fine-tunes a standard LM with many artificially-masked examples in an auto-regressive manner. Moreover, to ensure that the number of infilling contents equals the number of blanks, we design a new attention mechanism, where unmasked tokens can attend to each other but masked tokens can attend only to the preceding context (Fig.2 (A)). We also adopt two-level positional encoding to combine inter- and intra-blank positional information to automatically learn the length of blanks. 2) To achieve attribute-aware generation without modifying LM's architecture or re-training, we propose a plug-and-play discriminator that shifts the output distribution of the text infilling model towards the semantic space of given guide attributes. We also design two additional strategies to ensure the infilling content is well-formed with its bidirectional context without decreasing attribute relevance. The main contributions are summarized as follows:

• We propose a unified text infilling model that adopts a new attention mechanism and two-level positional encoding to enable our model to learn the number/location of blanks and infilling length for each blank automatically.

• To the best of our knowledge, A-TIP is the first attribute-aware text infilling model that does not require any modification of the language model's architecture or re-training on specific attributed datasets. Further, our plug-and-play discriminator can provide fine-grained control over fluency and attribute relevance, and can be applied to any transformer decoder-based text infilling model.

• The experimental results on three open datasets show that A-TIP achieves state-of-the-art performance compared with all baselines.

[1]Sentences with higher accuracy of attribute-based classification are said to have higher attribute relevance (Dathathri et al., 2020).

2 Related Work

In this section, we briefly review the most relevant studies to our work on pre-trained LMs, text infilling, and constrained text generation.

2.1 Pre-trained Language Models

Pre-trained LMs have made significant improvements in many natural language processing tasks by adopting self-supervised learning with abundant web texts (Chay-intr et al., 2021; You et al., 2022). They can be classified into three types. The first uses an auto-encoding model. For example, BERT (Devlin et al., 2019) and its variations are pre-trained as masked LMs to obtain bidirectional contextualized word representations. The second adopts an encoder-decoder architecture, which is pre-trained for seq2seq tasks, such as MASS (Song et al., 2019) and T5 (Raffel et al., 2020). The third adopts an auto-regressive model, which follows a left-to-right manner for text generation, such as GPT-2 (Radford et al., 2019) and XLNet (Yang et al., 2019). While we adopt GPT-2 as the LM in this paper, our method can be easily migrated to any type of pre-trained LMs.

2.2 Text Infilling Approaches

Current text infilling algorithms can be classified into four categories. *Generative adversarial networks (GAN)-based methods* train GANs to ensure that the generator can generate highly reliable infilling content to fool the discriminator (Guo et al., 2018; Fedus et al., 2018). *Intricate inference-based methods* adopt dynamic programming or gradient search to find infilling content that has a high likelihood within its surrounding context (Liu et al., 2019; Zaidi et al., 2020). *Masked LM-based methods* generate infilling content on the basis of its bidirectional contextual word embedding (Devlin et al., 2019; Shen et al., 2020). *LM-based methods* fine-tune off-the-shelf LMs in an auto-regressive manner, and a number of methods change the input format by putting an infilling answer after the masked input (Donahue et al., 2020), while others do not change the input format (Zhu et al., 2019). Unlike the aforementioned methods, we solve a more complex task: attribute-aware text infilling.

2.3 Constrained Text Generation

Traditional controlled generation models involve either fine-tuning existing models (He, 2021) or training conditional generative models (Keskar et al.,

2019). Dathathri et al. (2020) proposed a plug-and-play controlled generation model (PPLM), which does not modify or re-train the parameters of the original LM but can achieve comparable performance to fine-tuning methods. For example, PPCM (Madotto et al., 2020) updates the hidden state towards the direction of attribute enhancement to generate attribute-aware conversations. Pascual et al. (2021) designed a complex plug-and-play architecture to ensure that the generated content contains specific keywords. While GeDi (Krause et al., 2021) and its extension (Lin and Riedl, 2021) can accelerate the decoding process of PPLM, they assume the model is trained by large-scale labeled datasets, which is unrealizable for text infilling. Unlike the previous work, we should also consider the generated infilling content is well-formed with its corresponding bidirectional context, ensuring PPLM is suitable for text infilling.

3 Preliminaries

To clarify our method, we first introduce some essential background knowledge and then define the task of attribute-aware text infilling.

Language Models reveal the degree of how much a sentence (a sequence of words) is likely to be a realistic sequence of a human language. Formally, let $\mathcal{W}$ be the vocabulary set and $\boldsymbol{w}_{1:n} = \{w_1, \ldots, w_n\}$ is a sentence with n words, where $w_i \in \mathcal{W}$. An LM measures the joint probability by decomposing the sequence one by one:

$$p(\boldsymbol{w}_{1:n}) = \prod_{i=1}^{n} p(w_i|\boldsymbol{w}_{<i}), \tag{1}$$

where $\boldsymbol{w}_{<i} = \{w_1, \ldots, w_{i-1}\}$.

Constrained Text Generation: Given k explicit constraints $\mathbf{c} = \{c_1, \ldots, c_k\}$, our goal is to generate a sentence $\boldsymbol{w}$ that maximizes the conditional probability $p(\boldsymbol{w}|\mathbf{c})$:

$$p(\boldsymbol{w}|\mathbf{c}) = \prod_{i=1}^{n} p(w_i|\boldsymbol{w}_{<i}, \mathbf{c}). \tag{2}$$

Task Definition: Attribute-aware text infilling is to take incomplete text $\widetilde{\boldsymbol{w}}$, containing one or more missing blanks, and return completed text $\boldsymbol{w}$ under the constraints of $\mathbf{c}$. As in Fig.1, several attributes are listed in $\mathbf{c}$. Specifically, let [Mask] be a placeholder for a contiguous sequence of one or more missing tokens. Then, $\widetilde{\boldsymbol{w}}$ is a sequence of tokens in which a number of them are [Mask]. To map $\widetilde{\boldsymbol{w}}$ to $\boldsymbol{w}$, constrained with attribute $\mathbf{c}$, an infilling strategy must specify both how many and which tokens to generate for each [Mask]. Note that there may be many logical $\boldsymbol{w}$ for a given $\widetilde{\boldsymbol{w}}$. Hence, we are interested in learning a distribution $p(\boldsymbol{w}|\widetilde{\boldsymbol{w}}, \mathbf{c})$. Specifically, in accordance with Bayes' theorem, we formulate the probability of predicting the token w_i for its corresponding [Mask] as:

$$p(w_i|\boldsymbol{w}_{<i}, \mathbf{c}) \propto p(w_i|\boldsymbol{w}_{<i}) \cdot p(\mathbf{c}|\boldsymbol{w}_{1:i}), \tag{3}$$

where $p(w_i|\boldsymbol{w}_{<i}, \mathbf{c})$ can be decomposed into two parts that deal with the LM for $p(w_i|\boldsymbol{w}_{<i})$ and the discriminator for $p(\mathbf{c}|\boldsymbol{w}_{1:i})$. In Section 4, we introduce these two parts in detail. We assume that any two constraints are independent: $p(\mathbf{c}|\boldsymbol{w}_{1:i}) = \prod_{j=1}^{k} p(c_j|\boldsymbol{w}_{1:i})$.

4 Methodology

The overall framework of A-TIP is shown in Fig.2. A-TIP contains two components: a *text infilling model* and a *plug-and-play attribute-aware controller*.

4.1 Text Infilling Model

Given a corpus consisting of complete text examples, we first create *infilling examples* and then train the GPT-2 with these examples. Specifically, given an input example $\boldsymbol{w}_{1:n}$ with n tokens, we first randomly replace m non-overlapping word spans $\mathbf{S} = \{\boldsymbol{s}_1, \ldots, \boldsymbol{s}_m\}$ in $\boldsymbol{w}$ with [Mask] tokens to form a corrupted text $\widetilde{\boldsymbol{w}}$. We also assume each span $\boldsymbol{s}_i$ contains n_i consecutive tokens $[s_{(i,1)}, \ldots, s_{(i,n_i)}]$. Then, we concatenate the spans S separated by [Answer] tokens to form a training target $\widetilde{\mathbf{S}}$ = {[Answer], $s_{(1,1)}, \ldots, s_{(1,n_1)}$, [Answer], ..., [Answer], $s_{(m,1)}, \ldots, s_{(m,n_m)}$ }. Finally, we construct a complete infilling example by concatenating $\widetilde{\boldsymbol{w}}$ and $\widetilde{\mathbf{S}}$ (see Token Embedding in Fig.2).

There are two advantages of designing such an input format. First, we add only $2m$ additional tokens (one [Mask] and one [Answer] per blank as shown in Fig.2 "Token Embedding" add 4 tokens for two spans). Although memory usage for GPT-2 grows quadratically with sequence length, as m is small, additional training time complexity will be minimal. Second, we can apply two different attention strategies for the corrupted text $\widetilde{\boldsymbol{w}}$ and training target text $\widetilde{\boldsymbol{S}}$. As shown in Fig.2 (A), while tokens in the corrupted text have attentions on all other tokens in $\widetilde{\boldsymbol{w}}$, tokens in the training target can have attentions only on its previous tokens. By

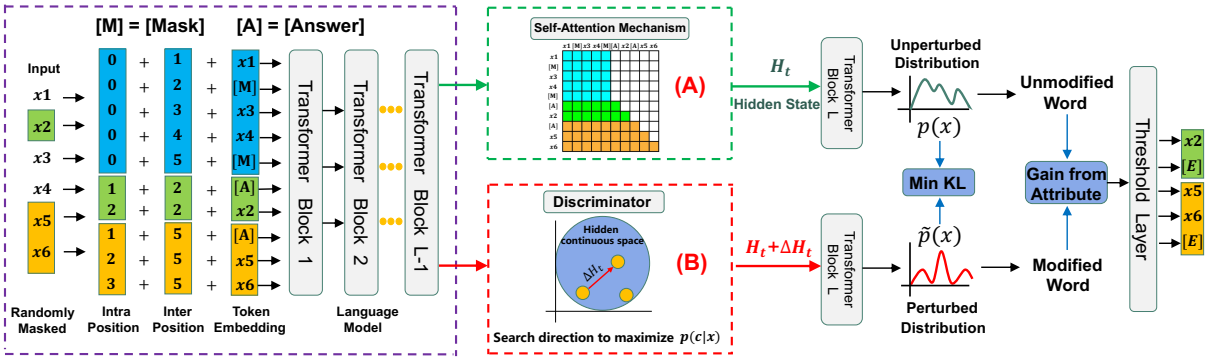

Figure 2: Model overview. We first fine-tune an off-the-shelf GPT-2 by adopting a new attention mechanism and two-level positional encoding to infill blanks. Then, we design a plug-and-play discriminator to guide generation in the direction of improving attribute relevance. We also adopt KL divergence and a threshold-based strategy to provide fine-grained control over fluency and attribute relevance.

adopting such an attention mechanism, when A-TIP infills the i-th blank $\boldsymbol{s}_i$, it will focus on the bidirectional context of the i-th blank, which can ensure the well-formedness and rationality of the whole sentence.

Current methods hardly perceive the number/location and infilling length for each blank. We design two-level positional encoding, which can provide fine-grained control over them. Specifically, each token is encoded with two position IDs. The first position ID represents the inter-position in the corrupted text $\widetilde{\boldsymbol{w}}$ and the second position ID represents the intra-position in each span.

Finally, A-TIP trains the GPT-2 with the *infilling examples* in an auto-regressive manner. When predicting missing tokens in each blank, A-TIP has access to the corrupted text $\widetilde{\boldsymbol{w}}$ and the previously predicted blanks. Formally, the probability of generating the i-th blank $\boldsymbol{s}_i$ is

$$p_\theta(\boldsymbol{s}_i|\widetilde{\boldsymbol{w}}, \boldsymbol{s}_{<i}) = \prod_{j=1}^{n_i} p(s_{i,j}|\widetilde{\boldsymbol{w}}, \boldsymbol{s}_{<i}, \boldsymbol{s}_{i,<j}), \quad (4)$$

where θ are parameters for the GPT-2, n_i represents the number of tokens in $\boldsymbol{s}_i$, $s_{i,j}$ represents the j-th token in the span $\boldsymbol{s}_i$, $\boldsymbol{s}_{<i}$ represents previously predicted blanks, and $\boldsymbol{s}_{i,<j} = \{s_{i,1}, \cdots, s_{i,j-1}\}$.

4.2 Plug-and-play Attribute-aware Controller

To clarify our approach, we follow the notation of Dathathri et al. (2020) and define the GPT-2 decoding process (Eq.(4)) in a recursive manner. Specifically, we first define $\mathbf{H}_t$, that contains all historical key-value pairs, i.e., $\mathbf{H}_t = [(\mathbf{K}_t^{(1)}, \mathbf{V}_t^{(1)}), \ldots, (\mathbf{K}_t^{(l)}, \mathbf{V}_t^{(l)})]$, where $(\mathbf{K}_t^{(l)}, \mathbf{V}_t^{(l)})$ stores all key-value pairs of t tokens in the l-th layer. Then, we formally define the recurrent decoding process to generate the i-th token as:

$$\boldsymbol{o}_i, \mathbf{H}_i = \text{GPT-2}(\boldsymbol{w}_{<i}, \mathbf{H}_{i-1}), \quad (5)$$

where $\boldsymbol{o}_i$ is the hidden state of the input at i-th time-step. Then, we sample the i-th generated token from the following distribution by beam search (Hokamp and Liu, 2017):

$$w_i \sim p_i = \text{Softmax}(\mathbf{W}\boldsymbol{o}_i), \quad (6)$$

where $\mathbf{W}$ is a parameter matrix that maps the hidden state $\boldsymbol{o}_i$ to a vector of the vocabulary size.

In accordance with Bayes' theorem in Eq.(3), we have $p(w_i|\boldsymbol{w}_{<i}, \mathbf{c}) \propto p(w_i|\boldsymbol{w}_{<i}) \cdot p(\mathbf{c}|\boldsymbol{w}_{1:i})$. To achieve attribute-aware text infilling, when we infill the i-th blank, we shift history matrix $\mathbf{H}_{i-1}$ towards the direction of the sum of two gradients: 1) To maximize the log-likelihood of the attribute $\mathbf{c}$ under the conditional attribute model $p(\mathbf{c}|\boldsymbol{w}_{1:i})$ and 2) To ensure high fluency of text infilling $p(w_t|\boldsymbol{w}_{<i})$. We update only $\mathbf{H}_{i-1}$ and fix other model parameters unchanged since next-token prediction depends only on the past key-value pairs via $\mathbf{H}_{i-1}$. Thus, we propose to gradually update $\mathbf{H}_{i-1}$ to guide future generation in the desired direction.

Let $\Delta\mathbf{H}_{i-1}$ be the update to $\mathbf{H}_{i-1}$ to shift the generation infilling content towards the desired attribute direction $\mathbf{c}$. At the beginning of the generation, $\Delta\mathbf{H}_{i-1}$ is initialized to zero, and we can obtain the unmodified distribution as p_i. Then, we update $\Delta\mathbf{H}_{i-1}$ with gradients from the attribute model that measures the extent to which the generated text possesses the desired attribute. Following Dathathri et al. (2020), we rewrite $p(\mathbf{c}|\boldsymbol{w}_{1:i})$ as Pb = $p(\mathbf{c}|\mathbf{H}_{i-1} + \Delta\mathbf{H}_{i-1})$ and define the gradient up-

date for $\Delta \mathbf{H}_{i-1}$ as

$$\Delta \mathbf{H}_{i-1} \leftarrow \Delta \mathbf{H}_{i-1} + \alpha \frac{\nabla_{\Delta \mathbf{H}_{i-1}} Pb}{\|\nabla_{\Delta \mathbf{H}_{i-1}} Pb\|^{\gamma}}, \quad (7)$$

where α is the learning rate and γ is the scaling coefficient for the normalization term to control the relevance of the attribute. We repeat Eq.(7) less than 10 times to generate attribute-aware tokens. Subsequently, the new $\widetilde{\mathbf{H}}_{i-1} = \mathbf{H}_{i-1} + \Delta \mathbf{H}_{i-1}$ is computed, and a new token is generated using $\widetilde{\boldsymbol{o}}_i, \mathbf{H}_i = \text{GPT-2}(\boldsymbol{w}_{<i}, \widetilde{\mathbf{H}}_{i-1})$. The described optimization process is repeated for every token in the generated sequence. Compared with the unconditional LM-based text generation task, this process will not take much time (see detail in experiments).

Although we can generate attribute-aware infilling content, we can easily generate low-quality, repetitive, and low-fluency text. Thus, we add two additional components to ensure the fluency and quality of generated infilling content with its bidirectional context. First, we minimize the KL divergence between the unmodified distribution p_i and modified distribution $\widetilde{p}_i$ for the i-th token:

$$\min \ \ \mathrm{D}_{KL}(\widetilde{p}_i \,||\, p_i). \quad (8)$$

Our objective function can be reformulated as

$$Loss = p(\mathbf{c}|\widetilde{\mathbf{H}}_{i-1}) + \lambda \mathrm{D}_{KL}(\widetilde{p}_i \,||\, p_i), \quad (9)$$

where λ is a parameter to balance the fluency and attribute relevance. Then, we update $\Delta \mathbf{H}_{i-1}$ as:

$$\Delta \mathbf{H}_{i-1} \leftarrow \Delta \mathbf{H}_{i-1} + \alpha \frac{\partial_{\Delta \mathbf{H}_{i-1}} Loss}{\|\partial_{\Delta \mathbf{H}_{i-1}} Loss\|^{\gamma}}. \quad (10)$$

Intuitively, we can generally find many words that have different levels of correlations with the specific attribute (Mohammad, 2018). For example, {perfect, good, bad, like} can mainly determine the sentiment of a sentence. Thus, we define *Gain* from the attribute to determine whether to change a generated word. As shown in Fig.2, two candidate words are sampled from the unmodified distribution (before back propagation) and modified distribution (after back propagation), respectively. *Gain* between two candidate words in the conditional model can be formulated as

$$Gain = p(\mathbf{c}|\boldsymbol{w}_{<i}, \widetilde{w}_i) - p(\mathbf{c}|\boldsymbol{w}_{<i}, w_i), \quad (11)$$

where $\widetilde{w}_i$ and w_i are samples from the modified and unmodified distributions, respectively.

To better control the relevance of the attribute, we define a threshold δ to determine whether to generate a word from the modified distribution. Specifically, *Gain* >δ represents that the word generated from the modified distribution can have a relatively remarkable effect on attributes. Otherwise, if the discriminator does not guide well at certain steps (*Gain* <δ), we select the word generated from the unmodified distribution to maintain the fluency to be the same as the original unconditional text infilling model to the greatest extent.

Discriminator Construction: As shown in Fig.2 (B), for simplicity, we train a linear classifier f as a discriminator with annotated datasets, indicating a sentence and label pair as $(\boldsymbol{w}, \boldsymbol{y})$. Specifically, for each sentence $\boldsymbol{w}$ of length t, we compute the set of hidden states $\boldsymbol{o} = \{o_1, \ldots, o_t\}$ from the GPT-2. Then, we compute the mean of $\boldsymbol{o}$ as $\bar{\boldsymbol{o}}$ and train f using the cross-entropy between the true label distribution $\boldsymbol{y}$ and predicted label distribution $f(\bar{\boldsymbol{o}})$. The number of parameters in this layer is (embedding dimension $\times$ number of attributes + number of attributes), which is negligible compared with the number of parameters in the text infilling model itself.

5 Experimentation

As shown in Table 1, we evaluated the proposed methods on three tasks to demonstrate that our framework is not custom tailored to a single domain: sentiment-aware, domain knowledge-aware, and topic-aware text infilling. We also show a case study for these tasks. We determined whether A-TIP can generate infilling text that satisfies the desired attribute and whether it can infill high-quality text in blanks by using both automated methods and human annotators.

Dataset	Examples	Words	Attributes
SST-5	11,855	215,154	5
Abstracts	200K	30M	8
ROCStories	100K	5M	13

Table 1: Descriptive statistics of three datasets.

5.1 Experimental Settings

Datasets In addition to using the datasets in Table 1 to train our text infilling model, we also adopted sentiment labels in SST-5 (Pang and Lee, 2005) for sentiment-aware text infilling, research

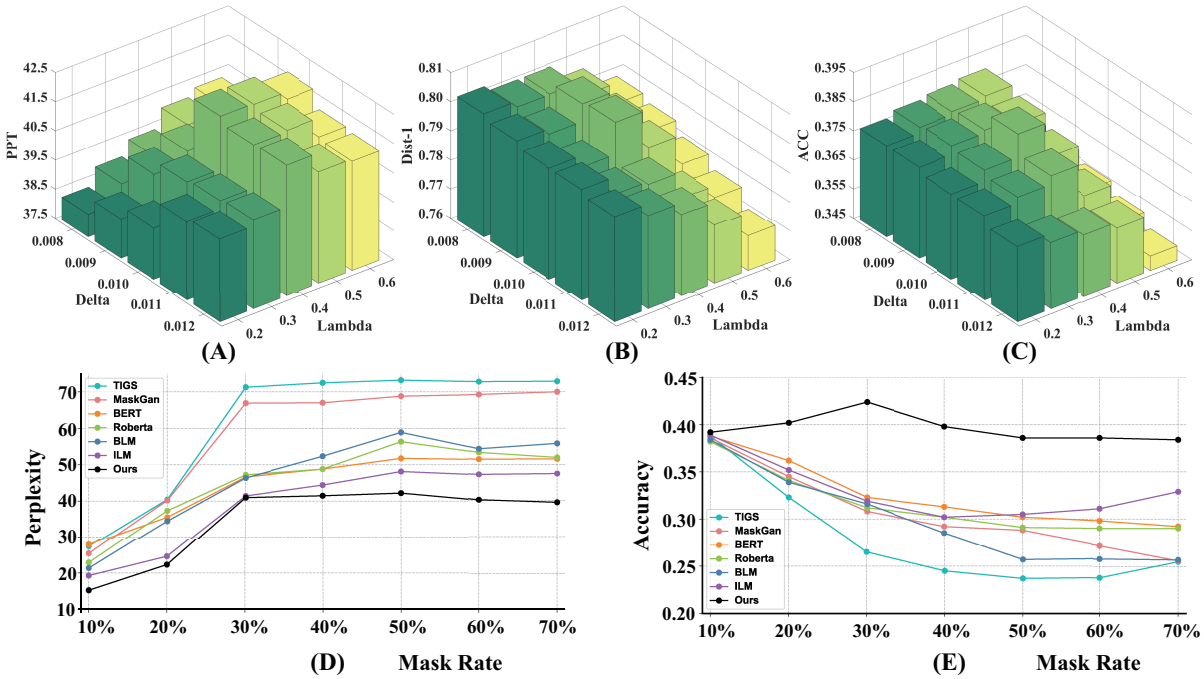

Figure 3: Based on the validation data of SST-5, we evaluated the parameter effect for Perplexity (A), Dist1 (B), and Accuracy (C). We draw the effect of mask rate on performance of text infilling for Perplexity (D) and Accuracy (E).

area labels in Abstracts (Donahue et al., 2020) for domain knowledge-aware text infilling, and topic labels in ROCStories (Mostafazadeh et al., 2016) for topic-aware text infilling. For the datasets with attribute labels like SST-5 and Abstracts, we can directly use their labels to train our plug-and-play discriminator. However, considering that most datasets do not have attribute labels, we adopted COMBINETM (Bianchi et al., 2021) to detect attributes for them (details in Appendix A). For example, for ROCStories, we can detect thirteen attributes and prove that A-TIP can generate a relevant topic in human evaluation (Table 3).

We split the datasets into $80\%/10\%/10\%$ as training/validation/test data, respectively. Following TIGS (Liu et al., 2019) and BLM (Shen et al., 2020), we randomly masked $r\%$ tokens in each document. To ensure that all experiments are performed on the same data, we removed infilling examples that exceed our training sequence length of 256 tokens.

Evaluation Metrics In automated evaluation, perplexity is a measure for fluency in open-domain text generation.[2] We measured it using GPT-2. The diversity of text was measured using the number of distinct n-grams (normalized by text length) as in Li et al. (2016). We reported Dist1, Dist2, and Dist3 scores for the distinct 1, 2, 3-grams. Following Dathathri et al. (2020), we used an external classifier to evaluate *Accuracy* (macro-average F-score) for sentence attribute labels. We evaluated the attribute control for sentiment (SST-5) with an external sentiment classifier with XLNet (Yang et al., 2019), which was trained with the IMDB dataset. We chose a BERT-based classifier (Lopes et al., 2021) for the Abstracts dataset. The t-test was used to evaluate the significant performance difference between two approaches (Yang and Liu, 1999) for both automated and human evaluations.

Baselines We compared A-TIP with six baselines that can be classified in four classes (Section 2.2): 1) **Inference-based:** We trained TIGS (Liu et al., 2019), an RNN-based seq2seq model. At inference time, we iteratively searched tokens in continuous space and projected their vectors to real words. 2) GAN-based: We trained the generator of MaskGan (Fedus et al., 2018) on PLM with a seq2seq architecture. The discriminator can make word distributions of the generator closer to those of the real word distribution. 3) **Masked LM-based:** We used representations of blanks as seeds to fine-tune BERT (Devlin et al., 2019) and Roberta (Liu et al., 2020). At inference time,

[2]Overlap-based metrics such as BLEU scores (Papineni et al., 2002) are not appropriate for evaluating infilling as there are many realistic infills that have no word-level overlap with the original.

Datasets	SST-5					Abstracts					ROCStories			
Metrics	PPL	Dist1	Dist2	Dist3	ACC	PPL	Dist1	Dist2	Dist3	ACC	PPL	Dist1	Dist2	Dist3
TIGS	73.23	0.475	0.424	0.425	0.237	49.70	0.659	0.657	0.644	0.453	63.30	0.672	0.675	0.691
MaskGan	68.83	0.385	0.758	0.728	0.288	48.82	0.652	0.662	0.642	0.494	63.32	0.677	0.671	0.701
BERT	51.76	0.773	0.732	0.732	0.302	28.86	0.683	0.656	0.624	0.508	64.16	0.673	0.636	0.560
Roberta	56.34	0.392	0.745	0.745	0.291	26.22	0.710	0.710	0.700	0.528	42.96	0.666	0.659	0.540
BLM	58.90	0.548	0.329	0.345	0.257	50.34	0.512	0.431	0.356	0.568	45.69	0.591	0.594	0.614
ILM	48.14	0.805	0.792	0.801	0.305	21.30	0.710	0.710	0.706	0.634	37.53	0.678	0.692	0.709
A-TIP/Dis	**40.26**	0.789	0.765	0.742	0.301	**18.82**	0.708	0.708	0.698	0.614	**30.35**	0.662	0.653	0.688
A-TIP/KL	51.22	0.797	0.788	0.782	**0.421**	28.97	0.711	0.711	0.706	**0.752**	47.35	0.685	0.693	0.718
A-TIP	42.21†	**0.805**†	**0.807**†	**0.808**†	0.386†	20.36†	**0.711**†	**0.711**†	**0.707**†	0.694†	32.13†	**0.685**†	**0.693**†	**0.721**†

Table 2: Overall performance comparison. PPL is perplexity, Dist scores measure divergence, and ACC is classification accuracy. † shows our results significantly surpass all baselines using t-test with p <0.005. Underlines mean our ablation algorithm can achieve better results than A-TIP for a metric.

blanks are infilled one after another and are conditioned on the previous generation. We trained BLM (Shen et al., 2020) with a seq2seq architecture, where the encoder module is a transformer (base) and the decoder process adopts beam search. 4) **LM-based:** We trained ILM (Donahue et al., 2020) by fine-tuning GPT-2 to output a full document from a masked input. Note that it may have invalid outputs that do not match the input format.

Implementation Details In our experiments, we set the learning rate $\alpha = 1e-4$ and the scaling coefficient $\gamma = 0.5$ for Eq. (10). Sequence representations were obtained by the GPT-2 module (12 layers, 12 heads, $n_{embd} = 768$, $n_{ctx} = 1024$, batch size $= 24$). We applied the Adam (Kingma and Ba, 2015) optimizer with an initial learning rate of 1e-4, and the weight decay and dropout were turned based on the loss on the validation data. Our discriminator has a linear layer on the head of GPT-2. For a fair comparison, we followed the default parameter settings of the baselines and repeated all experiments 10 times to report the average accuracy. The unpaired t-test was used to evaluate the significant difference between any two approaches as multiple comparisons (details in Appendix B) for both automated and human evaluations. We trained models with early stopping. Following Dathathri et al. (2020), we evaluated the attribute control for sentiment with an external sentiment classifier.

Parameter Sensitivity A-TIP uses two hyperparameters. λ dominates the attribute relevance of generated text and δ can control the fluency of infilling content. We analyzed the parameter sensitivity on all three validation data and selected the validation data of SST-5 as an example to determine the parameter sensitivity of A-TIP. As shown in Figs.3 (A-C), we observed how λ and δ affect the performance of A-TIP by varying λ from 0.2 to 0.6 in 0.1 intervals and δ from 0.008 to 0.012 in 0.001 intervals. The results indicated that A-TIP obtain the best performance when $\lambda \in [0.4, 0.5]$ and $\delta \in [0.010, 0.011]$. The reason why these parameters can affect the results is that when $\lambda < 0.4$, the attribute relevance becomes stronger and the fluency gets destroyed. $\lambda > 0.5$ weakens both the attribute relevance and text diversity. When $\delta < 0.01$, A-TIP tends to preserve modified words, which leads to low fluency. When $\delta > 0.012$, A-TIP preserves the original unmodified words, which causes low attribute relevance and diversity of text. To achieve a balanced performance, we set λ=0.4 and δ=0.01 on all datasets in our experiments.

Considering that the mask rate r is also a hyperparameter, we analyzed its effect on the results by varying it from 10% to 70%. We found the same trend on all datasets and took SST-5 as an example. As shown in Fig.3 (D), the fluency decreased when r varies from 10% to 40% because infilling content may be well-formed with its bidirectional context. As r increased from 40% to 70%, the fluency of text mainly depends on the baselines' original generation ability, which is stable. Fig.3 (E) shows that when r increases, the baselines cannot recover the attributes of infilling content well. However, A-TIP can generate attribute-aware text to improve the classification accuracy. All baselines can obtain stable fluency and classification accuracy when r = 50%, we fixed r= 50% to show numerical experimental results in the later experiments.

Dataset	SST-5		Abstracts		ROCStories	
Metrics	Fluency	Attri-Rele	Fluency	Attri-Rele	Fluency	Attri-Rele
TIGS	4.076	4.008	4.072	3.920	4.080	3.960
MaskGan	3.982	3.892	3.962	3.921	4.002	3.861
BERT	4.320	4.196	4.180	4.120	4.076	3.988
Roberta	4.168	4.132	4.068	3.892	4.016	4.032
BLM	4.084	3.956	3.856	3.968	4.072	3.992
ILM	4.236	4.076	4.104	3.964	4.048	3.992
A-TIP	**4.476**†	**4.320**†	**4.396**†	**4.296**†	**4.452**†	**4.348**†

Table 3: Human evaluation on three datasets. † indicates the results significantly surpass others.

5.2 Automated Evaluation

We evaluated the performance of A-TIP on attribute-aware text infilling by measuring PPL, Dist1, Dist2, Dist3, and ACC on the test data. Table 2 shows, A-TIP outperformed other baselines, indicating that our proposed framework can take advantage of the bidirectional context and attribute information. Additionally, ILM can achieve good results on PPL because it also adopts GPT-2 for text infilling. However, compared to one-layer positional encoding and auto-regression attention mechanism in ILM, A-Tip/Dis (A-Tip without discriminator) achieves better fluency (PPL) because it adopts the modifies attention mechanism (Fig.2 (A)) to effectively learn the length for each blank, and focus on the number/location of blanks by two-level positional encoding (intra- and inter-blank).

A-TIP obtained more accurate sentence attributes than other baselines, which demonstrates A-TIP can generate text that satisfies the desired attribute. While the accuracy was improved by 8% compared with the baselines, we observed ILM and BERT also yield high classification accuracy. This is because we randomly masked 50% of tokens in the original input without considering whether the token has a specific attribute. We did not generally mask attribute relevant tokens, that helps the sentence maintain its original attribute. If all attribute relevant tokens are masked, we can obtain better results. For a fair comparison, we randomly masked tokens instead of masking specific tokens.

5.3 Ablation Study

To verify the effect of each component in A-TIP, we conducted an ablation study. In specific, A-TIP/Dis does not include the plug-and-play discriminator, and the text infilling part remains unchanged. A-TIP/KL does not include the KL loss and threshold-based strategy. Table 2 shows A-TIP/Dis can improve text fluency while reducing attribute relevance. A-TIP/KL increases attribute relevance and decreases text fluency. Since the discriminator can guide generation towards the attribute-aware direction, while losing the fluency to a certain extent. By incorporating KL and a threshold, A-TIP achieves a better balanced performance.

5.4 Human Evaluation

We considered two types of human annotation: fluency and attribute relevance (Attri-Rele). Annotators were asked to evaluate the fluency/attribute relevance of each individual sample on a scale of 1∼5, with 1 being Not fluent/Not relevant at all and 5 being Very fluent/Very relevant, as in (Lample et al., 2019). We randomly selected 100 samples for each baseline from each test data and asked ten people on Amazon Mechanical Turk to identify the fluency and attribute relevance for each sample. We then used the average scores of ten annotations as final scores (see more detail in Appendix C).

As shown in Table 3, A-TIP achieved the highest score compared with the baselines, indicating that sentences infilled by A-TIP can be not only more fluent but also more attribute relevant. Somewhat surprisingly, we observed that BERT, TIGS, and MaskGan yield the worst performance. BERT performed poorly due to the intrinsic difficulty of finding convincing infilling content with a suitable length. TIGS and MaskGan may have performed poorly because, unlike ILM and A-TIP, they were not initialized from a large-scale pre-trained LM.

5.5 Running Time Comparison

To generate attribute-aware tokens, we update the Eq.(10) less than 10 times for each token. As shown in Fig.5, we compare the running time be-

Sentiment content infilling: Watching these [Mask] is both [Mask] and [Mask] .
Roberta: Watching these **kids** is both **funny** and **heartbreaking too**.
BLK: Watching these **teams** is both **inspiring** and **the action**.
A-TIP: Watching these **performances** is both **inspiring** and **artfully mesmerizing**. *(Positive)*
A-TIP: Watching these **shows** is both **boring** and **disgusting me much**. *(Negative)*

Expert knowledge infilling: [Mask] of [Mask] and [Mask] of their [Mask] as [Mask] using the [Mask] .
Bert: **One** of **her friends** and **one** of their **friends** as **a child** using the **same name** .
TIGS: **Systems** of **and control** and **capability** of their **distance** as **a process** using the **combination problem**.
A-TIP: **Analysis** of **data sources** and **simulation** of their **behaviors** as **a program** using the **execution model**. *(CS)*
A-TIP: **Introduction** of **randomness matrices** and **decomposition** of their **method** as a **tool** using the **matrix theory**. *(Math)*

Figure 4: Case study for sentiment content infilling and expert knowledge infilling.

tween A-TIP/Dis and A-TIP to ensure that we have less additional time-consuming. Specifically, we randomly select 30 samples from SST-5 and ROCStories datasets, where SST-5 contains short sentences and ROCStories contains almost long sentences. Then, we changed the mask rate from 30% to 70% for each selected sample to make our results more reliable. As shown in Fig.5, compared with the unconditional LM-based text generation task, updating the hidden state towards attribute-relevant direction will take less additional time.

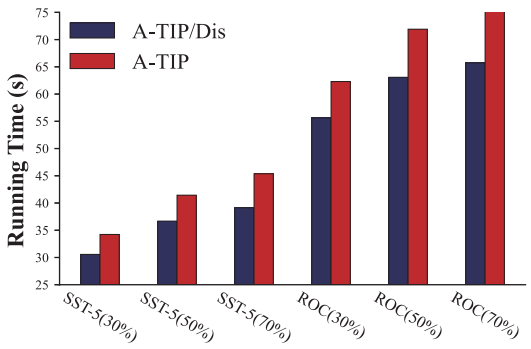

Figure 5: Running time comparison between A-TIP/Dis and A-TIP on SST-5 and ROCStories. We change the mask rate from 30% to 70%.

5.6 Case Study

We conducted a case study to show the infilling ability of A-TIP. Specifically, as shown in Fig.4, we first propose to infill the blanks with sentimental words. We choose Roberta and BLK as our compared examples. Because these two methods get the best result in this case. We can see Roberta infill the blanks with two contradictory words (funny and heartbreaking), where humans do not have such contradictory and complex emotional expressions. BLK can unify the expression of emotion, but it can not ensure the fluency of the generated sentence. In contrast, we can control A-TIP to generate positive or negative infilling contents with high fluency.

We want to explore if A-TIP can generate domain knowledge for a specific area for the second case. We choose BERT and TIGS as our compared examples. Since these two methods get the best result in domain knowledge infilling. We find that they cannot generate expert knowledge infilling content. And they tend to generate correct and high-frequency infilling content, while they are generally meaningless and monotonous (Sun et al., 2021; Lazar et al., 2021; Su et al., 2021). However, we can control A-TIP to generate both CS-related and Math-related infilling content by constraining the attribute as CS and Math.

6 Conclusion

In this paper, we presented a simple strategy for text infilling A-TIP that leverages an LM by proposing new attention mechanisms and two-level positional encoding to effectively improve the quality of generation in limited data settings. Furthermore, our plug-and-play discriminator can guide the generation towards the direction of improving text attribute relevance. In future work, we plan to incorporate the plug-and-play discriminator into more systems that assist humans in the writing process, where we hope that our work encourages more investigation of text infilling.

Acknowledgements

We would like to gratefully thank the anonymous reviewers for their helpful comments and feedback. Dongyuan Li and Jingyi You acknowledge the support from China Scholarship Council (CSC).

References

Yoav Benjamini and Yosef Hochberg. 1995. Controlling the false discovery rate: a practical and powerful approach to multiple testing. *Journal of the Royal statistical society: series B (Methodological)*, 57(1):289–300.

Federico Bianchi, Silvia Terragni, and Dirk Hovy. 2021. Pre-training is a hot topic: Contextualized document embeddings improve topic coherence. In *Proceedings of NAACL*.

Thodsaporn Chay-intr, Hidetaka Kamigaito, and Manabu Okumura. 2021. Character-based Thai word segmentation with multiple attentions. In *Proceedings of RANLP*, pages 264–273.

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 *Proceedings of ICLR*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: pre-training of deep bidirectional transformers for language understanding. In *Proceedings of NAACL*.

Chris Donahue, Mina Lee, and Percy Liang. 2020. Enabling language models to fill in the blanks. In *Proceedings of ACL*.

William Fedus, Ian J. Goodfellow, and Andrew M. Dai. 2018. Maskgan: Better text generation via filling in the _____. In *Proceedings of ICLR*.

Jiaxian Guo, Sidi Lu, Han Cai, Weinan Zhang, Yong Yu, and Jun Wang. 2018. Long text generation via adversarial training with leaked information. In *Proceedings of AAAI*.

Xingwei He. 2021. Parallel refinements for lexically constrained text generation with bart. In *Proceedings of EMNLP*.

Chris Hokamp and Qun Liu. 2017. Lexically constrained decoding for sequence generation using grid beam search. In *Proceedings of ACL*.

Nitish Shirish Keskar, Bryan McCann, Lav R. Varshney, Caiming Xiong, and Richard Socher. 2019. CTRL: A conditional transformer language model for controllable generation. In *Proceedings of ICLR*.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *Proceedings of ICLR*.

Ben Krause, Akhilesh Deepak Gotmare, Bryan McCann, Nitish Shirish Keskar, Shafiq R. Joty, Richard Socher, and Nazneen Fatema Rajani. 2021. Gedi: Generative discriminator guided sequence generation. In *Proceedings of EMNLP*.

Guillaume Lample, Sandeep Subramanian, Eric Michael Smith, Ludovic Denoyer, Marc'Aurelio Ranzato, and Y-Lan Boureau. 2019. Multiple-attribute text rewriting. In *Proceedings of ICLR*.

Koren Lazar, Benny Saret, Asaf Yehudai, Wayne Horowitz, Nathan Wasserman, and Gabriel Stanovsky. 2021. Filling the gaps in Ancient Akkadian texts: A masked language modelling approach. In *Proceedings of EMNLP*.

Jiwei Li, Michel Galley, Chris Brockett, Jianfeng Gao, and Bill Dolan. 2016. A diversity-promoting objective function for neural conversation models. In *Proceedings of NAACL*.

Zhiyu Lin and Mark Riedl. 2021. Plug-and-blend: A framework for controllable story generation with blended control codes. *arXiv preprint arXiv:2104.04039*.

Dayiheng Liu, Jie Fu, Pengfei Liu, and Jiancheng Lv. 2019. TIGS: An inference algorithm for text infilling with gradient search. In *Proceedings of ACL*.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2020. Roberta: A robustly optimized bert pretraining approach. In *Proceedings of ICLR*.

Lucas Gouveia Omena Lopes, Thales M. A. Vieira, and William Wagner M. Lira. 2021. Automatic evaluation of scientific abstracts through natural language processing. *arXiv preprint arXiv:2112.01842*.

Andrea Madotto, Etsuko Ishii, Zhaojiang Lin, Sumanth Dathathri, and Pascale Fung. 2020. Plug-and-play conversational models. In *Proceedings of EMNLP*.

Saif Mohammad. 2018. Obtaining reliable human ratings of valence, arousal, and dominance for 20,000 English words. In *Proceedings of ACL*.

Nasrin Mostafazadeh, Nathanael Chambers, Xiaodong He, Devi Parikh, Dhruv Batra, Lucy Vanderwende, Pushmeet Kohli, and James Allen. 2016. A corpus and cloze evaluation for deeper understanding of commonsense stories. In *Proceedings of NAACL*.

Jiefu Ou, Nathaniel Weir, Anton Belyy, Felix Yu, and Benjamin Van Durme. 2021. Infillmore: Frame-guided language generation with bidirectional context. In *Proceedings of *SEM*.

Bo Pang and Lillian Lee. 2005. Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales. In *Proceedings of ACL*.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of ACL*.

Damian Pascual, Beni Egressy, Clara Meister, Ryan Cotterell, and Roger Wattenhofer. 2021. A plug-and-play method for controlled text generation. In *Proceedings of EMNLP*.

Tuomas Puoliväli, Satu Palva, and J. Matias Palva. 2020. Influence of multiple hypothesis testing on reproducibility in neuroimaging research: A simulation study and python-based software. *Journal of Neuroscience Methods*, 337:108654.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. 2019. Language models are unsupervised multitask learners. *OpenAI blog*.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*.

Tianxiao Shen, Victor Quach, Regina Barzilay, and Tommi S. Jaakkola. 2020. Blank language models. In *Proceedings of EMNLP*.

Kaitao Song, Xu Tan, Tao Qin, Jianfeng Lu, and Tie-Yan Liu. 2019. MASS: masked sequence to sequence pre-training for language generation. In *Proceedings of ICML*.

Yixuan Su, David Vandyke, Simon Baker, Yan Wang, and Nigel Collier. 2021. Keep the primary, rewrite the secondary: A two-stage approach for paraphrase generation. In *Proceedings of ACL-IJCNLP*.

Bin Sun, Shaoxiong Feng, Yiwei Li, Jiamou Liu, and Kan Li. 2021. Generating relevant and coherent dialogue responses using self-separated conditional variational AutoEncoders. In *Proceedings of ACL-IJCNLP*.

Wilson L Taylor. 1953. "cloze procedure": A new tool for measuring readability. *Journalism quarterly*.

Yiming Yang and Xin Liu. 1999. A re-examination of text categorization methods. In *Proceedings of SIGIR*.

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime G. Carbonell, Ruslan Salakhutdinov, and Quoc V. Le. 2019. Xlnet: Generalized autoregressive pretraining for language understanding. In *Proceedings of NeurIPS*.

Jingyi You, Dongyuan Li, Hidetaka Kamigaito, Kotaro Funakoshi, and Manabu Okumura. 2022. Joint learning-based heterogeneous graph attention network for timeline summarization. In *Proceedings of NAACL*, pages 4091–4104.

Najam Zaidi, Trevor Cohn, and Gholamreza Haffari. 2020. Decoding as dynamic programming for recurrent autoregressive models. In *Proceedings of ICLR*.

Wanrong Zhu, Zhiting Hu, and Eric P. Xing. 2019. Text infilling. *arXiv preprint arXiv:1901.00158*.

A Detail Information for Datasets

As shown in Table 4, we give the number of examples, the total number of words and the detail attributes label for three widely used datasets, SST-5, ROCStories and Attributes, respectively. We selected these three datasets since we would like to check if A-TIP can infill the blanks with sentiment words, domain knowledge and topics.

We can directly use their labels to train our plug-and-play discriminator for datasets with attribute labels like SST-5 (sentiment labels) and Abstract (domain knowledge labels).

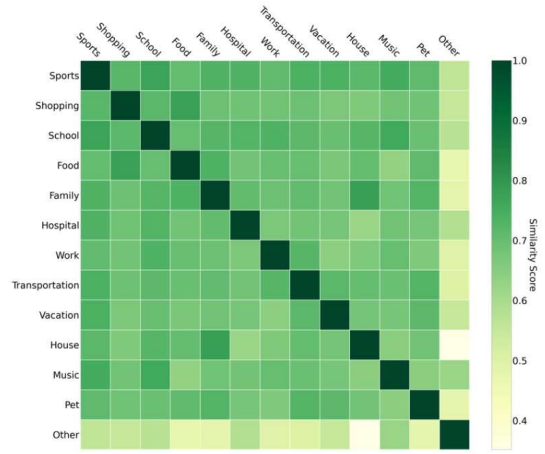

Figure 6: Topic similarity graph.

However, considering most datasets like ROCStories have no labels, we extend our method to deal with this situation. Intuitively, we can construct a general attribute-based plug-and-play discriminator to guide different datasets to generate different infilling content. However, in practical operation, it is unrealistic to build such an available attribute-based discriminator to guide the infilling generation because the downstream datasets have a variety of different attribute requirements. Therefore, we need to generate specific category labels for different downstream datasets to satisfy their specif attribute-related needs and use them to guide the infilling generation.

Specifically, we extend our model to more applications by combining our model with any topic exploration algorithms to mine topic labels on unlabeled datasets. For instance, we adopt COMBINETM (Bianchi et al., 2021) to detect topic attributes for ROCStories dataset by two methods *Contextual* and *Combined*. As shown in Table 5, we adopt three metrics to evaluate the quality of the attributes of ROCStories dataset: *Topic Coherence*, *Inverted RBO* and *NPMI*. And we choose 13 topics as our final labels since it has the best performance on average of all metrics.

As shown in Fig.6, we draw a topic similarity graph among thirteen topics. We find the similarity within topics is high, and the similarity between topics is low, demonstrating that the detected topics have high quality and low redundancy. We adopt 13 topic labels to train discriminators for ROCStories datasets, and we achieve the best performance about topic-relevant on human evaluation.

B Benjamini-Hochberg procedure

The Benjamini-Hochberg (*B-H*) Procedure is a powerful tool that decreases the false discovery rate (Benjamini and Hochberg, 1995). Considering the reproducibility of multiple significant test, we introduce how we adopt the *B-H* procedure and give the hyper-parameter values that we used.

Specifically, we first adopt t-test (Yang and Liu, 1999) with default parameters[3] to calculate p-value between each compared algorithm with A-TIP. Then, we put the individual p-values in ascending order as input to calculate p-value corrected by *B-H*. We directly use the "multipletests(*args)" function from python package[4] and set the hyper-parameter of false discover rate $Q = 0.05$ which is the widely used default value (Puoliväli et al., 2020). Finally, we get cut-off value as the output of "multipletests(*args)" function, where cut-off is a dividing line that distinguishes whether two groups of data are significant or not. Specifically, if the p-value is smaller than the cut-off value, we can get the conclusion that two groups of data are significant different.

C Detail Information for Human Evaluation

We show the human evaluation in Fig.7. We adopt fluency and attribute relevance as our evaluation metrics. We use their label as their attribute for labelled datasets SST-5 and Abstract. For unlabeled datasets like ROCStories, we manufacture labels as their attributes. And we list detailed scores from 1 to 5 for each metric.

[3] https://docs.scipy.org/doc/scipy/reference/generated/scipy.stats.ttest_ind.html

[4] https://www.statsmodels.org/dev/generated/statsmodels.stats.multitest.multipletests.html

Dataset	Examples	Words	Attributes
SST-5	11,855	215,154	Negative/ Somewhat negative/ Neutral/ Somewhat positive/ Positive
ROCStories	100K	5M	Sport/ Shop/ School/ Food/ Family/ Hospital/ Work/ Car/ Vacation/ House/ Music/ Pet/ Other
Abstracts	200K	30M	Condensed Matter/ CS/ Math/ Nonlinear Sciences/ Physics/ Bio/ Quant-Phy/ Statistics

Table 4: Descriptive statistics of datasets and their attributes.

	Contextual			Combined		
Topic Number	**Coherence**	**RBO**	**NPMI**	**Coherence**	**RBO**	**NPMI**
10	0.490	0.160	0.150	0.348	0.079	0.232
11	0.981	1.000	0.007	0.981	1.000	-0.008
12	0.986	1.000	0.030	0.933	1.000	-0.004
13	**0.993**	**1.000**	**0.053**	0.972	1.000	0.061
14	0.951	1.000	0.048	0.971	1.000	0.060
15	0.936	1.000	0.042	0.946	1.000	0.059
16	0.935	1.000	0.044	0.921	1.000	-0.008
17	0.905	0.998	0.042	0.922	0.992	0.037
18	0.906	0.982	0.045	0.868	0.989	0.038
19	0.892	0.977	0.043	0.822	0.982	0.021
20	0.882	0.972	0.040	0.802	0.978	0.022

Table 5: Contextual-based and Combined-based topic detection algorithms evaluate three widely used metrics: Topic Coherence, Inverted RBO, and NPMI.

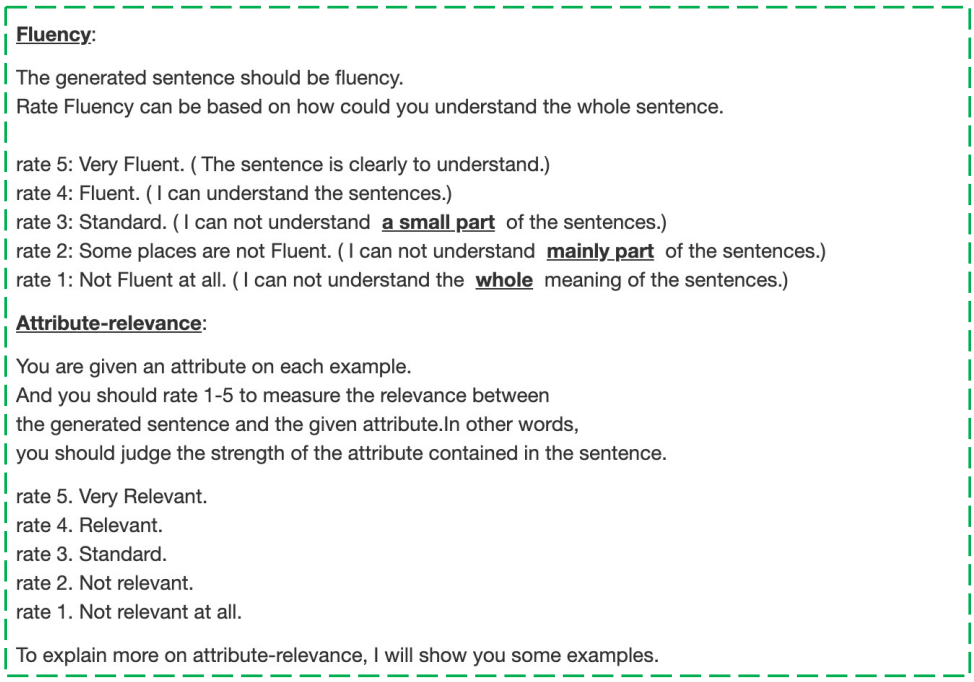

Figure 7: Human evaluations on Amazonmturk.