An Exploration of Text Infilling Techniques and Custom Mask Design

Anonymous Authors[1]

Abstract

Text infilling is the task of predicting and inserting missing segments of text within a given context. This study explores three approaches to tasks related to infilling: ILM (Donahue et al., 2020), HIER (Ippolito et al., 2019), and TIGS (Liu et al., 2019). The implementation is an extension of the work (Donahue et al., 2020), with the goal of exploring and understanding the mechanisms that enable language models to fill in blanks. The main steps involve creating a custom mask, retraining the language model, evaluating its performance, and comparing it with the original trained model provided in the paper. The results show that the retrained model generates outputs that are grammatically correct, semantically coherent, and make sense in context. In addition, it validates that the ILM model can consider both preceding and subsequent text.

1. Introduction

In the realm of natural language processing, the ability of language models to effectively fill in gaps or missing information is crucial for tasks such as text completion and generation. When editing or revising text, people usually do not write in linear order. Therefore, there is a need for models that can effectively infill text considering both past and future text. In both (Donahue et al., 2020) and (Liu et al., 2019), the researchers aim to tackle this problem. Building upon the foundation laid by (Donahue et al., 2020), who focused on language model adaptations for filling in blanks, the implementation introduces a custom mask on common nouns. Using the custom mask, the implementation follows the ILM framework to fine-tune a language model on the infilling examples. The goal is to observe whether the mask can simplify the infilling task and create more targeted and contextually meaningful infilling options. Moreover, learn how the ILM model works and examine if it can effectively generate infillment considering both preceding and subsequent text.

2. Background

Language models (LM) play a crucial role in natural language processing (NLP) tasks. They are a type of artificial intelligence (AI) model trained on vast amounts of textual data to learn the patterns, relationships, and structures of language. They can then generate coherent and contextually relevant text based on a given prompt or input. After pretraining, language models can be fine-tuned on specific tasks or domains to enhance their performance on particular applications, such as sentiment analysis, language translation, summarization, and infilling.

3. Review of paper to implement or extend

Paper of choice: Chris Donahue, Mina Lee, and Percy Liang. 2020. Enabling Language Models to Fill in the Blanks. In Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, pages 2492–2501, Online. Association for Computational Linguistics.

3.1. Storyline

High-level motivation/problem The goal of this research is to enable language models to effectively fill in missing spans of text in a document, a task referred to as text infilling. The motivation for this research is to extend the capabilities of language models beyond traditional language modeling, making them useful tools for various applications, such as assisting in text editing, connecting fragmented ideas, and restoring documents.

Prior work on this problem

- Prior research in this area has primarily focused on language modeling. Language models like GPT-2 (Radford et al., 2019) and GPT-3 (Brown et al., 2020) have been capable of generating coherent text but have limitations in infilling accurately, since they can only leverage context in one direction, which is usually the past.
- Bidirectional models like BERT (Devlin et al., 2019) and SpanBERT (Joshi et al., 2020) have improved infilling by considering both preceding and subsequent text, but they are limited to fixed-length spans.
- The self-attention model presented in (Zhu et al., 2019)

can handle variable-length spans but requires specialized architectures that cannot easily leverage large-scale pre-trained models.

Research gap The research gap identified in this paper is the need for a simple and general infilling framework that allows language models to infill variable-length spans effectively while preserving their advantages in generation quality, efficient sampling, and conceptual simplicity. Existing methods either have limitations in handling variable-length spans or require specialized architectures, and there is a need for a model-agnostic approach that leverages large-scale pre-trained language models.

Contributions The paper introduced the concept of a framework called *Infilling by Language Modeling* (ILM). The framework provides a straightforward formulation of the infilling task, making it compatible with existing language model architectures.

3.2. Proposed solution

The paper addresses the research gap by proposing ILM, a simple and quick approach to text infilling. An infilling task involves generating the complete text x based on incomplete text $\bar{x}$, where [blank] acts as a placeholder for a continuous span. To effectively map $\bar{x}$ to x, an infilling strategy needs to determine the quantity and type of tokens to generate for each [blank]. Thus, the goal is to learn the distribution $p(y|\bar{x})$. The ILM framework consists of two main steps:

1. Formulation: This step reparametrizes the infilling task to predict only the missing spans y that replace the [blank] tokens in $\bar{x}$. Then it is possible to replace the [blank] tokens in $\bar{x}$ with predicted spans y in a deterministic fashion to generate x. To accommodate multiple variable-length spans, we represent y as the concatenation of all missing spans separated by [answer] tokens (with one [answer] token assigned per [blank]). Consequently, the task of infilling can be framed as the learning of $p(y|\bar{x})$ without any loss of generality.

2. Training: To manufacture an infilling instance for a given x, the framework initially samples a $\bar{x}$ from a stochastic function mask(x), which randomly replaces a certain number of spans in x with [blank] tokens. Then, concatenate the spans that underwent replacement, with [answer] tokens serving as separators, to create a training target denoted as y. The complete infilling example is constructed by concatenating input $\bar{x}$, a special separator token [sep], and target y. Finally, the framework employs standard LM training methodologies to fine-tune the infilling samples, resulting in models in the form $p_\theta(y|\bar{x})$.

The framework is shown in Figure 1. There are several advantages of this framework. First, it considers both preceding and subsequent text. Second, compared to language modeling, it has almost no computational overhead. In addition, the framework requires minimal alterations to the existing vocabulary of a language model. The framework's simplicity, efficiency, and ability to leverage pre-trained LMs make it a promising solution for text infilling tasks.

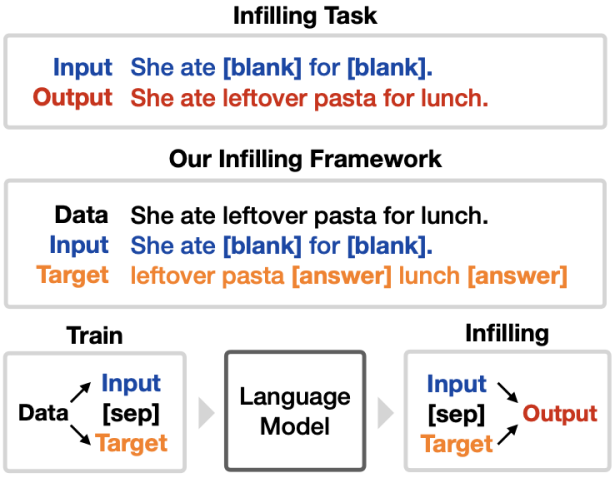

Figure 1. Task of infilling. Figure 1 from section 1 in (Donahue et al., 2020)

3.3. Claims-Evidence

Claim 1 ILM outperforms models that only consider past or future context in the sentence infilling task, and achieves similar performance to language modeling based on all available context (LM-All) with less memory usage.

Evidence 1 The quantitative evaluation results in Table 1 show that ILM achieves better test set perplexity (PPL) compared to models that only consider past (LM) or future (LM-Rev) context on all three datasets (ROC stories, CS paper Abstracts, and song lyrics). In addition, Table 1 reports the average length of all test set examples in tokens relative to the original sequence. The results demonstrate that ILM achieves similar PPL to LM-All while using shorter sequence lengths, which results in less memory usage.

Claim 2 ILM not only expands the potential of LMs by repurposing them for infilling tasks, but it also preserves their original capabilities,

Evidence 2 Document infilling PPL of all models is presented in Table 2. Scratch indicates that the baselines were initialized from scratch, otherwise, they are from the pre-trained checkpoint across the three datasets. Observe that the PPL of ILM is comparable to LM, suggesting that our

Table 1. Quantitative evaluation results. The table reports the test set perplexity (PPL) on the sentence infilling task for different model configurations on all three datasets. The Length is the average length of all test set examples in tokens relative to that of the original sequence. Table 1 from section 4.2 (Donahue et al., 2020).

	STO	ABS	LYR	Length
LM	18.3	27.9	27.7	1.00
LM-Rev	27.1	46.5	34.3	1.00
LM-All	15.6	22.3	21.4	1.81
ILM	15.6	22.4	22.6	1.01

infilling approach can tenably sustain language modeling capabilities while expanding infill capabilities.

Table 2. Document infilling PPL of all models on the three datasets. Table 3 from appendix D.1 of (Donahue et al., 2020).

	STO	ABS	LYR
LM (scratch)	33.4	52.1	25.1
LM-Rev (scratch)	32.9	53.9	24.7
LM-All (scratch)	30.4	44.6	26.2
ILM (scratch)	30.8	45.3	30.6
LM	17.6	25.7	20.8
LM-Rev	25.1	36.7	23.7
LM-All	17.8	25.2	21.5
ILM	18.1	23.9	23.0

Claim 3 ILM produces sentences that are harder for humans to recognize as machine-generated compared to other baseline strategies (LM, BERT, and SA).

Evidence 3 Table 3 shows the human evaluation results. The ILM model attains a superior score compared to all other models. Notably, the maximum achievable score is effectively 80%, given that a flawless model would prompt annotators to randomly select one of the five sentences.

Table 3. Human evaluation results. The project employs BERT, SA, LM, and ILM models to substitute random sentences within five-sentence stories from the ROC stories test set. Subsequently, human evaluators are asked to determine the sentence generated by a machine among the five. The score denotes the percentage of stories where the human evaluator incorrectly identifies the machine-generated sentence. Table 2 from section 6 (Donahue et al., 2020).

	BERT	SA	LM	ILM
Score%	20	29	41	45

3.4. Critique and Discussion

The most insightful contribution is the introduction of text infilling as a versatile task. The claims about the effectiveness of the ILM framework in infilling text are well-supported by the experimental results and the comparison with other models. The paper mainly focuses on a few specific domains for evaluation (short stories, scientific abstracts, and song lyrics). While this provides a good starting point, it would be interesting to see how the proposed method generalizes across a wider range of domains.

4. Review of paper to implement or extend

Paper of choice: Daphne Ippolito, David Grangier, Chris Callison-Burch, and Douglas Eck. 2019. Unsupervised Hierarchical Story Infilling. In Proceedings of the First Workshop on Narrative Understanding, pages 37–43, Minneapolis, Minnesota. Association for Computational Linguistics.

4.1. Storyline

High-level motivation/problem The goal of this research is to address the challenge of story infilling. The authors aim to create a model that can predict missing words or text spans in stories in a manner that balances fluency, coherence, and diversity. The ultimate vision is to develop interactive tools for creative writing that can assist human writers in a way that is more fluid and sophisticated than existing methods.

Prior work on this problem

- Automatic Story Generation: Early research in automatic story generation utilized classical AI algorithms based on symbolic and logical planning and graph construction. Later, statistical and story generation with neural models methods were introduced. These methods focused on generating complete stories rather than infilling specific text spans.

- Text Infilling: Rather than generating entire stories, some research has focused on text infilling, which is known as the cloze task. This task involves removing words or sequences of words from a text and asking a computer or human to predict them. Previous studies have employed word masking for constructing language models (Fedus et al., 2018) and contextualized word embeddings (Devlin et al., 2019). Additionally, investigations into bidirectional decoding for image captioning (Sun et al., 2017) have addressed the infilling of longer spans.

Research gap The research gap in this paper is the need for a model that can perform the story infilling task effectively. Specifically, the gap is in achieving a balance between fluency, coherence, and diversity, which is crucial for practical applications in creative writing. The existing models have difficulty with this balance, and the paper aims to address this gap by proposing a hierarchical two-stage model for story infilling.

Contributions Overall, the paper's contributions lie in the novel approach to story infilling, introducing a hierarchical model. The paper provides supported evidence that this approach improves the quality and diversity of generated text, with potential applications in interactive creative writing assistance.

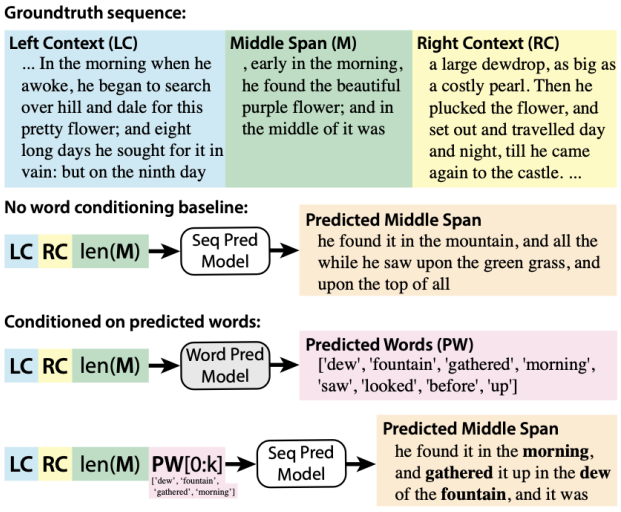

Figure 2. Comparison of one-stage baseline and the two-stage method. Table 1 from section 1 in (Ippolito et al., 2019)

4.2. Proposed solution

This paper addresses the research gap in story infilling, which involves predicting missing words or text spans in stories. As shown in Figure 2, in a one-stage baseline, the prediction involves determining the missing span based on the target length and context. In contrast, the paper introduces a hierarchical model (HIER) that decomposes the generation task into two stages:

1. Word Prediction: The word prediction model is a standard Transformer, it prepares the training data and reorders the subwords by increasing frequency. The model ingests the context data for each infilling instance, and predicts a sequence of subwords in frequency order, starting with rare words.

2. Word-Conditioned Generation: Also a Transformer, this stage is responsible for generating a text span given the surrounding context, a desired length marker, and a list of words predicted by the first-stage model. The concatenation of these three signals is taken as input. During the training phase, the model chooses a set of k words from the missing span to condition on, with k being uniformly sampled from 0 to half of the target length. During inference, the model takes the conditioning words from the first-stage model.

Organizing by frequency serves a dual purpose. First, predicting the existence of common words becomes more straightforward when there are nearby rare words compared to the reverse scenario. Secondly, prioritizing the prediction of rare words enables the model to halt decoding after a certain number of steps. Subsequently, we can assign the task of predicting more common words to our second-stage model.

Training with a varying number of conditioning words provides the flexibility to select the number of supplied words during the inference phase. It is observed that this selection must balance an adequate amount of information to impact coherence and novelty in the generated spans. Simultaneously, it should maintain space for the Word-Conditioned Generation model to propose its own common words and generate coherent text.

4.3. Claims-Evidence

Claim 1 HIER achieves higher diversity than baseline models.

Evidence 1 Table 4 presents automated evaluation results, comparing HIER with the baseline non-hierarchical approach using different decoding strategies. As the dist-1 and dist-2 numbers are higher, it is apparent that the HIER method outperforms the other baseline models in terms of diversity.

Claim 2 Human evaluators prefer the model conditioned on only the first three predicted words from the HIER model.

Evidence 2 As shown in Table 4, human evaluation results show that the model conditioned on the first three predicted words (HIER-3) is preferred by human raters.

Claim 3 HIER achieves a balance between fluent and coherent text generation while maintaining diversity.

Evidence 3 Table 4 demonstrates that human evaluators assign similar ratings to the generation of HIER-3 when compared to BASE sampling10, while the HIER model attains significantly greater diversity.

Table 4. Automated and human evaluation for our method (Hier) against baseline (base). Table 2 from section 4 (Ippolito et al., 2019)

Model	Decoding	dist-1	dist-2	ROUGE-1	PPL	% Votes against HIER-3	p-value
BASE	beam10	.057	.218	0.29	16.61	48.75	0.82
BASE	sampling10	.058	.304	0.26	16.61	56.67	0.30
BASE	sampling	.101	.477	0.23	16.61	27.78	0.000025
HIER-max	sampling+beam10	.107	.442	0.24	4.22	28.33	0.00079
HIER-3	sampling+beam10	.104	.347	0.27	6.62	-	-

4.4. Critique and Discussion

The hierarchical approach to story infilling, where rare words are first selected and then used to condition the generation of text, is an effective way to balance fluency, coherence, novelty, and diversity. The claims about the advantages of the HIER framework in infilling text in stories are well-supported by the experimental results and the comparison with other models. However, although the paper mentions that the hierarchical method can be extended to other text generation tasks, no experimental results or evidence are provided to support this claim, leaving room for further exploration and validation.

5. Review of paper to implement or extend

Paper of choice: Dayiheng Liu, Jie Fu, Pengfei Liu, and Jiancheng Lv. 2019. 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, Florence, Italy. Association for Computational Linguistics.

5.1. Storyline

High-level motivation/problem The high-level motivation of this research is to address the task of text infilling, which involves filling in the missing part of a sentence or paragraph. This task is relevant in various real-world natural language generation scenarios, such as missing value reconstruction, poetry generation, and text representation learning. The larger goal is to develop an effective and efficient method for text infilling that can be broadly applied to different neural sequence generative models.

Prior work on this problem Prior research has explored different approaches to text infilling:

- Approaches that train a model tailored specifically for tasks related to infilling: NADE (Berglund et al., 2015), MaskGan (Fedus et al., 2018), SA (Zhu et al., 2019). A significant constraint in these studies is their reliance on extensive fill-in-the-blank formatted data and the necessity to train a dedicated model. Moreover, they are restricted to unconditional text infilling tasks.
- Approaches that modify the inference algorithm within standard sequence generative models: GSN (Berglund et al., 2015) and BiBS (Sun et al., 2017). However, these approaches have limitations in handling conditional text infilling, and they are generally limited to decoders with bidirectional structures.

Research gap The research gap lies in the absence of a versatile and efficient inference algorithm that can be broadly applied to any neural sequence generative model for both conditional and unconditional text infilling tasks. Existing methods either require specialized models trained on specific data formats or rely on strong assumptions about token dependencies that may not hold in practical scenarios. There is a need for an inference algorithm that does not necessitate model modification or extensive training, offering a more generalized solution.

Contributions The paper aims to bridge the research gap in text infilling by presenting an innovative and versatile inference algorithm, which could have significant implications for improving text generation models across a range of practical applications. The paper proposes an iterative inference algorithm called *Text Infilling with Gradient Search* (TIGS), which applies to various neural sequence generative models for text infilling tasks. It introduces a parameterized vector approach for infilling blanks during inference, allowing the model to predict missing words by minimizing negative log-likelihood (NLL) while considering past and future information.

5.2. Proposed solution

This paper introduces a novel approach to address the research question, the framework is shown in Figure 3. It is designed to find the most probable words to complete sentences with missing parts using a gradient-based method. Here, x represents the input sequence, $|\mathbb{B}|$ is the number of blanks, and $|\mathbb{V}|$ is the size of the vocabulary. $\hat{y}^{emb} = \{\hat{y}_1, ..., \hat{y}_{|\mathbb{B}|}\}$ denotes the target sentence with

blanks for infilling. Given a trained seq2seq model along with a pair of text infilling data $(x, y^{\mathbb{B}})$, the primary aim is to find the infilled word set $\hat{y}$ to minimize the negative log-likelihood (NLL) of the complete sentence y^*. This is mathematically expressed as:

$$\hat{y} = \arg\min_{\hat{y}_j \in \mathcal{V}} \mathcal{L}_{NLL}(x, y^*) \quad (1)$$

Exploring this space for possible infilled word sets by brute force, which has a size of $|\mathbb{V}|^{\mathbb{B}}$, is a computationally challenging task due to its NP-hard nature. Therefore, the paper aims to narrow the range of search during inference by utilizing the gradient information.

Next, the algorithm represents the blanks to be filled as parameterized word embedding vectors, denoted as $\hat{y}^{emb} = \{\hat{y}_1^{emb}, ..., \hat{y}_{|\mathbb{B}|}^{emb}\}$. The word embedding matrix in the decoder of the trained model is represented by $\mathbb{W}^{emb}$. Each column of the matrix corresponds to the word embedding of an individual word in the vocabulary. The parameters of the trained model are then kept fixed, with optimization focused solely on these parameterized vectors within the continuous space. This allows the utilization of gradient information to minimize NLL loss $\mathcal{L}_{NLL}(x, y^*)$. Last, by measuring the distance between the $\hat{y}^{emb}$ and the word embedding in $\mathbb{W}^{emb}$, $\hat{y}^{emb}$ is discretized to valid words $\hat{y}$.

In tasks where the length of the unknown blank is not predetermined, allowing for an arbitrary number of tokens in each blank, TIGS can be employed as a black box inference algorithm across a spectrum of blank lengths. Subsequently, these solutions can be ranked to determine the most suitable one. First, the algorithm initializes the infilled word set $\hat{y}$ with valid words chosen randomly or heuristically by a left-to-right beam search. Then, it alternates between optimization steps (O-step) and projection steps (P-step) to update each infilled word in the set $\hat{y}$ until convergence or having reached the maximum number of rounds.

The complete TIGS algorithm is presented in Figure 4. As the method is tailored for a unidirectional decoder, it has a higher time complexity compared to the inference algorithms designed for a bidirectional decoder. However, the time complexity can be optimized by GPUs.

5.3. Claims-Evidence

Claim 1 By utilizing gradient information, the TIGS algorithm effectively finds the complete sentence with minimal NLL.

Evidence 1 The experimental results in Table 1 in section 5 of (Liu et al., 2019) show that TIGS achieves significantly lower negative log-likelihood (NLL) scores compared to other baselines, indicating better performance in generating infilled sentences. TIGS also achieves higher BLEU scores, indicating better fluency and similarity to the ground truth sentences.

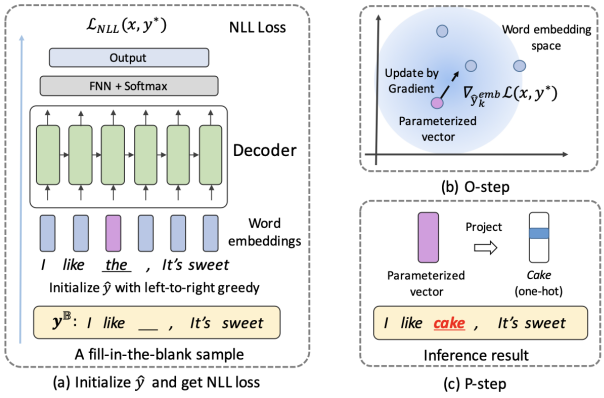

Figure 3. Framework of TIGS. Figure 2 from section 4 in (Liu et al., 2019)

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 $\hat{y}_j = \arg\min_{\hat{y}_j \in S} \mathcal{L}_{NLL}(\boldsymbol{x}, \boldsymbol{y}^*)$
  end for
  Update $\boldsymbol{y}^*$ with $\hat{y}_j$
  if convergence **then**
    break
  end if
end for
return $\boldsymbol{y}^*$

Figure 4. TIGS Algorithm. From section 4 in (Liu et al., 2019)

Claim 2 TIGS can be broadly applied to different neural sequence generative models.

Evidence 2 TIGS does not require modification or training of the model, making it applicable to any neural sequence generative models without the need for specific model training. The experimental results in Table 1 in section 5 of (Liu et al., 2019) show that TIGS can be applied to both unidirectional and bidirectional decoders, addressing the limitations of existing inference algorithms.

Claim 3 Human evaluators prefer the generations of the TIGS method compared to other methods.

Evidence 3 Table 5 demonstrates the result of human evaluation. The results show that TIGS consistently outperforms all baselines.

Table 5. Human evaluation results. The human evaluation involved collecting generations from four models on 50 randomly selected test instances. A crowd-sourcing online study was conducted with 10 evaluators who ranked the generations. The evaluators assigned scores of 4, 3, 2, and 1 based on the rank. Table 2 section 5 in (Liu et al., 2019)

	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

5.4. Critique and Discussion

This inference algorithm is unique in that it does not require any modification or training of the model and can be broadly applied to any neural sequence generative model. The approach is also interesting as it addresses the challenge of text infilling by leveraging both past and future information. However, in terms of the experimental setup, the authors conducted the evaluation on 50 randomly-selected test instances and used 10 evaluators for the human evaluation. While this provides some insights into the performance of the methods, the small number of test instances and evaluators may limit the accuracy of the findings. Increasing the sample size and the number of evaluators could strengthen the experimental setup.

6. Implementation

6.1. Implementation motivation

The main goal of this implementation is to investigate how language models perform infilling tasks, with a focus on understanding the effects of a custom mask function. The implementation aims to investigate whether concentrating on a particular part of speech, specifically common nouns, enhances the model's ability to generate more precise and improved infilling options when compared to the original model, which masks all parts of speech. Additionally, the study aims to examine the ILM model's performance in generating infillment by considering both preceding and subsequent text. Successful experiments will help determine whether the custom mask simplifies the infilling task, offering contextually meaningful options, and will help validate the unique traits of the ILM model. Expectations suggest that masks tailored to common nouns will result in superior word selection when filling in common noun blanks.

6.2. Implementation plan and setup

The plan is to ensure successful training with the custom mask on common nouns and evaluate the model's infilling performance. Common nouns are referred to as NN for singular and NNS for plural in the Natural Language Toolkit (NLTK) for part of speech tagging and tokenization. The major steps of the plan are as follows:

1. Clone repository provided from (Donahue et al., 2020).
2. Set up environment and install requirements.
3. Create a custom mask for common nouns.
4. Prepare and process the ROC stories dataset using the custom mask, following the process of Figure 5.
5. Train a new model with the infilling examples created in the previous step.
6. Perform infilling task with the new model.
7. Perform infilling task with the trained model from the ILM paper.
8. Compare the results.

The highest priority is given to successfully retrain the model with the custom mask, followed by performing infilling tasks and comparing results. The plan addresses the motivation by exploring and understanding the impact of the custom mask function on the model while providing insights into the adaptability and limitations of the custom mask function.

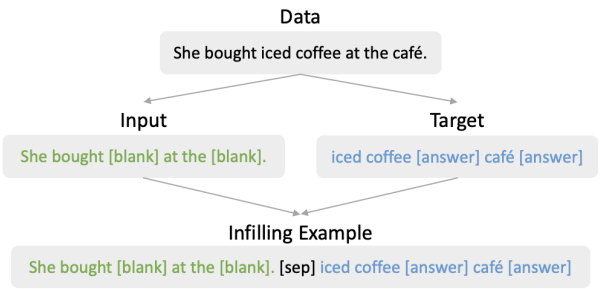

Figure 5. Process of producing infilling examples for implementation.

RM context a	Leo's Birthday It was Leo's Birthday yesterday. A birthday party was held at< |infill_common_noun| >. He received a< |infill_common_noun| > from his classmates.
RM output a	Leo's Birthday It was Leo's Birthday yesterday. A birthday party was held at school. He received a present from his classmates.
RM context b	Leo's Birthday It was Leo's Birthday yesterday. A birthday party was held at< |infill_common_noun| >. He received a< |infill_common_noun| > from his co-workers.
RM output b	Leo's Birthday It was Leo's Birthday yesterday. A birthday party was held at work. He received a card from his co-workers.
RM context c	Romantic Date On the way to meet his girlfriend, Leo stopped by a< |infill_common_noun| > shop. He bought a< |infill_common_noun| > for her.
RM output c	Romantic Date On the way to meet his girlfriend, Leo stopped by a lady shop. He bought a dress for her.
RM other outputs c	(wine, bottle) (girl, dress) (woman, necklace) (movie, movie) (girl, necklace)
TM context c	Romantic Date On the way to meet his girlfriend, Leo stopped by a< |infill_word| > shop. He bought a< |infill_word| > for her.
TM output c	Romantic Date On the way to meet his girlfriend, Leo stopped by a coffee shop. He bought a vanilla ice for her.
TM other outputs c	(souvenir, watch) (Japanese, sashimi) (massage, massage) (grocery, bowl) (Chinese, ring)

Table 6. Implementation outputs of retrained model (RM) with custom mask and original trained model (TM).

6.3. Implementation details

Code base: The implementation is extended from the repository `https://github.com/chrisdonahue/ilm` provided in (Donahue et al., 2020). The code base fine-tunes a language model to infill blanks.

Datasets: The trained model provided in the paper was originally trained on three datasets:

- Abstracts (200K examples, 30M words): Abstracts from CS papers on arXiv.
- Lyrics (2M examples, 60M words): Song lyrics from lyrics.com.
- Stories (100K examples, 5M words): Short stories that contain a title and five sentences from the ROC stories dataset (Mostafazadeh et al., 2016).

Due to computation and time limits, only one-tenth of the ROC stories dataset is processed for the retrained model with custom mask.

Mask creation: The mask function takes a document as input, tokenizes it, and performs part-of-speech tagging. It then iterates through the tokens, checking if they are common nouns ('NN' for singular and 'NNS' for plural) based on NLTK's part-of-speech tags. If a token is a common noun and a randomly generated number is less than the masking probability (p), it adds the span's information (type, starting offset, length) to the list of masked spans.

Framework: After creating the mask function and manufacturing the infilling examples following Figure 5, a language model is chosen for fine-tuning. In this implementation, GPT-2 is used for training.

Code changes: The main implementation code resides in the `project.ipynb` file. It completes various tasks, including dataset processing, model training, and demonstrating infilling examples using both the retrained model on custom mask (RM) and the original trained model provided in the paper (TM). Significant changes were introduced to the `ilm/ilm/mask/custom_mask.py` file to create the custom mask function for common nouns. To address a TypeError in `ilm/trained_ilm.py`, modifications were made to how the model's logits were obtained. To reduce processing time, an argument was added to `ilm/create_ilm_examples.py` to enable the processing of a specified fraction of the dataset. More detailed information can be found in the README file.

Libraries: In the mask function, the Natural Language Toolkit (NLTK) is used. It is an important library used for part-of-speech tagging and tokenization.

6.4. Results and interpretation

Example outputs of the retrained model (RM) with a custom mask are presented in Table 6. In contexts a and b we can see that the last word "classmates" and "co-workers" will affect the choice of "school" or "work". This demonstrates the trait for ILM models in which it considers both preceding and subsequent text. In context c, the outputs demonstrate that the RM model, while consistent, lacks the variety seen in the TM model. While the TM model produces a more diverse set of words, the RM model tends to stick to a limited set of words, potentially due to its training on a smaller dataset.

Despite acknowledging computational limits and reduced training time compared to the original paper, the RM model outputs meaningful infilling words. The results indicate that sentences generated by RM are generally grammatically correct, semantically coherent, and make sense in context. It also validates the fact that ILM models both preceding and following content. However, comparing the RM and TM models becomes challenging as they were not trained on identical datasets or for the same duration. Therefore, while the RM model showcases good performance, it cannot be compared with the original model.

7. Conclusion and Discussion

Throughout the project, insights into language model infilling techniques were gained, particularly regarding the influence of custom masks on model behavior. Further exploration and experiments could focus on optimizing custom mask functions for diverse datasets, addressing computation limits, and enhancing model generalization. The findings contribute to the broader understanding of language model capabilities and customization for specific tasks. The evaluation metrics used primarily focus on grammatical correctness and semantic coherence, leaving room for a more comprehensive evaluation framework or automatic evaluation methods.

References

Berglund, M., Raiko, T., Honkala, M., Kärkkäinen, L., Vetek, A., and Karhunen, J. Bidirectional recurrent neural networks as generative models - reconstructing gaps in time series, 2015.

Brown, T. B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., Agarwal, S., Herbert-Voss, A., Krueger, G., Henighan, T., Child, R., Ramesh, A., Ziegler, D. M., Wu, J., Winter, C., Hesse, C., Chen, M., Sigler, E., Litwin, M., Gray, S., Chess, B., Clark, J., Berner, C., McCandlish, S., Radford, A., Sutskever, I., and Amodei, D. Language models are few-shot learners, 2020.

Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. BERT: Pre-training of deep bidirectional transformers for language understanding. In Burstein, J., Doran, C., and Solorio, T. (eds.), *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pp. 4171–4186, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics. doi: 10.18653/v1/N19-1423. URL https://aclanthology.org/N19-1423.

Donahue, C., Lee, M., and Liang, P. Enabling language models to fill in the blanks. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, 2020.

Fedus, W., Goodfellow, I., and Dai, A. M. Maskgan: Better text generation via filling in the_____, 2018.

Ippolito, D., Grangier, D., Callison-Burch, C., and Eck, D. Unsupervised hierarchical story infilling. In *Proceedings of the First Workshop on Narrative Understanding*, pp. 37–43, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics. doi: 10.18653/v1/W19-2405. URL https://aclanthology.org/W19-2405.

Joshi, M., Chen, D., Liu, Y., Weld, D. S., Zettlemoyer, L., and Levy, O. SpanBERT: Improving pre-training by representing and predicting spans. *Transactions of the Association for Computational Linguistics*, 8:64–77, 2020. doi: 10.1162/tacl_a_00300. URL https://aclanthology.org/2020.tacl-1.5.

Liu, D., Fu, J., Liu, P., and Lv, J. TIGS: An inference algorithm for text infilling with gradient search. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pp. 4146–4156, Florence, Italy, July 2019. Association for Computational Linguistics. doi: 10.18653/v1/P19-1406. URL https://aclanthology.org/P19-1406.

Mostafazadeh, N., Chambers, N., He, X., Parikh, D., Batra, D., Vanderwende, L., Kohli, P., and Allen, J. A corpus and cloze evaluation for deeper understanding of commonsense stories. In Knight, K., Nenkova, A., and Rambow, O. (eds.), *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp. 839–849, San Diego, California, June 2016. Association for Computational Linguistics. doi: 10.18653/v1/N16-1098. URL https://aclanthology.org/N16-1098.

Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., Sutskever, I., et al. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019.

Sun, Q., Lee, S., and Batra, D. Bidirectional beam search: Forward-backward inference in neural sequence models for fill-in-the-blank image captioning, 2017.

Zhu, W., Hu, Z., and Xing, E. Text infilling, 2019.