TIGS: An Inference Algorithm for Text Infilling with Gradient Search

Dayiheng Liu[†], **Jie Fu**[‡], **Pengfei Liu**[§], **Jiancheng Lv**[†*]
[†]College of Computer Science, Sichuan University
[‡]Mila, IVADO, Polytechnique Montreal
[§]School of Computer Science, Fudan University
losinuris@gmail.com
lvjiancheng@scu.edu.cn

Abstract

Text infilling is defined as a task for filling in the missing part of a sentence or paragraph, which is suitable for many real-world natural language generation scenarios. However, given a well-trained sequential generative model, generating missing symbols conditioned on the context is challenging for existing greedy approximate inference algorithms. In this paper, we propose an iterative inference algorithm based on gradient search, which is the first inference algorithm that can be broadly applied to any neural sequence generative models for text infilling tasks. We compare the proposed method with strong baselines on three text infilling tasks with various mask ratios and different mask strategies. The results show that our proposed method is effective and efficient for fill-in-the-blank tasks, consistently outperforming all baselines.[1]

1 Introduction

Text infilling aims at filling in the missing part of a sentence or paragraph by making use of the past and future information around the missing part, which can be used in many real-world natural language generation scenarios, for example, fill-in-the-blank image captioning (Sun et al., 2017), lexically constrained sentence generation (Liu et al., 2018b), missing value reconstruction (e.g. for damaged or historical documents) (Berglund et al., 2015), acrostic poetry generation (Liu et al., 2018a), and text representation learning (Devlin et al., 2018).

Text infilling is an under-explored challenging task in the field of text generation. Recently, sequence generative models like sequence-to-sequence (seq2seq) models (Sutskever et al., 2014; Bahdanau et al., 2014; Gehring et al., 2017; Vaswani et al., 2017) are widely used in text generation tasks, including neural machine translation (Wu et al., 2016; Vaswani et al., 2017), image captioning (Anderson et al., 2017), abstractive summarization (See et al., 2017), and dialogue generation (Mei et al., 2017). Unfortunately, given a well-trained[2] neural seq2seq model or unconditional neural language model (Mikolov et al., 2010), it is a daunting task to directly apply it to text infilling task. As shown in Figure 1, we observe that the infilled words should be conditioned on past and future information around the missing part, which is contrary to the popular learning paradigm, namely, each output symbol is conditioned on all previous outputs during inference by using unidirectional Beam Search (BS) (Och and Ney, 2004).

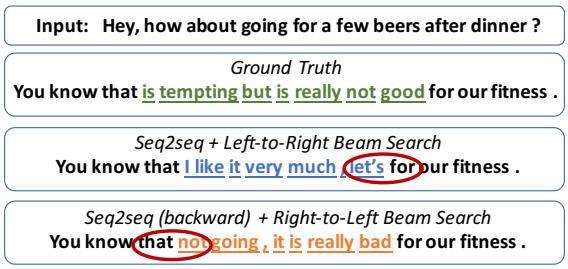

Figure 1: Our key observation on text infilling for Dialogue task. The inability of unidirectional BS to consider both the future and past contexts leads models to fill the blank with words that clash abruptly with the context around the blanks (see red circles).

To solve the issues above, one family of methods for text infilling is "trained to fill in blanks" (Berglund et al., 2015; Fedus et al., 2018; Zhu et al., 2019), which requires large amounts of data in fill-in-the-blank format to train a new model that takes the output template as a conditional in-

[*] Correspondence to Jiancheng Lv.

[1] Our code and data are available at https://github.com/dayihengliu/Text-Infilling-Gradient-Search

[2] Here "well-trained" means that ones focus on popular model settings and data sets, and follow standard training protocols.

Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics, pages 4146–4156
Florence, Italy, July 28 - August 2, 2019.

put. Such methods are only used for unconditional text infilling tasks, whereas many text infilling tasks are conditional, e.g., conversation reply with templates. Another kind of promising approach (Berglund et al., 2015; Wang et al., 2016; Sun et al., 2017) is an inference algorithm that can be directly applied to other generative models. These inference algorithms are applied to Bidirectional RNNs (BiRNNs) (Schuster and Paliwal, 1997; Baldi et al., 1999) which can model both forward and backward dependencies. The latest work is Bidirectional Beam Search (BiBS) (Sun et al., 2017) which proposes an approximate inference algorithm in BiRNN for image caption infilling. However, this method is based on some unrealistic assumptions, such as that given a token, its future sequence of words is independent of its past sequence of words. We experimentally find that these assumptions often generate non-smooth or unreal complete sentences. Moreover, these inference algorithms can be only used to decoders with bidirectional structures, whereas almost all sequence generative models use a unidirectional decoder. As a result, it is highly expected to develop an inference algorithm that could be applied to the unidirectional decoder.

In this paper, we study the general inference algorithm for text infilling to answer the question:

- *Given a well-trained neural sequence generative model, is there any inference algorithm that can effectively fill in the blanks in the output sequence?*

To investigate such a possibility, we propose a dramatically different inference approach called Text Infilling with Gradient Search (TIGS), in which we search for infilled words based on gradient information to fill in the blanks. To the best of our knowledge, **this could be the first inference algorithm** that does not require any modification or training of the model and can be broadly used in any sequence generative model to solve the fill-in-the-blank tasks as verified in our experiments.

To be specific, we treat the blanks to be filled as parameterized vectors during inference. More concretely, we first randomly or heuristically project each blank to a valid token and initialize its parameterized vector with the word embedding of the valid token. The goal is seeking the words to be infilled by minimizing the negative log-likelihood (NLL) of the complete sequence. Then the algorithm alternately performs optimization step (**O-step**) and projection step (**P-step**) until convergence. In **O-step**, we fix all other parameters of the model and only optimize the blank parameterized vectors by gradients. In **P-step**, heuristics like local search and projected gradient are used to project the blank parameterized vectors to valid tokens (i.e., discretization).

The contribution and novelty of this work could be summarized as below:

- We propose an iterative inference algorithm based on gradient search, which could be the first inference algorithm that can be broadly applied to any neural sequence generative models for text infilling tasks.
- Extensive experimental comparisons show the effectiveness and efficiency of the proposed method on three different text infilling tasks, compared with five state-of-the-art methods.

2 Related Works

There are some effective solutions to the text infilling task: a) training a model specifically for text infilling tasks (Berglund et al., 2015; Fedus et al., 2018; Zhu et al., 2019); b) using standard sequence generative model with modified inference algorithm (Berglund et al., 2015; Wang et al., 2016; Sun et al., 2017).

As one typical work of the first category, NADE (Berglund et al., 2015) is proposed to train a specific BiRNNs for filling in blanks, which concatenates an auxiliary vector to input vectors for indicating a missing input during training and inference. Fedus et al. (2018) propose MaskGAN which 1) uses some specific "missing" tokens to indicate the blanks and takes the whole sequence with blanks (called template) as the input of encoder, and 2) uses an RNN as a decoder to generate the whole sentence after filling in the blanks. Similarly, Zhu et al. (2019) use self-attention model (Vaswani et al., 2017), which takes the template as the input for unconditional text infilling task. One major limitation of these works is that they require large amounts of data in fill-in-the-blank format and need to train a specific model. Besides, they are only used for unconditional text infilling tasks. Different from them, our new inference algorithm does not require any modification or training of the model, which can be broadly applied to any neural

seq2seq models for both conditional and unconditional text infilling tasks.

As with the second category, some inference algorithms based on BiRNNs have been proposed for fill-in-the-blank tasks thanks to their ability to model both forward and backward dependencies. For example, Berglund et al. (2015) propose Generative Stochastic Networks (GSN) to reconstruct the blanks of sequential data. The idea is to first randomly initialize the symbols in the blanks and then resample an output y_t from $\mathrm{P}_{BiRNN}(y_t \mid \{y_d\}_{d \neq t}, x)$ one at a time until convergence. More recently, Sun et al. (2017) propose the Bidirectional Beam Search (BiBS) inference algorithm of BiRNNs for fill-in-the-blank image captioning task. However, this method is based on some strong assumptions, which may be violated in practice. As shown in our experiments, we provide empirical analysis on cases where this approach fails. Moreover, GSN and BiBS can be only applied to decoders with bidirectional structures, while almost all sequence generative models use a unidirectional decoder. In contrast, our proposed inference method decouples from these assumptions and can be applied to the unidirectional decoder.

3 Preliminary

Since our method utilizes gradient information, it could smoothly cooperate with other architectures, such as models proposed in (Vaswani et al., 2017; Gehring et al., 2017). Considering the popularity of RNNs, and the related work is based on RNN model, we use RNN-based models as a showcase in this paper.

3.1 RNN-based Seq2Seq Model

We firstly introduce the notations and briefly describe the standard RNN-based seq2seq model. Let $\boldsymbol{x} = \{\boldsymbol{x}_1, \boldsymbol{x}_2, ..., \boldsymbol{x}_n\}$ denotes one-hot vector representations of the conditional input sequence, $\boldsymbol{y} = \{y_1, y_2, ..., y_m\}$ denotes scalar indices of the corresponding target sequence, and $\mathcal{V}$ denotes the vocabulary. n and m represent the length of the input sequence and the output sequence, respectively.

The seq2seq model is composed of an encoder and a decoder. For the encoder part, each $\boldsymbol{x}_t$ will be firstly mapped into its corresponding word embedding $\boldsymbol{x}_t^{emb}$. Then $\{\boldsymbol{x}_t^{emb}\}$ are input to a bidirectional or unidirectional long short-term memory (LSTM) (Hochreiter and Schmidhuber, 1997) RNN to get a sequence of hidden states $\{\boldsymbol{h}_t^{enc}\}$.

For the decoder, at time t, similarly y_t is first mapped to $\boldsymbol{y}_t^{emb}$. Then a context vector $\boldsymbol{c}_t$ is calculated with attention mechanism (Bahdanau et al., 2014; Luong et al., 2015) $\boldsymbol{c}_t = \sum_{i=1}^{n} \boldsymbol{a}_{ti} \boldsymbol{h}_i^{enc}$, which contains useful latent information of the input sequence. Here, $\boldsymbol{a}_t$ is an attention distribution vector to decide which part to focus on. The context vector $\boldsymbol{c}_t$ and the embedding $\boldsymbol{y}_t^{emb}$ are fed as input to a unidirectional RNN language model (LM), which will output a probability distribution of the next word $P(y_{t+1}|\boldsymbol{y}_{1:t}, \boldsymbol{x})$, where $\boldsymbol{y}_{1:t}$ refers to $\{y_1, ..., y_t\}$.

During training, the negative log-likelihood (NLL) of the target sequence is minimized using standard maximum-likelihood (MLE) training with stochastic gradient descent (SGD), where the NLL is calculated as follows:

$$-\log P(\boldsymbol{y}|\boldsymbol{x}) = -\sum_{t=1}^{m} \log P(y_t|\boldsymbol{y}_{1:t-1}, \boldsymbol{x}). \quad (1)$$

During inference, the decoder needs to find the most likely output sequence $\boldsymbol{y}^*$ by giving the input $\boldsymbol{x}$:

$$\boldsymbol{y}^* = \arg\max_{\boldsymbol{y}} P(\boldsymbol{y}|\boldsymbol{x}). \quad (2)$$

Since the number of possible sequences grows as $|\mathcal{V}|^m$ ($|\mathcal{V}|$ is the size of vocabulary), exact inference is NP-hard and approximate inference algorithms like left-to-right greedy decoding or beam search (BS) (Och and Ney, 2004) are commonly used.

3.2 Problem Definition

In this paper, instead of setting some restrictions such as limiting the number of blanks or restricting the position of the blanks in previous work (Sun et al., 2017), we consider a more general case of text infilling task where the number and location of blanks are arbitrary.

Let $\underline{B}$ be a placeholder for a blank, $\mathbb{B}$ be the set that records all the blanks' position index, and $\boldsymbol{y}^{\mathbb{B}}$ be a target sequence where portions of text body are missing as indicated by $\mathbb{B}$. For instance, if a target sequence has two blanks at the position i and j, then $\mathbb{B} = \{i, j\}$ and $\boldsymbol{y}^{\mathbb{B}} = \{y_1, .., y_{i-1}, \underline{B}, y_{i+1}, ..., y_{j-1}, \underline{B}, y_{j+1}, ..., y_m\}$.

Given an input sequence $\boldsymbol{x}$ and a target sequence $\boldsymbol{y}^{\mathbb{B}}$ containing blanks indicated by $\mathbb{B}$, we aim at filling in the blanks of $\boldsymbol{y}^{\mathbb{B}}$. This procedure needs to consider the global structure of sentences

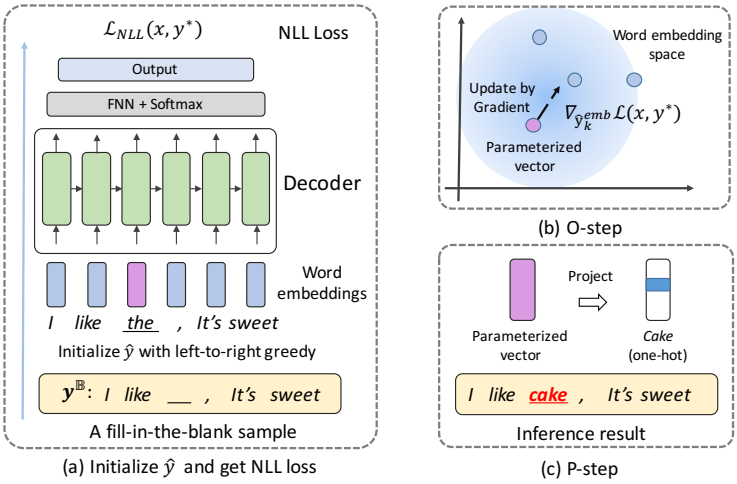

Figure 2: Overall framework.

and provide meaningful details under the condition $\boldsymbol{x}$.

4 Methodology

In this section, we present our inference method in detail. The overall framework is shown in Figure 2. Given a well-trained seq2seq model and a pair of text infilling data ($\boldsymbol{x}$, $\boldsymbol{y}^{\mathbb{B}}$), the method aims at finding an infilled word set $\hat{\boldsymbol{y}} = \{\hat{y}_1, ..., \hat{y}_{|\mathbb{B}|}\}$ to minimize the NLL of the complete sentence $\boldsymbol{y}^*$ via:

$$\hat{\boldsymbol{y}} = \arg\min_{\hat{y}_j \in \mathcal{V}} \mathcal{L}_{NLL}(\boldsymbol{x}, \boldsymbol{y}^*), \tag{3}$$

where $\boldsymbol{y}^*$ denotes the complete sentence after filling the blanks of $\boldsymbol{y}^{\mathbb{B}}$ with $\hat{\boldsymbol{y}}$, $|\mathbb{B}|$ denotes the number of blanks. Since the number of possible infilled word set is $|\mathcal{V}|^{|\mathbb{B}|}$, naïvely searching in this space is NP-hard.

Our key idea is to utilize the gradient information to narrow the range of search during inference. This idea is similar to the "white-box" adversarial attacks (Goodfellow et al., 2014b; Szegedy et al., 2013; He and Glass, 2018). However, the adversarial attack aims to slightly modify the inputs in order to mislead the model to make wrong predictions, while our goal is to search for the reasonable words that should be filled into the blanks.

Unlike the continuous input space (e.g., images) in other tasks, applying gradient search directly to the input would make it invalid (i.e., no longer a one-hot vector) for text infilling tasks. More specifically, we firstly treat the blanks to be filled as parameterized word embedding vectors $\hat{\boldsymbol{y}}^{emb} = \{\hat{\boldsymbol{y}}_1^{emb}, ..., \hat{\boldsymbol{y}}_{|\mathbb{B}|}^{emb}\}$. Then, we fix the parameters of the well-trained model and only optimize these parameterized vectors in the continuous space, where the gradient information can be used to minimize the NLL loss $\mathcal{L}_{NLL}(\boldsymbol{x}, \boldsymbol{y}^*)$. Finally, the $\hat{\boldsymbol{y}}^{emb}$ is discretized into valid words $\hat{\boldsymbol{y}}$ by measuring the distance between the $\hat{\boldsymbol{y}}^{emb}$ and the word embeddings in $\mathbb{W}^{emb}$. Here $\mathbb{W}^{emb}$ denotes the word embedding matrix in the decoder of the well-trained seq2seq model, and each column of $\mathbb{W}^{emb}$ represents the word embedding of one word in the vocabulary.

As every word in the set $\hat{\boldsymbol{y}}$ is dependent on each other, the simultaneous discretization of all parameterized word embeddings in $\hat{\boldsymbol{y}}^{emb}$ into valid words at the same time usually make the complete sentence $\boldsymbol{y}^*$ non-smooth. As a concrete example, when infilling the two blanks in "Amy likes eating __ __, so she goes to snack bars very often.", the $\hat{\boldsymbol{y}}^{emb}$ may be close to the word embeddings of {"ice", "cream"} and {"fried", "chips"}. However, if one discretizes the two blanks simultaneously, one might get answers like {"ice", "chips"} or {"fried", "cream"}. Therefore, we adopt an iterative algorithm which is similar to Gibbs sampling. At each inference step, we focus on one single infilled word $\hat{y}_j$ for j-th blank and update it while keeping other words in the infilled word set $\hat{\boldsymbol{y}}$ fixed. For the unknown blank length tasks (each blank may contain an arbitrary unknown number of tokens), we can apply the TIGS as a black box inference algorithm over a range of blank lengths and then rank these solutions.

At the beginning, we initialize the infilled word set $\hat{\boldsymbol{y}}$ with some valid words randomly or heuris-

tically (from a left-to-right beam search). Then we perform optimization step (**O-step**) and projection step (**P-step**) alternately to update each infilled word in the infilled word set $\hat{\boldsymbol{y}}$ until convergence or reach the maximum number of rounds T.

In **O-step**, we aim to optimize the $\hat{\boldsymbol{y}}_j^{emb}$ in continuous space using gradient information with respect to $\mathcal{L}_{NLL}(\boldsymbol{x}, \boldsymbol{y}^*)$. Firstly, we get the complete sentence $\boldsymbol{y}^*$ by filling $\hat{\boldsymbol{y}}$ in the blanks of $\boldsymbol{y}^{\mathbb{B}}$ and obtain the $\mathcal{L}_{NLL}(\boldsymbol{x}, \boldsymbol{y}^*)$ of $\boldsymbol{y}^*$ after putting $\boldsymbol{x}$ into the encoder and $\boldsymbol{y}^*$ into the decoder of the well-trained seq2seq model. Then we treat the vector $\hat{\boldsymbol{y}}_j^{emb}$ as parameterized vector, and fix all other parameters of the seq2seq model and only optimize the parameterized vector $\hat{\boldsymbol{y}}_j^{emb}$ with gradient information to minimize $\mathcal{L}_{NLL}(\boldsymbol{x}, \boldsymbol{y}^*)$.

However, directly optimizing $\mathcal{L}_{NLL}(\boldsymbol{x}, \boldsymbol{y}^*)$ may lead to the final $\hat{\boldsymbol{y}}_j^{emb}$ not like a feasible word embedding in $\mathbb{W}^{emb}$, and its nearest neighbor word embedding in $\mathbb{W}^{emb}$ could be far away from it. So we add an $L2$ penalty to make the $\hat{\boldsymbol{y}}_j^{emb}$ get close to $\mathbb{W}^{emb}$:

$$\mathcal{L}(\boldsymbol{x}, \boldsymbol{y}^*) = \mathcal{L}_{NLL}(\boldsymbol{x}, \boldsymbol{y}^*) + \lambda \cdot \sum_j \left\| \hat{\boldsymbol{y}}_j^{emb} \right\|_2, \quad (4)$$

where λ is a hyperparameter. We also tried to add an additional regularization term that directly narrow the distance between $\hat{\boldsymbol{y}}_j^{emb}$ and its nearest word embedding in $\mathbb{W}^{emb}$, which is used in Cheng et al. (2018) for seq2seq adversarial attacks, but no obvious improvement was found.

Given the loss $\mathcal{L}(\boldsymbol{x}, \boldsymbol{y}^*)$, $\hat{\boldsymbol{y}}_j^{emb}$ is updated with $\nabla_{\hat{\boldsymbol{y}}_j^{emb}} \mathcal{L}(\boldsymbol{x}, \boldsymbol{y}^*)$ by one-step gradient descent:

$$\hat{\boldsymbol{y}}_j^{emb} = \hat{\boldsymbol{y}}_j^{emb} - \alpha \cdot \nabla_{\hat{\boldsymbol{y}}_j^{emb}} \mathcal{L}_{NLL}(\boldsymbol{x}, \boldsymbol{y}^*). \quad (5)$$

Instead of updating $\hat{\boldsymbol{y}}_j^{emb}$ by naïvely SGD algorithm, we experimentally find that Nesterov (Sutskever et al., 2013) optimizer performed better than other optimizers to update $\hat{\boldsymbol{y}}_j^{emb}$. As discussed in Dong et al. (2017b), this momentum based optimizer can stabilize update directions and escape from poor local maxima during the iterations for adversarial attack.

In **P-step**, we aim to project the $\hat{\boldsymbol{y}}_j^{emb}$ into a valid infilled word $\hat{y}_j$. A naïve way is to find the word whose word embedding in $\mathbb{W}^{emb}$ is nearest to $\hat{\boldsymbol{y}}_j^{emb}$ based on the distance metric function $dist(\cdot)$[3]. However, due to its high dimensionality, the obtaining word embedding may be far from satisfactory. Instead, similar to the idea of beam search, we first obtain a set S containing K candidate words whose word embedding is K-nearest to $\hat{\boldsymbol{y}}_j^{emb}$:

$$S = \underset{y_k \in \mathcal{V}}{\text{nearest-K}}\, dist(\hat{\boldsymbol{y}}_j^{emb}, \boldsymbol{y}_k^{emb}), \quad (6)$$

and then we select one word with lowest NLL from these K words in S as $\hat{y}_j$. Our experiments suggest that just setting the size of K to 1% of the vocabulary size works well.

The whole algorithm is further summarized in 1. Since our method is designed for the unidirectional decoder, the time complexity is expected to be slightly higher than that of the inference algorithm designed for the bidirectional decoder. In brief, our approach requires $mKT|\mathbb{B}|$ RNN steps, while the GSN (Berglund et al., 2015) requires $mT|\mathbb{B}|$ BiRNN steps, and the BiBS (Sun et al., 2017) requires $2mKT$ RNN steps. Fortunately, our inference algorithm can be easily optimized with GPUs.

Algorithm 1 TIGS algorithm

Input: a trained seq2seq model, a pair of text infilling data $(\boldsymbol{x}, \boldsymbol{y}^{\mathbb{B}})$, output length m.
Output: a complete output sentence $\boldsymbol{y}^*$.
Initialize the infilled word set $\hat{\boldsymbol{y}}$ and initialize $\boldsymbol{y}^*$ by infilling $\boldsymbol{y}^{\mathbb{B}}$ with $\hat{\boldsymbol{y}}$.
Initialize $\hat{\boldsymbol{y}}^{emb}$ by looking up the word embedding matrix $\mathbb{W}^{emb}$.
for $t = 1, 2, \ldots, T$ **do**
  for $j = 1, 2, \ldots, |\mathbb{B}|$ **do**
    O-step:
    Update $\hat{\boldsymbol{y}}_j^{emb}$ with gradient $\nabla_{\hat{\boldsymbol{y}}_j^{emb}} \mathcal{L}(\boldsymbol{x}, \boldsymbol{y}^*)$
    P-step:
    Set $S = \underset{y_k \in \mathcal{V}}{\text{nearest-K}}\, dist(\hat{\boldsymbol{y}}_j^{emb}, \boldsymbol{y}_k^{emb})$
    Set $\hat{y}_j = \underset{\hat{y}_j \in S}{\arg\min}\, \mathcal{L}_{NLL}(\boldsymbol{x}, \boldsymbol{y}^*)$
  end for
  Update $\boldsymbol{y}^*$ with $\hat{y}_j$
  if convergence **then**
    break
  end if
end for
return $\boldsymbol{y}^*$

5 Experiments

5.1 Datasets

In the experiments, we evaluate the proposed method on three text infilling tasks with three widely used publicly available corpora.

The first task is **conversation reply with a template** (denoted as Dialog) which is conducted on the DailyDialog (Li et al., 2017) dataset. We use

[3]Through experiments we find that using Euclidean distance as metric function $dist(\cdot)$ perform slightly better than Cosine distance for our method.

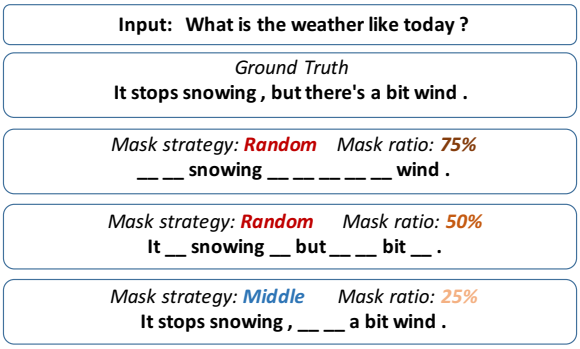

Figure 3: Some testing samples of conversation reply with templates task with different mask strategies and ratios.

its single-turn data, which contains 82,792 conversation pairs. The query sentence is taken as encoder input $\boldsymbol{x}$, and the reply sentence is taken as $\boldsymbol{y}$.

The second task is **Chinese acrostic poetry generation** (denoted as Poetry). Here we use a publicly available Chinese poetry dataset[4] which contains 232,670 Chinese four-line poems. For each poem, the first two lines are used as encoder input $\boldsymbol{x}$, and the last two lines are $\boldsymbol{y}$.

The third task is **infilling product reviews** (denoted as APRC). The Amazon Product Reviews Corpus (APRC) (Dong et al., 2017a), which is built upon Amazon product data (McAuley et al., 2015) and contains 347,061 reviews, is used in this task. Unlike the first two tasks, this task is an unconditional text infilling task (without conditional input $\boldsymbol{x}$). We use each product review in Dong et al. (2017a) as $\boldsymbol{y}$.

For each task, we take 5,000 samples in the test set to construct the data with blanks ($\boldsymbol{y}^{\mathbb{B}}$) for testing, we create a variety of test samples by masking out text $\boldsymbol{y}$ with varying missing ratios and two mask strategies. More specifically, the first mask strategy is called **middle** which is followed as the setting in Sun et al. (2017), namely, removing $r = 25\%$, 50%, or 75% of the words from the middle of $\boldsymbol{y}$ for each data. The second mask strategy is called **random**, namely, randomly removing $r = 25\%$, 50%, or 75% of the words in $\boldsymbol{y}$ for each data. To sum up, we have three test tasks, and each task has six types of test sets (two mask strategies and three mask ratios). Each test set contains 5,000 test samples. We show some data examples in Figure 3.

[4] https://github.com/chinese-poetry/chinese-poetry

5.2 Baselines

We compare our approach **TIGS** with several strong baseline approaches:

Seq2Seq-f: it runs beam search (BS) with beam width K on a well-trained seq2seq model (forward) to fill the blanks from left to right.

Seq2Seq-b: it runs BS with beam width K on a well-trained seq2seq model (backward) to fill the blanks from right to left.

Seq2Seq-f+b: it fills the blanks by both Seq2Seq-f and Seq2Seq-b, and then selects the output with a maximum of the probabilities assigned by the seq2seq models. This method is used in Wang et al. (2016).

BiRNN-GSN: it runs GSN (Berglund et al., 2015) on a well-trained seq2seq model with BiRNN as the decoder to fill the blanks.

BiRNN-BiBS: it runs bidirectional beam search (BiBS) (Sun et al., 2017) on a well-trained seq2seq model with BiRNN as the decoder to fill the blanks. The method has achieve the state-of-the-art results on fill-in-the-blank image captioning task in Sun et al. (2017).

Except for BiRNN-GSN and BiRNN-BiBS, all the above baselines and our method perform inference on the same well-trained seq2seq model. BiRNN-GSN and BiRNN-BiBS perform inference on a well-trained seq2seq model in which the decoder is BiRNN. These models are trained on the complete sentence dataset with standard maximum-likelihood. Moreover, the sentences with blanks are only used in the inference stage. For fari comparison, BiRNN-BiBS, BiRNN-GSN, and the proposed method use the same initialization strategy (left-to-right greedy). The maximum number of iterations T is set to 50 to ensure that all the algorithms can achieve their best performance.

In addition to the above inference based approaches, we also compare two model-based approaches: **Mask-Seq2Seq** and **Mask-Self-attn** (Fedus et al., 2018; Zhu et al., 2019). These baselines take the output template as an additional input and are trained on the data in fill-in-the-blank format. We use LSTM RNNs for Mask-Seq2Seq, and use the self-attention model (Vaswani et al., 2017) for Mask-Self-attn (Zhu et al., 2019) which is shown to have better performance than GAN-based models (Goodfellow et al., 2014a) for text infilling.

Datasets	Metrics	Methods	r=25%		r=50%		r=75%	
			Random	Middle	Random	Middle	Random	Middle
Dialog	NLL	Seq2Seq-f	3.573	3.453	3.653	3.316	3.328	2.975
		Seq2Seq-b	3.657	3.558	3.911	3.542	3.713	3.421
		Seq2Seq-f+b	3.397	3.321	3.491	3.213	3.233	2.932
		BiRNN-BiBS	3.248	3.279	3.268	3.294	3.245	3.217
		BiRNN-GSN	3.239	3.270	3.219	3.199	3.086	2.938
		Mask-Seq2Seq	3.406	3.368	3.434	3.347	3.279	3.177
		Mask-Self-attn	3.567	3.524	3.694	3.466	3.509	3.205
		TIGS (ours)	**3.143**	**3.164**	**3.050**	**3.030**	**2.920**	**2.764**
	BLEU	Template	0.780	0.823	0.621	0.700	0.552	0.601
		Seq2Seq-f	0.834	0.861	0.670	0.737	0.584	0.640
		Seq2Seq-b	0.837	0.862	0.675	0.739	0.584	0.627
		Seq2Seq-f+b	0.860	0.881	0.692	0.751	0.594	0.643
		BiRNN-BiBS	0.828	0.852	0.661	0.725	0.575	0.626
		BiRNN-GSN	0.894	0.892	**0.726**	0.752	0.600	0.643
		Mask-Seq2Seq	0.867	0.887	0.719	0.769	0.614	**0.662**
		Mask-Self-attn	0.858	0.864	0.719	0.743	**0.623**	0.643
		TIGS (ours)	**0.895**	**0.894**	0.724	**0.754**	0.596	0.644
Poetry	NLL	Seq2Seq-f	4.107	4.022	3.901	3.642	3.430	3.294
		Seq2Seq-b	4.180	4.124	4.051	3.837	3.638	3.511
		Seq2Seq-f+b	4.021	3.994	3.825	3.630	3.390	3.275
		BiRNN-BiBS	3.939	3.966	3.735	3.701	3.476	3.430
		BiRNN-GSN	3.953	3.976	3.739	3.652	3.405	3.296
		Mask-Seq2Seq	4.103	4.071	3.996	3.886	3.738	3.637
		Mask-Self-attn	4.052	4.028	3.911	3.810	3.666	3.548
		TIGS (ours)	**3.860**	**3.912**	**3.601**	**3.567**	**3.268**	**3.181**
	BLEU	Template	0.727	0.815	0.581	0.687	0.508	0.559
		Seq2Seq-f	0.779	0.842	0.629	0.704	0.536	0.576
		Seq2Seq-b	0.774	0.835	0.623	0.702	0.534	0.576
		Seq2Seq-f+b	0.789	0.844	0.635	0.705	0.538	0.577
		BiRNN-BiBS	0.776	0.836	0.625	0.702	0.533	0.575
		BiRNN-GSN	0.802	0.848	0.648	**0.707**	0.541	**0.579**
		Mask-Seq2Seq	0.785	0.843	0.635	0.705	0.537	0.577
		Mask-Self-attn	0.790	0.845	0.640	0.706	0.539	**0.579**
		TIGS (ours)	**0.805**	**0.850**	**0.650**	**0.707**	**0.542**	**0.579**
APRC	NLL	Seq2Seq-f	3.554	3.129	3.687	2.650	3.068	2.122
		Seq2Seq-b	3.694	3.215	4.039	2.826	3.494	2.349
		Seq2Seq-f+b	3.354	3.002	3.515	2.553	2.962	2.045
		BiRNN-BiBS	2.999	3.001	2.943	2.759	2.733	2.456
		BiRNN-GSN	2.969	2.967	2.907	2.515	2.628	2.012
		Mask-Seq2Seq	3.080	2.983	2.951	2.567	2.472	2.088
		Mask-Self-attn	3.002	2.946	2.847	2.551	2.448	2.085
		TIGS (ours)	**2.831**	**2.857**	**2.722**	**2.394**	**2.451**	**1.913**
	BLEU	Template	0.503	0.692	0.127	0.432	0.009	0.182
		Seq2Seq-f	0.781	0.897	0.623	0.881	0.682	0.879
		Seq2Seq-b	0.779	0.896	0.616	0.872	0.683	0.864
		Seq2Seq-f+b	0.812	0.905	0.658	0.887	0.703	**0.884**
		BiRNN-BiBS	0.867	0.896	0.715	0.869	0.740	0.856
		BiRNN-GSN	0.879	0.904	0.751	0.884	0.736	0.882
		Mask-Seq2Seq	0.860	0.900	0.750	0.856	0.754	0.835
		Mask-Self-attn	0.878	**0.914**	**0.778**	0.882	**0.778**	0.870
		TIGS (ours)	**0.883**	0.911	0.774	**0.889**	0.769	0.878

Table 1: BLEU and NLL results.

5.3 Metrics

Following Sun et al. (2017), we compare methods on standard sentence-level metric BLEU scores (4-gram) (Papineni et al., 2002) which considers the correspondence between the ground truth and the complete sentences. However, such a metric also has some deficiencies in text infilling tasks. For example, given two complete sentences with only one word different, the sentence level statistics of them may be quite similar, whereas a human can clearly tell which one is most natural. Moreover, given a template, there may be several reasonable ways to fill in the blanks. For example, given a template, "i __ this book, highly recommend it", it is reasonable to fill the word "love" or "like" in the blank. However, since there is only one ground truth, the BLEU scores of these two complete sentences are quite different. We find that this issue is more severe for the unconditional text filling task which has fewer restrictions, leading to more ways of filling in the blanks.

Therefore, for the unconditional text filling task (APRC), instead of calculating the BLEU score with only the ground truth as the reference, we also follow Yu et al. (2017) and use 10,000 sentences which are randomly sampled from the test set as references to calculate BLEU scores to evaluate the fluency of the complete sentences.

Besides BLEU scores, we conduct a model-based evaluation. We train a conditional LM for each task (unconditional LM for APRC task) and use its NLL to evaluate the quality of the complete sentence y^*given the input x.

5.4 Results

The BLEU (the higher the better) and NLL (the lower the better) results are shown in Table 1. Generally, we find that bidirectional methods (BiRNN-BiBS, BiRNN-GSN, and Seq2Seq-f+b) outperform unidirectional ones (Seq2Seq-f and Seq2Seq-b) in most cases. The model-based methods (Mask-Seq2Seq and Mask-Self-attn) perform well on unconditional text infilling task (APRC), but slightly poorly on conditional text infilling tasks (Dialog and Poetry). In line with the evaluation results in Zhu et al. (2019), the Mask-Self-attn performs consistently better than Mask-Seq2Seq. It has also achieved the highest BLEU score in some cases of unconditional text infilling tasks. However, in most cases of conditional text infilling tasks, the proposed method performs better than Mask-Self-attn.

Since the goal of the proposed method TIGS is to find the complete sentence with minimal NLL by utilizing gradient information. As expected, it achieves the lowest NLL in all cases of all tasks. Also, the BLEU scores of TIGS is highest in most cases of conditional text infilling tasks, while BiRNN-GSN and BiRNN-BiBS provide comparable performance. Although TIGS is used in RNN-based seq2seq model, it still achieves very competitive BLEU results on unconditional text infilling task compare with Mask-Self-attn.

Methods	Dialog	Poetry	APRC
BiRNN-BiBS	1.524	1.478	1.558
BiRNN-GSN	2.979	2.675	2.261
Mask-Self-attn	2.270	2.727	3.042
TIGS	**3.226**	**3.120**	**3.137**

Table 2: Human evaluation resutls

5.5 Human Evaluation

We also conduct the human evaluation to further compare TIGS, BiRNN-BiBS, BiRNN-GSN, and Mask-Self-attn. Following the setting in Zhu et al. (2019), we collect generations of each of the four methods on 50 randomly-selected test instances. Then we launch a crowd-sourcing online study, asking 10 evaluators to rank the generations. The method with the best generation receives a score of 4, and the other three methods receive scores of 3, 2, and 1 according to the rank, respectively. The results are shown in Table 2. We can see that TIGS consistently outperforms all baselines.

Template	**love __ book and __ __ club series**
Ground Truth	love this book and the camel club series
Seq2seq-f	love love book and ca n't club series
Seq2seq-b	love next book and 's murder club series
Mask-Seq2Seq	love this book and the murder club series
Mask-Self-attn	love this book and the book club series
BiRNN-BiBS	love this book and the book club series
BiRNN-GSN	love this book and all the club series
TIGS	love this book and the motorcycle club series
Template	**really __ this __ __ __ believable plot .**
Ground Truth	really enjoyed this futuristic book . believable plot .
Seq2seq-f	really enjoyed this book . the believable plot .
Seq2seq-b	really with this development and a believable plot .
Mask-Seq2Seq	really enjoyed this fast paced and believable plot
Mask-Self-attn	really enjoyed this book .. very believable plot .
BiRNN-BiBS	really enjoyed this story and a believable plot .
BiRNN-GSN	really enjoyed this book and a believable plot .
TIGS	really enjoyed this book . very believable plot .
Template	**so __ better __ the __ one . __ __ . __ now i __ <num> more __ __ __ __ submit .**
Ground Truth	so much better than the last one . really good . and now i need <num> more words before i can submit .
Seq2seq-f	so far better than the first one . ca n't . wait now i have <num> more books to read . submit .
Seq2seq-b	so getting better for the next one . it down . .. now i write <num> more words so i can submit .
Mask-Seq2Seq	so much better than the first one . loved it . and now i have <num> more words to submit this submit .
Mask-Self-attn	so much better than the first one . loved it . now now i need <num> more words to describe and submit .
BiRNN-BiBS	so much better than the first one . loved it . and now i have <num> more to read . i submit .
BiRNN-GSN	so much better than the first one . i cried . so now i have <num> more words to go to submit .
TIGS	so much better than the first one . highly recommend . but now i need <num> more words to go to submit .

Figure 4: Example outputs of different methods on APRC task.

5.6 Samples and Analysis

Figure 4 and 5 show some qualitative examples of APRC and Dialog tasks. Because the inability of Seq2Seq-f and Seq2Seq-b to reason about the past and future simultaneously. We can see that Seq2Seq-f and Seq2Seq-b usually generate sentences that do not satisfy grammatical rules and are not fluent. Seq2Seq-f struggle to reason about word transitions on the forward side of the blank, so the words filled in by Seq2Seq-f usually abruptly clash with existing words behind the blank. Similarly, the words filled in by Seq2Seq-b usually abruptly clash with existing words before

Input (Query)	**can you study with the radio on ?**
Template	**__ , __ listen __ __ music .**
Ground Truth	no , I listen to background music .
Seq2seq-f	i'd , I'm listen to the music .
Seq2seq-b	music , can listen to the music .
Mask-Seq2Seq	yes , they listen to the music .
Mask-Self-attn	yes , it's a lot of music .
BiRNN-BiBS	I , to listen to the music .
BiRNN-GSN	yes , I'll listen to the music .
TIGS	yes , I listen to classical music .
Input (Query)	**pretty good , thanks . i'm going to see my uncle .**
Template	**__ __ then __ and keep __ touch .**
Ground Truth	good bye then . and keep in touch .
Seq2seq-f	nice to then . and keep your touch .
Seq2seq-b	minutes . then go and keep in touch .
Mask-Seq2Seq	ok , then go then keep in touch .
Mask-Self-attn	then . then keep and keep in touch .
BiRNN-BiBS	you <UNK> then <UNK> and keep it touch .
BiRNN-GSN	ok , then go and keep in touch .
TIGS	alright . then . and keep in touch .
Input (Query)	**don ' t do that again . you are riding the tiger .**
Template	**no problem __ __ can __ with __ __ .**
Ground Truth	no problem . i can deal with it well .
Seq2seq-f	no problem . i can ' with my boss .
Seq2seq-b	no problem think i can up with trouble <UNK> .
Mask-Seq2Seq	no problem . you can stay with me down .
Mask-Self-attn	no problem . i can do it that . .
BiRNN-BiBS	no problem . i can just with the <UNK> .
BiRNN-GSN	no problem . i can help with my baggage .
TIGS	no problem . i can deal with my bags .

Figure 5: Example outputs of different methods on Daily task.

the blank.

BiRNN-BiBS makes assumption that $P(y_t|\boldsymbol{y}_{1:t-1}, \boldsymbol{y}_{t+1:m}, \boldsymbol{x}) = P_{\overrightarrow{\text{URNN}}}(y_t|\boldsymbol{y}_{1:t-1}, \boldsymbol{x}) \cdot P_{\overleftarrow{\text{URNN}}}(y_t|\boldsymbol{y}_{t+1:m}, \boldsymbol{x})$. This assumption may cause some sentences generated by BiRNN-BiBS are non-smooth or unreal. For example, in the top instance, the BiRNN-BiBS generates a non-smooth sentence "i, to listen to the music". At the third time-step, because both $P_{\overrightarrow{\text{URNN}}}(y_3 =\text{"to"}|y_{4:m} =\text{"listen to the music"}, \boldsymbol{x})$ and $P_{\overleftarrow{\text{URNN}}}(y_3 =\text{"to"}|y_{1:2} =\text{"i,"}, \boldsymbol{x})$ are relatively large, resulting in this blank being filled with an inappropriate word "to" by BiRNN-BiBS. However, $P(y_3 =\text{"to"}|y_{1:2} =\text{"i,"}, y_{4:m} =\text{"listen to the music"}, \boldsymbol{x})$ should be lower. In addition, we find that BiRNN-BiBS tends to use the unknown token "<unk>" to fill the blanks compared to other methods. The reason we analyze may be that sometimes both $P_{\overrightarrow{\text{URNN}}}(y_t=\text{"<unk>"}|\boldsymbol{y}_{1:t-1}, \boldsymbol{x})$ and $P_{\overleftarrow{\text{URNN}}}(y_t=\text{"<unk>"}|\boldsymbol{y}_{t+1:m}, \boldsymbol{x})$ would be relatively large.

As for Mask-Seq2Seq and Mask-Self-attn, although they directly take the template $\boldsymbol{y}^{\mathbb{B}}$ as an additional input and are trained with data in fill-in-the-blank format. We experimentally found that the generalization ability of these models is still limited, especially for conditional text infilling tasks. In the Dialog task, 21% and 16% of the samples generated by Mask-Self-attn and Mask-Seq2Seq with beam search could not even reconstruct the template (see Figure 5).[5]

Because the BiRNN-GSN fills the blank from the probability $P_{\text{BiRNN}}(y_t|\boldsymbol{y}_{1:t-1}, \boldsymbol{y}_{t+1:m}, \boldsymbol{x})$, and the proposed method filling the blank directly with the gradient $\nabla_{\hat{\boldsymbol{y}}_t^{emb}} \mathcal{L}(\boldsymbol{x}, \boldsymbol{y}^*)$. Both of them have the ability to reason about the past and future simultaneously without any unrealistic assumptions. We can see that the complete sentences generated by them are better than all other algorithms. However, BiRNN-GSN uses the bidirectional structure as the decoder, which makes it challenging to apply to most sequence generative models, but the proposed method is gradient-based, which can be broadly used in any sequence generative models.

6 Conclusions

In this paper, we propose a general inference algorithm for text infilling. To the best of our knowledge, the method is the first inference algorithm that does not require any modification or training of the model and can be broadly used in any sequence generative model to solve the fill-in-the-blank tasks. We compare the proposed method and several strong baselines on three text infilling tasks with various mask ratios and different mask strategies. The results show that the proposed method is an effective and efficient approach for fill-in-the-blank tasks, consistently outperforming all baselines.

Acknowledgment

This work is supported by the National Key R&D Program of China under contract No. 2017YFB1002201, the National Natural Science Fund for Distinguished Young Scholar (Grant No. 61625204), and partially supported by the State Key Program of National Science Foundation of China (Grant Nos. 61836006 and 61432014).

[5]For BLEU and NLL evaluation, we force them to reconstruct the template during inference.

References

Peter Anderson, Basura Fernando, Mark Johnson, and Stephen Gould. 2017. Guided open vocabulary image captioning with constrained beam search. In *EMNLP*.

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

Pierre Baldi, Søren Brunak, Paolo Frasconi, Giovanni Soda, and Gianluca Pollastri. 1999. Exploiting the past and the future in protein secondary structure prediction. *Bioinformatics*.

Mathias Berglund, Tapani Raiko, Mikko Honkala, Leo Kärkkäinen, Akos Vetek, and Juha T Karhunen. 2015. Bidirectional recurrent neural networks as generative models. In *NIPS*.

Minhao Cheng, Jinfeng Yi, Huan Zhang, Pin-Yu Chen, and Cho-Jui Hsieh. 2018. Seq2sick: Evaluating the robustness of sequence-to-sequence models with adversarial examples. *arXiv preprint arXiv:1803.01128*.

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*.

Li Dong, Shaohan Huang, Furu Wei, Mirella Lapata, Ming Zhou, and Ke Xu. 2017a. Learning to generate product reviews from attributes. In *ACL*.

Yinpeng Dong, Fangzhou Liao, Tianyu Pang, Hang Su, Xiaolin Hu, Jianguo Li, and Jun Zhu. 2017b. Boosting adversarial attacks with momentum. *arXiv preprint arXiv:1710.06081*.

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

Jonas Gehring, Michael Auli, David Grangier, Denis Yarats, and Yann N Dauphin. 2017. Convolutional sequence to sequence learning. *arXiv preprint arXiv:1705.03122*.

Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron C. Courville, and Yoshua Bengio. 2014a. Generative adversarial nets. In *NIPS*.

Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. 2014b. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*.

Tianxing He and James Glass. 2018. Detecting egregious responses in neural sequence-to-sequence models. *arXiv preprint arXiv:1809.04113*.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*.

Yanran Li, Hui Su, Xiaoyu Shen, Wenjie Li, Ziqiang Cao, and Shuzi Niu. 2017. Dailydialog: A manually labelled multi-turn dialogue dataset. In *ICLR*.

Dayiheng Liu, Quan Guo, Wubo Li, and Jiancheng Lv. 2018a. A multi-modal chinese poetry generation model. In *IJCNN*.

Dayiheng Liu, Jiancheng Lv, Feng He, and Yifan Pu. 2018b. Bfgan: Backward and forward generative adversarial networks for lexically constrained sentence generation. *arXiv preprint arXiv:1806.08097*.

Minh-Thang Luong, Hieu Pham, and Christopher D Manning. 2015. Effective approaches to attention-based neural machine translation. *arXiv preprint arXiv:1508.04025*.

Julian McAuley, Rahul Pandey, and Jure Leskovec. 2015. Inferring networks of substitutable and complementary products. In *SIGKDD*.

Hongyuan Mei, Mohit Bansal, and Matthew R Walter. 2017. Coherent dialogue with attention-based language models. In *AAAI*.

Tomáš Mikolov, Martin Karafiát, Lukáš Burget, Jan Černockỳ, and Sanjeev Khudanpur. 2010. Recurrent neural network based language model. In *INTERSPEECH*.

Franz Josef Och and Hermann Ney. 2004. The alignment template approach to statistical machine translation. *Computational linguistics*.

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

Mike Schuster and Kuldip K Paliwal. 1997. Bidirectional recurrent neural networks. *IEEE Transactions on Signal Processing*.

Abigail See, Peter J Liu, and Christopher D Manning. 2017. Get to the point: Summarization with pointer-generator networks. In *ACL*.

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 *CVPR*.

Ilya Sutskever, James Martens, George E. Dahl, and Geoffrey E. Hinton. 2013. On the importance of initialization and momentum in deep learning. In *ICML*.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *NIPS*.

Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. 2013. Intriguing properties of neural networks. *arXiv preprint arXiv:1312.6199*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *NIPS*.

Cheng Wang, Haojin Yang, Christian Bartz, and Christoph Meinel. 2016. Image captioning with deep bidirectional lstms. In *ACM Multimedia*.

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, et al. 2016. Google's neural machine translation system: Bridging the gap between human and machine translation. *arXiv preprint arXiv:1609.08144*.

Lantao Yu, Weinan Zhang, Jun Wang, and Yong Yu. 2017. Seqgan: Sequence generative adversarial nets with policy gradient. In *AAAI*.

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