Blank Language Models

Tianxiao Shen[*] **Victor Quach**[*] **Regina Barzilay** **Tommi Jaakkola**
MIT CSAIL
{tianxiao, quach, regina, tommi}@csail.mit.edu

Abstract

We propose Blank Language Model (BLM), a model that generates sequences by dynamically creating and filling in blanks. The blanks control which part of the sequence to expand, making BLM ideal for a variety of text editing and rewriting tasks. The model can start from a single blank or partially completed text with blanks at specified locations. It iteratively determines which word to place in a blank and whether to insert new blanks, and stops generating when no blanks are left to fill. BLM can be efficiently trained using a lower bound of the marginal data likelihood. On the task of filling missing text snippets, BLM significantly outperforms all other baselines in terms of both accuracy and fluency. Experiments on style transfer and damaged ancient text restoration demonstrate the potential of this framework for a wide range of applications.[1]

1 Introduction

Neural language models have shown impressive performance across many applications such as machine translation and summarization where the text is generated from scratch (Bahdanau et al., 2014; Rush et al., 2015). However, a broader set of text generation tasks — including text editing, information fusion, and ancient text restoration — requires the model to start with partially specified text and generate the missing fragments. In the general setup, the input document may have any number of missing spans, and each span may have an unknown number of missing tokens. To perform this text infilling task (Zhu et al., 2019), a model should: (1) provide fine-grained control over the generation location, (2) accommodate a variable number of missing tokens, and (3) respect both the preceding and following context.

They also have ____ which ____ .
They also have *ice cream* which *is really good* .

Figure 1: BLM fills in blanks of arbitrary length.

Existing approaches focus on adapting left-to-right language models for text infilling. Intricate inference algorithms leveraging dynamic programming or gradient search are proposed to find the filling content that has a high likelihood within the surrounding context (Sun et al., 2017; Liu et al., 2019a; Zaidi et al., 2020). These methods make simplified Markov assumptions, require high decoding time complexity, and cannot adapt to variable infilling length. Alternatively, Donahue et al. (2020) predict the concatenation of the infilling content, but do not guarantee that the output will match the number of missing spans in the input.

In this work, we introduce the Blank Language Model (BLM), which uses a special "__" symbol to control where tokens can be placed. The generation of BLM follows the grammar of replacing a blank with a word and possibly adjoining blanks. By jointly modeling context and missing content, BLM supports the control of generation location and produces consistent infilling of variable length.

Our model can start from a single blank or partial text with blanks in specified locations. It maps the entire input into a sequence of vector representations, and further processes the representations in blank positions to determine the generation action. Generation actions are performed iteratively until there are no blanks. Since multiple trajectories of BLM actions can produce the same final text, we train the model by maximizing a lower bound of the log-likelihood marginalized over trajectories. At test time, we can use simple greedy decoding or beam search to fill in the blanks in the input text.

BLM shows superior performance in text infilling (Zhu et al., 2019), ancient text restoration (As-

[*]Equal contribution
[1]Our code is available at https://github.com/Varal7/blank_language_model

	Canvas c	Action a			
Step t		Location b	Word w	(Left l,	Right r)
0.	__#1__	#1	is	Y	Y
1.	__#1__ is __#2__	#1	customer	N	Y
2.	customer __#1__ is __#2__	#2	awesome	N	N
3.	customer __#1__ is awesome	#1	service	N	N
4.	customer service is awesome	-End-			

Figure 2: An example trajectory that generates the sentence "customer service is awesome". Each action is a tuple (b, w, l, r), indicating the blank location b selected for expansion, the word w to fill in, whether to create a left blank l, and whether to create a right blank r.

sael et al., 2019) and style transfer (Shen et al., 2017), demonstrating its flexibility to generate text in diverse conditions. Our model achieves 92.5% accuracy and BLEU score of 23.1 on the Amazon dataset for sentiment transfer. On the task of restoring ancient text that lost half of the characters, we reduce the error rate by 3.3 points compared to previous methods.

2 Related Work

Recent work has explored various sequence models for non-autoregressive machine translation (Gu et al., 2017). The Insertion Transformer supports dynamic canvas with word insertion (Stern et al., 2019), but does not allow users to specify where to insert. The model is unaware of which parts of the canvas are contiguous text spans that should remain intact, and which (potentially scattered) parts need to be filled in. Directly forcing the Insertion Transformer to perform text infilling can therefore lead to suboptimal solutions. The Levenshtein Transformer combines insertion and deletion through complex policy learning (Gu et al., 2019b). Its insertion mechanism is a two-stage process in which placeholders are first predicted and then filled-in in a masked language model (MLM) manner. In text infilling where the blanks/placeholders are given, it reduces to an MLM.

MLMs are commonly used in representation learning (Devlin et al., 2018; Joshi et al., 2020). To use them in rewriting tasks, one needs to specify the insertion length in advance and heuristically determine the generation order among the masks (Fedus et al., 2018; Wang and Cho, 2019; Ghazvininejad et al., 2019). Similarly, XL-Net requires absolute positional embedding and thus does not support unknown-length text infilling (Yang et al., 2019; Shih et al., 2019). BLM provides a natural formulation for generative modeling that can dynamically accommodate insertions of various length.

Another line of work focuses on finding an optimal language generation order, such as syntax-based generation (Dyer et al., 2016) and learning adaptive generation order (Gu et al., 2019a). These approaches are tailored to generation from scratch in a specific order. Our model instead is attuned for text rewriting, where the missing parts can be located anywhere in the input text, and the algorithm must flexibly complete them.

3 Blank Language Models

A blank language model (BLM) generates sequences by creating and filling in blanks. Generation starts with a single blank and ends when there is no blank. In each step, the model selects a blank "__", predicts a word w, and fills the blank with "w", "__ w", "w __", or "__ w __". This way, a blank can be expanded to any number of words.

We define a *canvas* as a sequence of words interspersed with special "__" tokens. The subsequent action is conditioned on this intermediate stage of generation. Suppose the current canvas is $c = (c_1, \cdots, c_n)$ with blanks located at indices $b_1, \cdots, b_k$ (i.e. $c_{b_i} =$ "__", for $i = 1, \ldots, k$). BLM maps this canvas to a distribution over actions specifying how the canvas is to be revised:

$$p(b, w, l, r | c; \theta) = \text{BLM}(c) \tag{1}$$

where $b \in \{b_1, \cdots, b_k\}$ is a blank location; w is a word in the vocabulary V; $l, r \in \{0, 1\}$ denote whether or not to create a blank to the left and right of w; and θ are the model parameters. The *action*, defined as the tuple (b, w, l, r) uniquely specifies the next state of canvas (see Fig. 2 for illustration).

Alternatively, we can view the actions in BLM as production rules in a grammar. Each blank represents a nonterminal symbol or the start symbol,

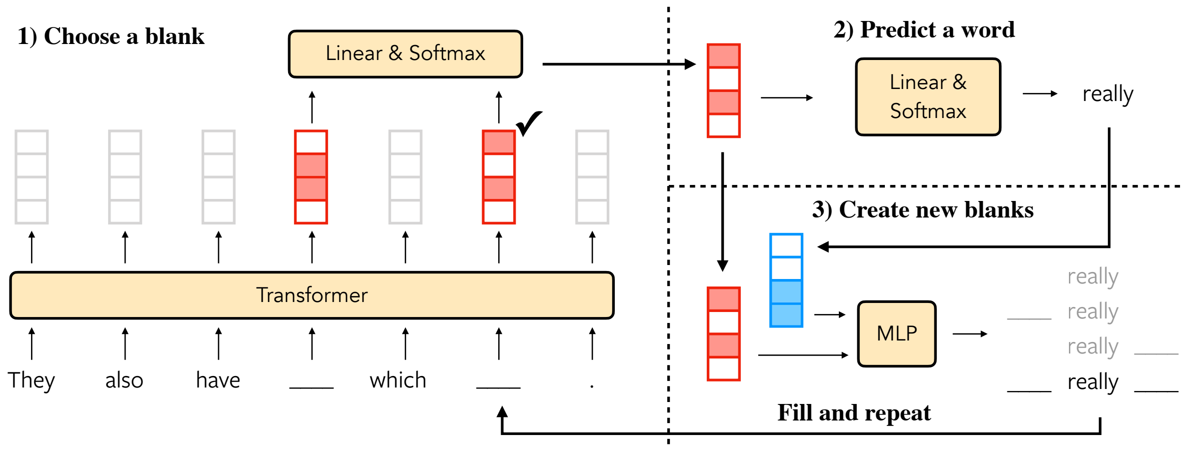

Figure 3: Architecture of the BLM. In the first stage, an index is chosen among all current blank positions. For that location, a word is selected in the second stage. In the final stage, the blank representation is concatenated with the chosen word's embedding and fed into an MLP to determine the creation of the following blanks.

and the terminal symbols come from the vocabulary V. The production rules are restricted to be of the form "__" → "__?w__?" for $w \in V$, where "?" indicates that the preceding symbol is optional. In contrast to context-free grammars, the probability distribution over production rules in BLM is conditioned on the entire canvas generated so far.

Model Architecture We encode the canvas c into a sequence of representations $(z_1, \cdots, z_n)$, and take representations $Z = (z_{b_1}, \cdots, z_{b_k})$ where the blanks are located. Let d denote the dimension of z's. We factorize the joint distribution $p(b, w, l, r|c; \theta)$ into three parts (shown in Fig. 3):

1. Choose a blank:

$$p(b_i|c; \theta) = \text{Softmax}(u^T Z) \tag{2}$$

where $u \in \mathbb{R}^d$ is a parameter vector to project z's into one-dimensional logits.

2. Predict a word for the selected blank:

$$p(w|c, b_i; \theta) = \text{Softmax}(W z_{b_i}) \tag{3}$$

where $W \in \mathbb{R}^{|V| \times d}$ is a parameter matrix to project z_{b_i} into the vocabulary.

3. Decide whether or not to create blanks to the left and right of the predicted word:

$$p(l, r|c, b_i, w; \theta) = \text{MLP}(z_{b_i}, v_w) \tag{4}$$

where v_w is the word vector of w, and MLP is a multilayer perceptron with 4 output classes: Left.Yes/No × Right.Yes/No.

Likelihood Now let us consider the probability $p(x; \theta)$ of generating a sentence/paragraph $x = (x_1, \cdots, x_n)$ under the BLM. We call the generating process from an initial blank to complete text a *trajectory*. The same final text x can be realized by multiple trajectories. However, if we specify the order in which the words in x are generated, the trajectory will be uniquely determined. Consider the example trajectory of a 4-word sentence in Fig. 2. Given the order $(3, 1, 4, 2)$, at step 0 when we generate x_3, both left and right blanks are created for future generations of x_1 and x_2, x_4. In step 1 of generating x_1, only a right blank is created for the future x_2. Subsequent steps can be deduced by analogy. The correspondence between trajectories and generation orders allows us to write the marginal likelihood as:

$$\begin{aligned} p(x; \theta) &= \sum_{\sigma \in S_n} p(x, \sigma; \theta) \\ &= \sum_{\sigma \in S_n} \prod_{t=0}^{n-1} p(a_t^{x,\sigma} | c_t^{x,\sigma}; \theta) \end{aligned} \tag{5}$$

where S_n is the set of all n-permutations; $a_t^{x,\sigma}, c_t^{x,\sigma}$ denote the action and canvas at step t under sentence x and order σ, respectively (cf. Fig. 2).

Training Different losses have been proposed to train generalized sequence models. For instance, BERT and XL-Net mask and predict 15% of tokens conditioned on the rest. This strategy is more suitable for representation learning rather than generation. Insertion Transformer masks different numbers of tokens and weights them with uniform loss or binary tree loss (Stern et al., 2019; Chan et al.,

Algorithm 1 BLM training[2]

```
1: Initialize model parameters $\theta$
2: repeat
3:     Sample a training example $x = (x_1, \cdots, x_n)$
4:     Sample $t$ from 0 to $n-1$
5:     Sample an $n$-permutation $\sigma$
6:     Construct canvas $c$ that keeps tokens $x_{\sigma_j} (j = 1, \cdots, t)$ and collapses remaining tokens as blanks
7:     Get $n-t$ target actions $a_{j-t}$ for filling $x_{\sigma_j}$ $(j = t+1, \cdots, n)$ into canvas $c$
8:     Compute $\text{loss}(\{a_1, \cdots, a_{n-t}\}, \text{model.forward}(c))$ from Eq. (8)
9:     Update $\theta$ by gradient descent
10: until Convergence
```

They also have ____ which ____ .
They also have ice cream which is really good .

τε εγγονον εισαι??????σοφιαι
τε εγγονον εισαιου του σοφιαι

The employees were **super nice** and **efficient** !
The employees were rude and unprofessional !

Figure 4: Examples of input and output for text infilling, ancient text restoration, and style transfer tasks.

2019). It aims to perform fast inference through parallel decoding. Here, we present a training objective from the language modeling perspective by estimating the log likelihood of generating x.

Directly computing the marginal likelihood over $n!$ orders is intractable. We apply Jensen's inequality to lower bound the log likelihood:

$$\log p(x;\theta) = \log \sum_{\sigma \in S_n} \prod_{t=0}^{n-1} p(a_t^{x,\sigma} | c_t^{x,\sigma}; \theta)$$

$$\geq \log(n!) + \frac{1}{n!} \sum_{\sigma \in S_n} \sum_{t=0}^{n-1} \log p(a_t^{x,\sigma} | c_t^{x,\sigma}; \theta) \quad (6)$$

where equality holds when the posterior $p(\sigma|x;\theta)$ is uniform. By maximizing this lower bound, we do not favor any particular order, but encourage the model to realize x equally well in all orders. It can help the model to complete any partial input text regardless of the position of blanks.

A naive training algorithm is to directly estimate the lower bound in Eq. (6): first uniformly sample a permutation σ from S_n and a step t from 0 to $n-1$, then construct the canvas $c_t^{x,\sigma}$, and compute the estimated loss $[-\log(n!) - n \cdot \log p(a_t^{x,\sigma} | c_t^{x,\sigma}; \theta)]$. However, this procedure has a large variance and can only compute the loss of a single action in one pass (in contrast to left-to-right language models that compute n word losses per pass).

To train the model more efficiently, we note that the canvas $c_t^{x,\sigma}$ depends only on the first t elements of σ. Hence we can combine into one pass the loss calculations of trajectories that are the same in the first t steps but different at the $t+1$ step. Switching the summation order of σ and t, we have:

$$\begin{aligned}
&\sum_{t=0}^{n-1} \frac{1}{n!} \sum_{\sigma \in S_n} \log p(a_t^{x,\sigma} | c_t^{x,\sigma}; \theta) \\
&= n \cdot \mathbb{E}_t \mathbb{E}_{\sigma_{1:t}} \mathbb{E}_{\sigma_{t+1}} \mathbb{E}_{\sigma_{t+2:n}} [\log p(a_t^{x,\sigma} | c_t^{x,\sigma}; \theta)] \\
&= n \cdot \mathbb{E}_t \mathbb{E}_{\sigma_{1:t}} \mathbb{E}_{\sigma_{t+1}} [\log p(a_t^{x,\sigma} | c_t^{x,\sigma}; \theta)] \\
&= \mathbb{E}_t \mathbb{E}_{\sigma_{1:t}} \left[\frac{n}{n-t} \sum_{\sigma_{t+1}} \log p(a_t^{x,\sigma} | c_t^{x,\sigma}; \theta) \right] \quad (7)
\end{aligned}$$

which leads to our efficient training algorithm: sample t from 0 to $n-1$ and partial permutation $\sigma_{1:t}$, construct the canvas $c_t^{x,\sigma}$, and compute loss:

$$-\log(n!) - \frac{n}{n-t} \sum_{\sigma_{t+1}} \log p(a_t^{x,\sigma} | c_t^{x,\sigma}; \theta) \quad (8)$$

The whole process is illustrated in Algorithm 1. In this way, we can compute in expectation $n/2$ action losses per pass.

4 Experiments

We test BLM's capacity to rewrite specified portions of text on three tasks: text infilling (Zhu et al., 2019), ancient text restoration (Assael et al., 2019) and style transfer (Shen et al., 2017). Fig. 4 displays example inputs and outputs for these tasks. We also measure the perplexity of BLM on language modeling benchmarks and compare with traditional left-to-right language models.

Experimental Details In all experiments, the sequence representations in BLM are obtained using the encoder module of `transformer_base` (Vaswani et al., 2017) (6 layers, 8 heads, $d_{model} = 512$, $d_{ff} = 2048$, $d_k = d_v = 64$). The MLP used for blank prediction has one hidden layer of size 1024. Weight decay, learning rate, and dropout are tuned based on the loss on the validation set for each dataset respectively. When decoding, we use beam size in $\{1, 5, 10\}$ and choose the best value as

[2]We implement a batch version of the algorithm.

	BLEU					PPL				
Mask ratio	10%	20%	30%	40%	50%	10%	20%	30%	40%	50%
No infill	75.2	55.0	37.4	23.6	13.0	98.4	163.0	266.3	421.0	647.9
InsT	84.8	72.3	58.9	46.0	33.8	**48.3**	**44.2**	41.8	39.7	37.7
MLM (oracle length)	83.7	69.3	55.5	43.2	32.2	58.4	59.8	59.8	59.0	56.8
BERT+LM	82.8	66.3	50.3	37.4	26.2	55.1	55.2	54.9	56.5	53.6
Seq2seq-full	86.3	72.9	59.4	46.3	34.0	51.3	46.9	41.0	**31.9**	**20.6**
Seq2seq-fill	82.8	67.5	52.9	39.9	28.6	64.6	71.0	73.4	65.6	48.7
BLM	**86.5**	**73.2**	**59.6**	**46.8**	**34.8**	50.2	44.9	**39.9**	35.0	32.7

Table 1: BLEU scores and perplexity of generated documents by different models for text infilling. The perplexity is measured by a pre-trained left-to-right language model, and the original documents have perplexity 55.8.

Mask ratio	10%	20%	30%	40%	50%
Seq2seq-full	15.0	22.4	28.7	33.3	40.6
Seq2seq-fill	31.0	28.4	34.5	42.5	47.2

Table 2: Infilling failure rate (%) of seq2seq models. Other methods always produce valid outputs.

observed on the validation set. We note that beam search in BLM does not search for the sentence with the maximum marginal likelihood $p(x; \theta)$, but instead for a sentence *and* a trajectory that have the maximum joint likelihood $p(x, \sigma; \theta)$.

4.1 Text Infilling

Dataset We experiment on the Yahoo Answers dataset, which has 100K/10K/10K documents for train/valid/test respectively (Yang et al., 2017). A document has a maximum length of 200 words, with an average of 78 words. Following Zhu et al. (2019), we automatically compile test data by deleting portions of documents. For each document x, we randomly mask a given ratio r of its tokens. Contiguous masked tokens are collapsed into a single "__", resulting in a canvas c to be completed.

Metrics We measure generation's accuracy by computing its BLEU score against the original document x, and fluency as its perplexity evaluated by a pre-trained (left-to-right) language model. We also report the failure rate, which is the percentage of invalid generations, such as missing existing words or not filling in all the blanks.

Baselines We compare BLM with five baselines:

- *Insertion Transformer (InsT)*: By default, InsT does not support controlling the insertion position. We force it to produce valid generations by normalizing the predictions over valid locations, disabling the $\langle$eos$\rangle$ prediction unless all blanks have been filled, and prioritizing slots that have not been filled yet. Without these steps, InsT would have a failure rate $\geq 88\%$.
- *MLM (oracle length)*: MLM for text infilling requires predicting the length of each blank. Here we replace blanks with the target number of $\langle$mask$\rangle$ tokens, and fill them autoregressively by the most-confident-first heuristic.
- *BERT+LM*: We use BERT's representation of each blank as seed for a left-to-right language model that learns to generate the tokens in the corresponding blank. At inference time, the multiple blanks are filled in one after another, conditioned on previous generations.
- *Seq2seq-full* (Donahue et al., 2020): We train a seq2seq model to output the full document x from input c. Note that it may have invalid outputs that do not match the input format, such as missing existing tokens in c or generating tokens in incorrect locations.
- *Seq2seq-fill* (Donahue et al., 2020): We train a seq2seq model to output only tokens to be placed in the blanks, with a special '|' token to indicate separation. For the example in Fig. 4, its target output will be "ice cream |is really good". Unlike *seq2seq-full*, *seq2seq-fill* does not have the problem of losing existing tokens in c. However, it may still fail to generate the correct number of '|' that matches the input.

Results As shown in Table 1, BLM achieves the highest BLEU score at all mask ratios: 0.7 to 1.7 higher than InsT, 2.6 to 4.1 higher than MLM with oracle length, and 3.7 to 9.4 higher than BERT+LM. InsT is not trained with insertion position control. Restricting it to generate at the specified positions thus bias the model towards making suboptimal

	Mask-ratio 10%	Mask-ratio 50%
Blanked	when time flies , ____ does it go ? ____ the center of the ____ to be recycled ____ made into new time .	when time ____ , where ____ ? ____ the ____ of ____ universe to ____ recycled ____ made into ____ .
BLM	when time flies , *where* does it go ? *for* the center of the *earth* to be recycled *and* made into new time .	when time *was created* , where *did it come from* ? *it was* the *first part* of *the* universe to *be* recycled *and* made into *space* .
InsT	when time flies , *where* does it go ? *for* the center of the *earth has* to be recycled *and* made into new time .	when time *was created* , where *was it* ? *what was* the *name* of *the* universe to *be* recycled *and* made into *space* .
MLM (oracle len)	when time flies , *where* does it go ? *from* the center of the *earth* to be recycled *converted* made into new time .	when time *is* , where *is the universe* ? *from* the *creation* of *the* universe to *be* recycled *and* made into *the universe* .
BERT+LM	when time flies , *where* does it go ? *to* the center of the *earth* to be recycled *came* made into new time .	when time *is* , where *to* ? *i need to find* the *way* of *the* universe to *be* recycled *and* made into *a lot* .
Seq2seq-full	when time flies , *where* does it go ? *at* the center of the *earth* to be recycled *and* made into new time .	when time *heals* , where *does it go* ? *it 's* the *end* of *the* universe to *be* recycled *and* made into *space* .
Seq2seq-fill	when time flies , *how* does it go ? *at* the center of the *earth* to be recycled *and* made into new time . how |at |earth |and	when time *is time* , where *is time* ? *time is* the *time* of *time* universe to *the* recycled *be* made into *and* . *the universe* is time |is time |time is |time |time |the |be |and |the universe
Original	when time flies , **where** does it go ? **to** the center of the **universe** to be recycled **and** made into new time .	when time **flies** , where **does it go** ? **to** the **center** of **the** universe to **be** recycled **and** made into **new time** .

Figure 5: Example generations of different models for text infilling on Yahoo Answers. Completions are in italic. Invalid completions are in red. For Seq2seq-fill, we present model outputs along with the merged document.

completions. MLM is trained to independently predict masked tokens instead of jointly modeling them. Even with the target number of ⟨mask⟩ tokens given, its performance is still inferior to BLM. BERT+LM lags behind other models. In BERT training, one mask corresponds to one token, whereas a blank here can cover multiple tokens, and the distance between words is not fixed. Hence, it is difficult for the LM to complete the sentence from BERT representations.

Seq2seq-full has BLEU scores closest to BLM. However, its failure rate ranges from 15% to 40.6% as the mask ratio increases. Seq2seq-fill performs worse than Seq2seq-full, possibly because the decoder has to model segmented text while counting the number of blanks.

In terms of fluency, outputs of BLM, InsT and Seq2seq-full all have perplexity lower than original data perplexity. This is because with beam search, models tend to generate the most typical output with the highest likelihood (Holtzman et al., 2019).

Examination of model generations confirms the superiority of BLM. In Fig. 5, we showcase example outputs by each model at different mask ratios. In low mask ratio settings, models only need to fill in the blanks with a single word to produce grammatical completions. Most models succeed in this task. With a higher mask ratio of 50%, the main ideas of the document are concealed, and the infilling task is much more challenging. Models need to creatively generate sentences that fit the imposed canvas. Although the original meaning of the sentence is not recovered, BLM is the only model able to produce a coherent document with consistency between the question and the answer.

Overall, BLM displays the best performance both quantitatively and qualitatively. Its inherent text infilling ability frees it from length, order, or termination heuristics used by other methods.

4.2 Ancient Text Restoration

Ancient text restoration is a form of text infilling where there are fragments in ancient documents that are illegible due to time-related damages and need to be recovered. Assael et al. (2019) introduces the PHI-ML dataset made of fragments of ancient Greek inscriptions. Restoration is performed at the character-level. The number of characters to recover is assumed to be known and indicated by a corresponding number of '?' symbols, as shown in the second row of Fig. 4. In reality, when epigraphists restore a deteriorated document, the length of the lost fragment is unknown and needs to be guessed as a first step. While models proposed by Assael et al. (2019) relies on expert conjectures, we note that BLM can bypass this limitation and flexibly generate completions without this additional knowledge. However, in order to compute the character error rate (CER) for each '?' and have a fair comparison with previous work, we evaluate our model in the length-aware setting.

	Single-	Multi-slot		
Mask ratio	1%	25%	40%	50%
Human	57.3%	-	-	-
Pythia	32.5%	-	-	-
Pythia-Word	**29.1%**	**36.9%**	42.3%	44.9%
L-BLM	33.7%	37.1%	**37.9%**	**41.6%**

Table 3: CER for ancient text restoration.

Length-aware BLM (L-BLM) We present a variant of BLM adapted to the specific features of this task. The vocabulary V is an alphabet of characters from the ancient Greek language. We extend V with special "__[t]__" tokens that denote the length of the fragment to recover. Specifically, as a preprocessing step, consecutive '?' characters are collapsed into a single "__[t]__" token, where t is the number of '?' symbols. For each such blank token, L-BLM is trained to predict a character to fill in and the length $l \in \{0, \cdots, t-1\}$ of the new blank to its left. The length of the new blank on the right is accordingly $t - 1 - l$.

Dataset The PHI-ML dataset contains about 3 million words / 18 million characters. We evaluate models in two settings: *single-slot* and *multi-slot*. For the single-slot setting, we use the testing script of Assael et al. (2019) which samples a context of length $L = 1000$ from an inscription, then samples a slot of length $C \in [1, 10]$ from that context. The characters from the slot are replaced with '?' and constitute the target. For the multi-slot setting, we progressively increase the number of slots, yielding mask ratios of 25%, 40% and 50% respectively.

Baselines Assael et al. (2019) proposed two models: *Pythia*, a character-level seq2seq-based approach; and *Pythia-Word*, a variant of Pythia that uses both character and word representations as input. During training, the model learns to recover the missing characters of examples where a random slot has been masked. When testing on the multi-slot setting, Pythia(-Word) is applied iteratively with beam size 20 for each slot.

Results Table 3 summarizes the CER of all models in both settings. L-BLM achieves similar CER as Pythia in the single-slot setting, significantly outperforming human experts. Augmented with word representations, Pythia-Word further decreases the error rate compared to character-only methods.

In reality, restoring damaged inscriptions requires reconstructing multiple lost fragments. As a larger proportion of text is missing, Pythia-Word's performance is degraded. L-BLM is robust to the setting change and outperforms Pythia-Word at the mask ratio of 40% and 50% by 4.4 and 3.3 points, respectively. We posit that L-BLM's advantage lies in its ability to maximize the joint likelihood of the completions over all slots. In contrast, Pythia-Word's is only aware of one slot at a time, and beam search is performed locally within each slot.

4.3 Sentiment Transfer

The goal of sentiment transfer is to modify the sentiment of a sentence while maintaining its topic (Shen et al., 2017). An example is described on the third row of Fig. 4. Inspired by the way humans perform rewriting, we follow a recent line of work in style transfer that adopts a two-step approach (Li et al., 2018; Xu et al., 2018; Wu et al., 2019b):

1. Remove words and expressions of high polarity from the source sentence;
2. Complete the partial sentence with words and expressions of the target sentiment.

Specifically, we adapt the *Mask-And-Infill (M&I)* framework of Wu et al. (2019b). We perform Step 1 by training a Bi-LSTM sentiment classifier and masking words whose attention weight is above average. We evaluate the contribution of our model as an infilling module in Step 2 in place of their fine-tuned BERT model. To this end, we train two instances of BLM on the dataset, one for each sentiment. At test time, the corresponding BLM is used to produce completions of the target sentiment.

Wu et al. (2019b) further train the infilling model with the classifier to improve transfer accuracy. They use soft words relaxation to backprop gradients from the classifier to the generator. For BLM, however, we cannot pick locations or insert blanks as "soft" choices, making it challenging to employ a classifier at training time. Nevertheless, we can easily apply the classifier to guide inference. We sample 10 outputs and keep the one with the highest classifier ranking. It is not slower than beam search with size 10 and can be fully parallelized.

Datasets We test on the Yelp and Amazon review datasets (Shen et al., 2017; Li et al., 2018). The Yelp dataset has 450K/4K/1K non-parallel sentences for train/valid/test respectively, and the Amazon dataset has 555K/2K/1K sentences. Each sentence is labeled as either positive or negative.

	Yelp		Amazon	
	ACC	BLEU	ACC	BLEU
Li et al. (2018)	88.7	8.4	48.0	22.8
Zhang et al. (2018)	96.6	22.8	84.1	33.9
Wu et al. (2019a)	91.5	**29.9**	40.2	**41.9**
M&I with MLM	41.5	15.9	31.2	32.1
+ classifier	**97.3**	14.1	75.9	28.5
M&I with BLM	79.6	21.9	52.0	24.7
+ classifier	96.5	21.5	**92.5**	23.1

Table 4: Accuracy and BLEU scores for style transfer.

everyone that i spoke with was **very helpful** and **kind** .
everyone that i spoke with was *rude* and *unprofessional* .
everyone that i spoke with wasn't helpful or kind.

the beans were in the burro in the rice was **nowhere to be** found .
the beans were in the burro in the rice was *the best i* found .
the beans were in the burro and the rice was plentiful

there is **definitely not** enough **room** in that part of the venue .
there is *always* enough *parking* in that part of the venue .
there is so much room in that part of the venue

it is n't **terrible** , but it is **n't** very good either .
it is n't *fancy* , but it is *still* very good either .
it is n't perfect , but it is very good .

Figure 6: Example generations by BLM for sentiment transfer on Yelp. The first line is the source sentence with masked words in bold. The second line is BLM's completion. The third line is a human reference.

Metrics We use evaluation methods introduced by prior work (Shen et al., 2017; Li et al., 2018). To assess the accuracy of generated sentences with respect to the target sentiment, we use a pretrained CNN classifier that achieves 97.7% accuracy on the Yelp dataset and 82.2% accuracy on the Amazon dataset. We also measure the BLEU score between transferred sentences and human references.

Results In Table 4, we can see that directly applying BLM as the infilling module is significantly better than MLM. The accuracy on Yelp and Amazon datasets is increased by 38.1% and 20.8%, respectively. In addition to the aforementioned problem of MLM being trained to predict masked tokens independently, it must generate the same number of tokens as in the source sentence, whereas our BLM formulation is not subject to this constraint. Our simple use of a classifier at inference time further improves accuracy. It achieves the highest accuracy of 92.5% on Amazon with a small decrease in BLEU, indicating that BLM can easily find high-quality outputs.

In Fig. 6, we show examples generated by BLM on Yelp. It can dynamically adapt to the imposed canvas and fill in blanks with expressions of varied lengths, e.g., "**nowhere to be** found" $\rightarrow$ "the best i found" and "**definitely not**" $\rightarrow$ "always". We note that failure cases arise when negative words like "either" are left unmasked; BLM is then unable to produce satisfactory outputs from the canvas.

m	1	10	100	1000
Estimated PPL	46.3	44.4	43.3	42.5

Table 5: The estimated perplexity of BLM with the number of MC samples m on WikiText-103.

	PTB	WT2	WT103
LSTM (Grave et al., 2016)	82.3	99.3	48.7
AWD-LSTM (Merity et al., 2017)	57.3	**65.8**	-
TCN (Bai et al., 2018)	88.7	-	45.2
Transformer (Dai et al., 2019)	-	-	30.1
Adaptive (Baevski and Auli, 2018)	-	-	18.7
Transformer-XL (Dai et al., 2019)	**54.5**	-	**18.3**
InsT (our implementation)	77.3	91.4	39.4
BLM	69.2	81.2	42.5

Table 6: Perplexity on the PTB and WikiText datasets.

4.4 Language Modeling

Language modeling is a special case of text infilling where sequences are generated from scratch. Traditional left-to-right models dominate this task, but are not suitable for text infilling. Conversely, unconventional sequence models are rarely evaluated on language modeling. Here, we study the perplexity of BLM and Insertion Transformer, and compare them with left-to-right language models to provide additional insights.

We use the Monte-Carlo method to estimate the likelihood in Eq. (5) with m samples. While the estimate is unbiased, given that per-word perplexity is a convex function of per-sentence likelihood, sampling estimates like ours are likely yielding a value higher than the actual perplexity (see Appendix B for a proof). As m increases, it converges to the actual perplexity.

Datasets We test on three benchmark datasets: Penn Treebank (PTB) which has about 1M tokens (Mikolov et al., 2010), WikiText-2 (WT2) which has 2M tokens, and WikiText-103 (WT103) which has 103M tokens (Merity et al., 2016).

Results Table 5 shows the trend of estimated PPL with the number of samples m. We choose $m = 1000$ in our evaluation, which is close to convergence. Table 6 summarizes the perplexity of our model in comparison with previous work. The top results are achieved by the Transformer-XL (Dai et al., 2019) and the adaptive embedding method (Baevski and Auli, 2018). They use larger model sizes and supplementary techniques that can also be combined with our model. BLM rivals the Insertion Transformer and outperforms left-to-right language models with LSTM and Temporal Convolutional Network (TCN) architecture. Language modeling seems to still be challenging for free-order models. By reporting the perplexity of unconventional models like BLM, we hope to stimulate future work in this area to close the performance gap with traditional left-to-right models.

5 Conclusion

In this paper, we proposed the Blank Language Model for flexible text generation. Given partially specified text with one or more blanks, BLM will fill in the blanks with a variable number of tokens consistent with the context. We demonstrate the effectiveness of our model on various text rewriting tasks, including text infilling, ancient text restoration and style transfer.

The action of BLM consists of selecting a blank and replacing it with a word and possibly adjoining blanks. We train BLM by optimizing a lower bound on the marginal data likelihood that sums over all possible generation trajectories. In this way, we encourage the model to realize a sentence equally well in all orders, which is suitable for filling arbitrary blanks. Appendix C shows examples generated by BLM along with their trajectories. Depending on the application, we could also train the model to generate in specific orders by placing higher weights on the corresponding trajectories.

BLM has plenty of future applications, including template filling, information fusion, assisting human writing, etc. Moreover, we can extend our formulation to a conditional generative model. Such models can be used in machine translation to support editing and refining translation, as well as in dialogue systems to compose a complete sentence with given elements. While we proposed BLM for language generation, it would also be interesting to compare the representations learned by BLM with those produced by other pre-training methods.

Acknowledgments

We thank all reviewers and the MIT NLP group for their thoughtful feedback.

References

Yannis Assael, Thea Sommerschield, and Jonathan Prag. 2019. Restoring ancient text using deep learning: a case study on Greek epigraphy. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 6368–6375.

Alexei Baevski and Michael Auli. 2018. Adaptive input representations for neural language modeling. *arXiv preprint arXiv:1809.10853*.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

Shaojie Bai, J Zico Kolter, and Vladlen Koltun. 2018. An empirical evaluation of generic convolutional and recurrent networks for sequence modeling. *arXiv preprint arXiv:1803.01271*.

William Chan, Nikita Kitaev, Kelvin Guu, Mitchell Stern, and Jakob Uszkoreit. 2019. Kermit: Generative insertion-based modeling for sequences. *arXiv preprint arXiv:1906.01604*.

Zihang Dai, Zhilin Yang, Yiming Yang, Jaime Carbonell, Quoc V Le, and Ruslan Salakhutdinov. 2019. Transformer-xl: Attentive language models beyond a fixed-length context. *arXiv preprint arXiv:1901.02860*.

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

Chris Donahue, Mina Lee, and Percy Liang. 2020. Enabling language models to fill in the blanks. *arXiv preprint arXiv:2005.05339*.

Chris Dyer, Adhiguna Kuncoro, Miguel Ballesteros, and Noah A Smith. 2016. Recurrent neural network grammars. *arXiv preprint arXiv:1602.07776*.

William Fedus, Ian Goodfellow, and Andrew M Dai. 2018. Maskgan: better text generation via filling in the _. *arXiv preprint arXiv:1801.07736*.

Marjan Ghazvininejad, Omer Levy, Yinhan Liu, and Luke Zettlemoyer. 2019. Constant-time machine translation with conditional masked language models. *arXiv preprint arXiv:1904.09324*.

Edouard Grave, Armand Joulin, and Nicolas Usunier. 2016. Improving neural language models with a continuous cache. *arXiv preprint arXiv:1612.04426*.

Jiatao Gu, James Bradbury, Caiming Xiong, Victor OK Li, and Richard Socher. 2017. Non-autoregressive neural machine translation. *arXiv preprint arXiv:1711.02281*.

Jiatao Gu, Qi Liu, and Kyunghyun Cho. 2019a. Insertion-based decoding with automatically inferred generation order. *Transactions of the Association for Computational Linguistics*, 7:661–676.

Jiatao Gu, Changhan Wang, and Junbo Zhao. 2019b. Levenshtein transformer. In *Advances in Neural Information Processing Systems*, pages 11179–11189.

Ari Holtzman, Jan Buys, Li Du, Maxwell Forbes, and Yejin Choi. 2019. The curious case of neural text degeneration. *arXiv preprint arXiv:1904.09751*.

Mandar Joshi, Danqi Chen, Yinhan Liu, Daniel S Weld, Luke Zettlemoyer, and Omer Levy. 2020. Spanbert: Improving pre-training by representing and predicting spans. *Transactions of the Association for Computational Linguistics*, 8:64–77.

Juncen Li, Robin Jia, He He, and Percy Liang. 2018. Delete, retrieve, generate: a simple approach to sentiment and style transfer. In *NAACL*.

Dayiheng Liu, Jie Fu, Pengfei Liu, and Jiancheng Lv. 2019a. TIGS: An inference algorithm for text infilling with gradient search. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4146–4156.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019b. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*.

Stephen Merity, Nitish Shirish Keskar, and Richard Socher. 2017. Regularizing and optimizing lstm language models. *arXiv preprint arXiv:1708.02182*.

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. 2016. Pointer sentinel mixture models. *arXiv preprint arXiv:1609.07843*.

Tomáš Mikolov, Martin Karafiát, Lukáš Burget, Jan Černockỳ, and Sanjeev Khudanpur. 2010. Recurrent neural network based language model. In *Eleventh annual conference of the international speech communication association*.

Myle Ott, Sergey Edunov, Alexei Baevski, Angela Fan, Sam Gross, Nathan Ng, David Grangier, and Michael Auli. 2019. fairseq: A fast, extensible toolkit for sequence modeling. In *Proceedings of NAACL-HLT 2019: Demonstrations*.

Alexander M Rush, Sumit Chopra, and Jason Weston. 2015. A neural attention model for abstractive sentence summarization. *arXiv preprint arXiv:1509.00685*.

Tianxiao Shen, Tao Lei, Regina Barzilay, and Tommi Jaakkola. 2017. Style transfer from non-parallel text by cross-alignment. In *Advances in neural information processing systems*, pages 6830–6841.

Yong-Siang Shih, Wei-Cheng Chang, and Yiming Yang. 2019. Xl-editor: Post-editing sentences with xlnet. *arXiv preprint arXiv:1910.10479*.

Mitchell Stern, William Chan, Jamie Kiros, and Jakob Uszkoreit. 2019. Insertion transformer: Flexible sequence generation via insertion operations. *arXiv preprint arXiv:1902.03249*.

Qing Sun, Stefan Lee, and Dhruv Batra. 2017. Bidirectional beam search: Forward-backward inference in neural sequence models for fill-in-the-blank image captioning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6961–6969.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in neural information processing systems*.

Alex Wang and Kyunghyun Cho. 2019. Bert has a mouth, and it must speak: Bert as a markov random field language model. *arXiv preprint arXiv:1902.04094*.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, R'emi Louf, Morgan Funtowicz, and Jamie Brew. 2019. Huggingface's transformers: State-of-the-art natural language processing. *ArXiv*, abs/1910.03771.

Chen Wu, Xuancheng Ren, Fuli Luo, and Xu Sun. 2019a. A hierarchical reinforced sequence operation method for unsupervised text style transfer. *arXiv preprint arXiv:1906.01833*.

Xing Wu, Tao Zhang, Liangjun Zang, Jizhong Han, and Songlin Hu. 2019b. Mask and infill: Applying masked language model for sentiment transfer. In *IJCAI*.

Jingjing Xu, Xu Sun, Qi Zeng, Xiaodong Zhang, Xuancheng Ren, Houfeng Wang, and Wenjie Li. 2018. Unpaired sentiment-to-sentiment translation: A cycled reinforcement learning approach. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 979–988.

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Russ R Salakhutdinov, and Quoc V Le. 2019. Xlnet: Generalized autoregressive pretraining for language understanding. In *Advances in neural information processing systems*, pages 5754–5764.

Zichao Yang, Zhiting Hu, Ruslan Salakhutdinov, and Taylor Berg-Kirkpatrick. 2017. Improved variational autoencoders for text modeling using dilated convolutions. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 3881–3890. JMLR. org.

Najam Zaidi, Trevor Cohn, and Gholamreza Haffari. 2020. Decoding as dynamic programming for recurrent autoregressive models. In *International Conference on Learning Representations*.

Zhirui Zhang, Shuo Ren, Shujie Liu, Jianyong Wang, Peng Chen, Mu Li, Ming Zhou, and Enhong Chen. 2018. Style transfer as unsupervised machine translation. *arXiv preprint arXiv:1808.07894*.

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

Appendix

A Implementation Details for Text Infilling Baselines

A.1 Insertion Transformer

We implement the Insertion Transformer in our own framework, using the same Transformer encoder module as for BLM and replacing the prediction layers by Insertion Transformer's mechanism. The canvas is also generated according to the training procedure of Insertion Transformer.

A.2 Masked Language Model

We use the `RobertaForMaskedLM` architecture in the Transformers library for MLM (Wolf et al., 2019; Liu et al., 2019b).

At test time, the model is given an easier version of the text infilling task where blanks are expanded into sequences of $\langle\text{mask}\rangle$ tokens of the target length (or equivalently, the model uses an oracle to predict the length of the infilling).

We experiment with three decoding strategies: (1) one-shot: the model predicts all masks simultaneously (2) left-to-right: the model fills in the masks from left to right (3) confident-first: the model fills one mask at a time that has the highest score. We report results for the confident-first strategy which has the best performance.

A.3 BERT+LM

We use the `bert-base-uncased` model as served by the Transformers library (Wolf et al., 2019; Devlin et al., 2018). The left-to-right language model is a Transformer decoder to predict tokens in a blank. Its input word embedding is concatenated with BERT's output in the blank position at each time step.

A.4 Seq2seq-full and Seq2seq-fill

For both seq2seq baselines, we use Fairseq's `transformer_iwslt_de_en` architecture (Ott et al., 2019). To generate training data, we apply the blanking procedure to the input dataset and generate k copies of each sentence with different masks. We experiment with $k \in \{1, 10, 100\}$ and report the best performance, obtained by $k = 10$.

B Monte-Carlo Estimate of Perplexity

For a sentence x of length n, we estimate $p(x;\theta)$ in Eq. (5) with m samples:

$$X_m = \frac{n!}{m} \sum_{i=1}^{m} p(x, \sigma_i; \theta)$$

where σ_i's are randomly sampled orders.

Note that X_m is an unbiased estimate of $p(x;\theta)$:

$$\mathbb{E}[X_m] = p(x;\theta)$$

The estimated PPL is accordinly:

$$Y_m = X_m^{-1/n}$$

Since $z^{-1/n}$ is a convex function of z,

$$\mathbb{E}[Y_m] = \mathbb{E}[X_m^{-1/n}] \geq \mathbb{E}[X_m]^{-1/n} = p(x;\theta)^{-1/n}$$

i.e., the expectation of the estimated PPL $\geq$ the actual PPL. As m increases, the variance of X_m decreases, and the inequality becomes tighter.

Hence, we will observe that as m increases, the estimated PPL becomes smaller and converges to the real PPL.

C Generation Trajectory

```
____
____ also ____
the ____ also ____
the ____ also ____ choice ____
the salsa ____also ____ choice ____
the salsa was also ____ choice ____
the salsa was also ____ only choice ____
the salsa was also ____ only choice .
the salsa was also my only choice .
```

```
____
____ , ____
____ , ____ terrible ____
____ poor ____ , ____ terrible ____
____ poor ____ , ____ terrible , ____
____ poor ____ , ____ terrible , very ____
____ poor selection , ____ terrible , very ____
very poor selection , ____ terrible , very ____
very poor selection , service terrible , very ____
very poor selection , service terrible , very ____ !
very poor selection , service terrible , very slow !
```

```
____
____ favorite ____
my favorite ____
my favorite ____ pittsburgh ____
my favorite ____ pittsburgh .
my favorite restaurant ____ pittsburgh .
my favorite restaurant in pittsburgh .
```

```
____
____ the ____
____ is ____ the ____
____ is ____ the ____ .
____ is ____ the ____ are ____ .
____ food is ____ the ____ are ____ .
____ food is ____ the ____ are ____ friendly .
____ food is ____ and the ____ are ____ friendly .
____ food is delicious and the ____ are ____ friendly .
____ food is delicious and the ____ are very friendly .
____ food is delicious and the owners are very friendly .
the food is delicious and the owners are very friendly .
```

Figure 7: Examples of BLM generation trajectory on the Yelp review dataset.