



# Efficient Training of Language Models to Fill in the Middle

220803  
윤주성



## New GPT-3 Capabilities: Edit & Insert

We've released new versions of GPT-3 and Codex which can edit or insert content into existing text, rather than just completing existing text. These new capabilities make it practical to use the OpenAI API to revise existing content, such as rewriting a paragraph of text or refactoring code. This unlocks new use cases and improves existing ones; for example, insertion is already being piloted in GitHub Copilot with promising early results.


- ▶ READ EDIT DOCUMENTATION
- ▶ READ INSERT DOCUMENTATION
- ▶ TRY IN PLAYGROUND

```
def fib(n):  
    if n <= 1:  
        return 1  
    return fib(n-1) + fib(n-2)  
  
fib(10)
```

# Authors

Mohammad Bavarian \* Heewoo Jun\*  
Nikolas Tezak John Schulman Christine  
McLeavey Jerry Tworek Mark Chen  
OpenAI

7월 28일자 신작



**Mohammad Bavarian** 📧 📧 📧

OpenAI  
openai.com의 이메일 확인됨 - [호환이치](#)  
Machine Learning and Distr...

---

제목	인용	연도
Evaluating large language models trained on code M Chen, J Tworek, H Jun, Q Yuan, HPO Pinto, J Kaplan, H Edwards, ... arXiv preprint arXiv:2107.03374	166	2021

**Heewoo Jun**  
OpenAI

---

**Names**

Heewoo Jun (Preferred)

---

**Emails**

\*\*\*@gpm.me

---

**Personal Links**

DBLP Semantic Scholar

---

**Education & Career History**

Researcher	OpenAI (openai.com)	2019 - Present
Researcher	Baidu (baidu.com)	2015 - 2019
MS student	Stanford University (stanford.edu)	2013 - 2015
Undergrad student	University of Toronto (toronto.edu)	2008 - 2013

# TABLE OF CONTENTS



01	Introduction	03	FIM training & inference
02	Evaluation	04	Pretraining results
05	Finetuning results	06	Discussion
07	Related work	08	Conclusion



# Overview & Abstract

- free-form generation을 위한 AR모델의 infilling 도전 논문
- Input **data transformation** 논문
- 문서를 3등분해서 middle section을 끝으로 보내서 학습 (FIM)
- **<PRE> Enc(prefix) <SUF> Enc(suffix) <MID> Enc(middle)**
- FIM을 적용해도 AR loss에 영향을 주지 않는 것 확인  
(FIM-for-free-property 강조)
- **FIM Rate**는 문서를 FIM 스타일로 바꿀지 결정하는 확률 값
- 미래의 AR LM은 FIM을 **default**로 학습하길 제안

```
/**
 * Recursive Fibonacci function with memoization.
 * @param {number} n
 * @returns {number}
 */
function fibonacci(n) {
  var memo = {};
  return (function fib(n, memo) {
    return n in memo ? memo[n] : (memo[n] = n <= 1 ? 1 : fib(n-1, memo) +
    fib(n-2, memo));
  })(n, memo);
}
```





# 01

## INTRODUCTION

모델 구조에 따라서  
infilling 능력이 제한되는 것 극복해보자





# Introduction

## 모델 구조에 따른 infilling 능력제한

- Left-to-right 모델은 prefix에 의존
- Encoder-only, encoder-decode는 suffix도 볼 수 있지만, training시 infill regions의 길이가 짧음



## Application에서는 앞 뒤 context 모두 활용해야

- Coding assistant
  - Docstring generation
  - Import statement generation
  - Completing a partially written function





# Introduction

## Our goal in this work

to address this limitation by **adding fill-in-the-middle (FIM)** capability to causal decoder-based language models

## Points

- 간단한 학습데이터 변형만으로 AR 모델이 infilling 배우면서 left-to-right capability 유지
- document  $\rightarrow$  (prefix, middle, suffix)  $\rightarrow$  (prefix, suffix, middle)
  - Random하게 3등분
    - Token-level
    - Char-level (더 좋음)

**(PRE) prefix (SUF) suffix (MID) middle**

- **FIM model** trained jointly on a Mixture of FIM & ordinary left-to-right data **가 성능 좋음**



# Introduction

## Contributions

- **FIM-for-free property**  
(기존 AR loss에 영향 안 줌)
- **Best practices for FIM in pretraining**  
(FIM rate 찾아냄)
- **Finetuning inefficiency**  
(finetuning으로 하기엔 pretraining만큼 필요함)
- **New infilling benchmarks**  
(random span infilling and random span infilling light 개발)
- **Need for sampling evaluations**  
(FIM test loss가 안 떨어져도 sampling based benchmark에서 차이남 (생성기반))



# Introduction

## FIM-for-free property

- 같은 데이터로 FIM 적용 유무 결과
- left-to-right loss가 동일함
- 50% time만으로도 left-to-right loss 동일 + New capa 배움
- FIM loss는 FIM rate 0.5가 당연히 더 낮음

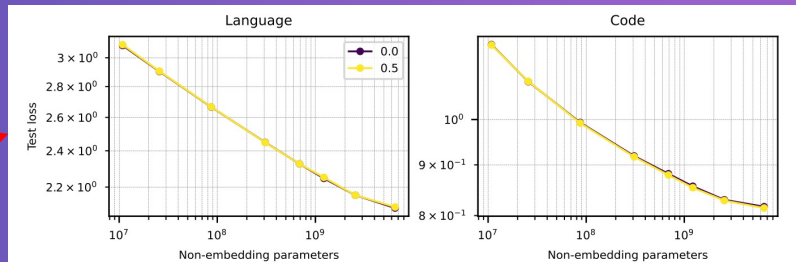


Figure 1: FIM can be learned for free. We pretrain language models with 50% and 0% FIM rates on two domains, natural language and code, and evaluate the test loss of all the final snapshots. All models are trained on 100B tokens of data. We observe that joint FIM training incurs no cost as the original left-to-right loss trend remains the same even though FIM models see the original data only 50% of the time and the models are learning a new capability. See Figure 3 for more evidence for the FIM-for-free property.

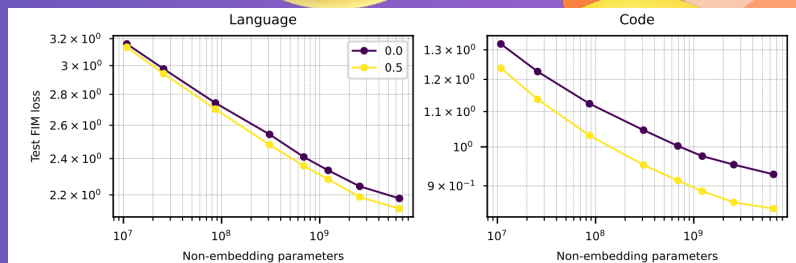


Figure 2: Evaluation of infilling capabilities of the same model scans from Figure 1 using FIM test losses. Models trained with FIM (yellow) obtain lower FIM test loss than baseline (purple) AR models. This shows that the FIM models are indeed learning to condition on the suffix while predicting the middle section allowing them to achieve lower test loss on FIM test set. Figures 1 and 2 together indicate that FIM models can be considered strictly better than AR models as they achieve the same left-to-right autoregressive loss but lower FIM loss.



2

Evaluation



# Evaluation

## AR loss



Cross entropy  
on normal  
left-to-right data

## FIM loss



100% FIM  
transformed  
data

## Sample-based benchmark



use nucleus  
sampling (0.95)





# Evaluation



## Autoregressive evaluation

- 대부분 benchmark는 few-shot prompting으로 평가 (DROP, QuAC 제외)
- Code는 HumanEval pass rate사용

## Infilling evaluation

- middle span token loss 계산
  - P (middle | prefix, suffix) for FIM models and P(middle | prefix) for AR models
- generative infilling capabilities 측정
  - code는 open ended generation이어도 정답체크 가능해서 code쪽에 초점 맞춤

# Evaluation



## Infilling evaluation

- middle span token loss 계산

$P(\text{middle} \mid \text{prefix}, \text{suffix})$  for FIM models and  $P(\text{middle} \mid \text{prefix})$  for AR models

- generative infilling capabilities 측정

code는 open ended generation이어도 정답체크 가능해서 code쪽에 초점 맞춤

- sampling based infilling benchmarks

single-line, multi-line, random span infilling (이건 새로만든 것)

- FIM → PSM, SPM

```
def unique(l: list):  
    """Return sorted unique elements in a list  
>>> unique([5, 3, 5, 2, 3, 3, 9, 0, 123])  
[0, 2, 3, 5, 9, 123]  
    """  
    return sorted(list(set(l)))
```



3

FIM training and inference



# FIM training and inference

## Training & inference format

### ● Document level

- split prior to tokenization, when the document is still a sequence of characters.  
(토큰화하기 전에 3등분한다)
- FIM이든 AROI든 문서사이는 <EOT> 토큰 추가 후 붙여줌 (<EOT> 토큰은 signal로 사용되서 학습해야)

$\langle \text{PRE} \rangle \circ \text{Enc}(\text{prefix}) \circ \langle \text{SUF} \rangle \circ \text{Enc}(\text{suffix}) \circ \langle \text{MID} \rangle \circ \text{Enc}(\text{middle}),$  (PSM)

where  $\circ$  denotes concatenation. The different documents, whether FIM or AR, then are concatenated with  $\langle \text{EOT} \rangle$  and are given to the model during training. We reiterate that we keep the loss on all three sections prefix, middle, and suffix, so FIM training does not cause a decrease in the autoregressive learning signal. Preliminary experiments, although not reported here, suggest that this choice is crucial for the FIM-for-free property to hold. This property does not change whether the sentinels are masked or not; however, it is important to always train on the  $\langle \text{EOT} \rangle$  tokens as it signals a successful join to the suffix.

For inference, we encode the given prefix and suffix and prompt the model with

$\langle \text{PRE} \rangle \circ \text{Enc}(\text{prefix}) \circ \langle \text{SUF} \rangle \circ \text{Enc}(\text{suffix}) \circ \langle \text{MID} \rangle.$ <sup>3</sup> (PSM inference)

We continue sampling from the model until it generates the  $\langle \text{EOT} \rangle$  token which is how the model communicates it has connected the prefix and the suffix.

If the model fails to generate an  $\langle \text{EOT} \rangle$  token within a reasonable allotted inference token budget, it is often a sign the model is having a difficult time connecting the prefix and the suffix, and the resulting samples often will be of worse quality, which motivates the procedure of EOT aware best-of-n sampling. See Appendix H for more discussion.

# FIM training and inference

## SPM mode

- variant of PSM

- SPM improved key-value caching during inference (?)
- appending tokens to the prefix no longer invalidates the keys and values computed in the suffix section (?)
- apply the FIM transformation with 50% probability in PSM mode and with 50% probability in SPM mode, so the model is able to handle both types of formatting in inference

Train distribution	FIM rate	Single-line		Multi-line		Random span	
		PSM	SPM	PSM	SPM	PSM	SPM
Joint	0.5	0.550	0.595	0.265	0.293	0.367	0.379
Joint	0.9	0.616	0.622	0.290	0.305	0.397	0.420
PSM	0.9	0.583	0.625	0.273	0.305	0.362	0.274
SPM	0.9	0.023	0.586	0.008	0.301	0.007	0.386

Table 1: Comparison of FIM performance when trained and evaluated in various SPM, SPM settings. All the joint runs put 50% of the total FIM rate on PSM and 50% on SPM. All results are obtained with temperature 0.2 and 100 samples per task.

# FIM training and inference

## SPM mode

### D Details of SPM encoding

As mentioned in Section 3, in SPM we use the ordering [suffix, prefix, middle]. In this section, we briefly discuss the choices regarding the sentinel tokens in SPM mode. A natural choice of encoding for SPM data would be to use

$\langle \text{SUF} \rangle \circ \text{Enc}(\text{suffix}) \circ \langle \text{PRE} \rangle \circ \text{Enc}(\text{prefix}) \circ \langle \text{MID} \rangle \circ \text{Enc}(\text{middle}) \circ \langle \text{EOT} \rangle$ . (SPM variant 1)

However, the encoding of SPM we use in this work is

$\langle \text{PRE} \rangle \circ \langle \text{SUF} \rangle \circ \text{Enc}(\text{suffix}) \circ \langle \text{MID} \rangle \circ \text{Enc}(\text{prefix}) \circ \text{Enc}(\text{middle}) \circ \langle \text{EOT} \rangle$ . (SPM variant 2)

The reason that we do not use the former is that it creates a separation between PSM and SPM, which may result to less transfer between SPM and PSM. To understand, note that with the second variant SPM data occurs naturally as part of PSM training since when we split a document uniformly at random, sometimes the chosen prefix will be empty. This is the reason pure PSM runs achieve strong performance when evaluated in SPM mode as in Table 1.

Despite this, we note that the first SPM variant has its own advantages. In particular, it can be stronger in handling of subtokens at the end of prefix. Hence, the choice of which variant of SPM to use may depend on application in mind. As such, especially when training in pure SPM mode, it could be preferable to use the former simpler form. However, in this work, due to our emphasis on joint training of PSM and SPM and to maximize transfer between them, we opt for the second variant.

# FIM training and inference

## Training & inference format

### ● Context level

- 언어모델 학습시 model context length에 맞게 boundary token을 활용해서 chunking하는 과정에서 prefix or suffix가 잘려나갈 수 있음 (fragmented FIM data)
- Chunking step 이후에 FIM 적용하는게 context level FIM (성능 더 좋음)
- <EOT>로 잘라서 FIM하고 다시 <EOT>로 붙이기

```
def token_level_psm_fim(document: str, vocab: Vocab) -> List[int]:
    tokens = vocab.encode(document)
    prefix, middle, suffix = randomly_split(tokens)
    return [
        vocab.sentinel("prefix"), *prefix,
        vocab.sentinel("suffix"), *suffix,
        vocab.sentinel("middle"), *middle,
    ]
def character_level_psm_fim(document: str, vocab: Vocab) -> List[int]:
    prefix, middle, suffix = randomly_split(document)
    return [
        vocab.sentinel("prefix"), *vocab.encode(prefix),
        vocab.sentinel("suffix"), *vocab.encode(suffix),
        vocab.sentinel("middle"), *vocab.encode(middle),
    ]
```





4

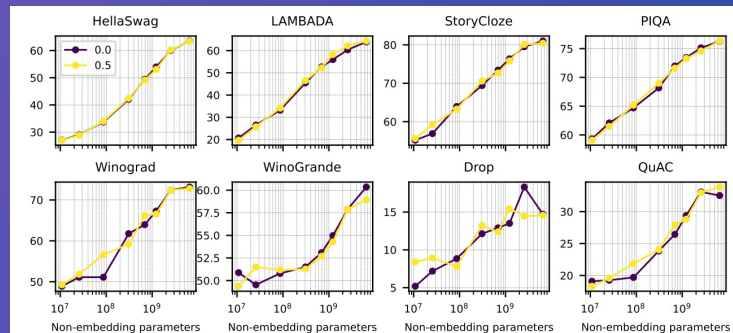
Pretraining results



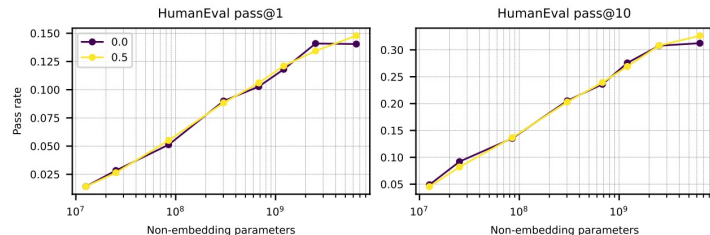
# Pretraining results

## Evaluation of left-to-right capabilities in downstream benchmarks

- train a series of models from 50M to 6.9B parameters from scratch with and without 50% FIM augmentation on natural language and code domains



(a) Comparison of natural language results. We report F1 for Drop and QuAC and accuracy for the rest.



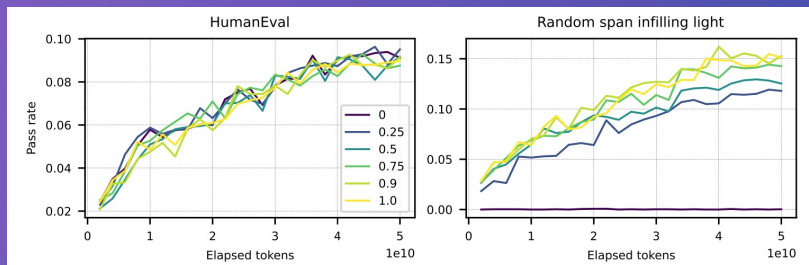
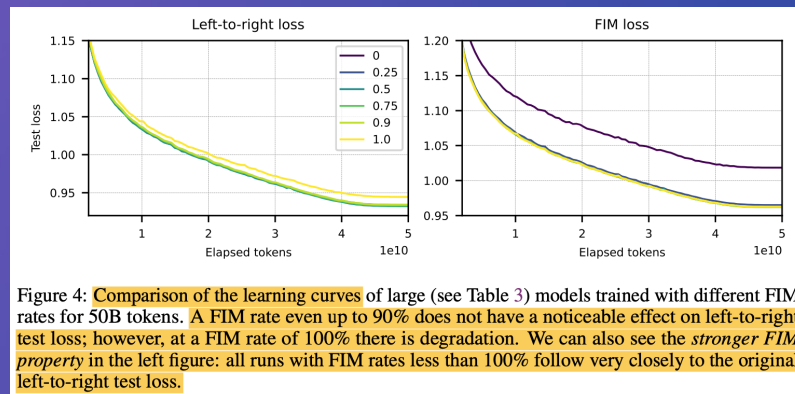
(b) Comparison of code results. We use temperature 0.8 and 400 samples per task for both pass@k.

Figure 3: Comparison of performance on standard benchmarks for the natural language (top) and code (bottom) domains. Joint training of next-token prediction and FIM allows the model to learn the new infilling task without affecting the original capabilities. This provides further evidence for FIM-for-free property.

# Pretraining results

## FIM rate

- FIM rate even up to 90% does not cause any degradation in left-to-right capabilities. However, there is a clear sign of degradation in ordinary AR test loss with 100% FIM rate
- FIM rate does significantly affect infilling capabilities. Even though the gain in FIM perplexity in Figure 4 due to a higher FIM rate is negligible



# Pretraining results

## SPM vs PSM vs joint SPM+PSM training

- in SPM, there is **no distinction between the prefix and the middle sections as they are one contiguous sequence of text**. This makes it more natural for the model to continue from the prefix in contrast to PSM where attention has to first identify where the span token is.

- Not only is joint pretraining the most efficient, but it also yields the most flexible model with two inference modes.

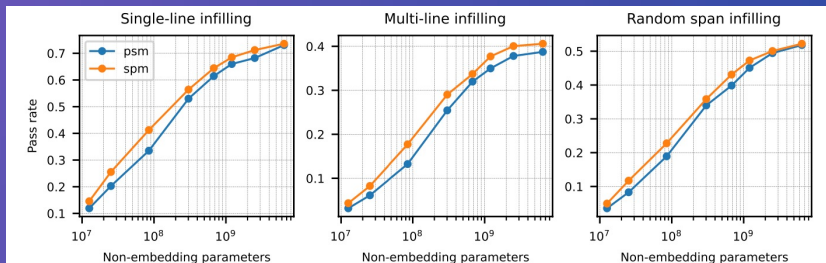


Figure 6: SPM mode shows a slight advantage in performance across scale. All the evaluations in this plot are at temperature 0.2 and 100 samples per task for single-line and multi-line infilling and 200 samples per task for random span infilling.

Train distribution	FIM rate	Single-line		Multi-line		Random span	
		PSM	SPM	PSM	SPM	PSM	SPM
Joint	0.5	0.550	0.595	0.265	0.293	0.367	0.379
Joint	0.9	0.616	0.622	0.290	0.305	0.397	0.420
PSM	0.9	0.583	0.625	0.273	0.305	0.362	0.274
SPM	0.9	0.023	0.586	0.008	0.301	0.007	0.386

Table 1: Comparison of FIM performance when trained and evaluated in various SPM, SPM settings. All the joint runs put 50% of the total FIM rate on PSM and 50% on SPM. All results are obtained with temperature 0.2 and 100 samples per task.

# Pretraining results

## Context-level vs document-level FIM

- **perplexity** evaluation **does not always capture** the gains in the sampling performance
- Figure 8 (left) shows that training on these invalid examples in **document-level FIM does not affect the left-to-right evaluation**. Hence practitioners might still sometimes prefer document-level FIM due to its simpler implementation.

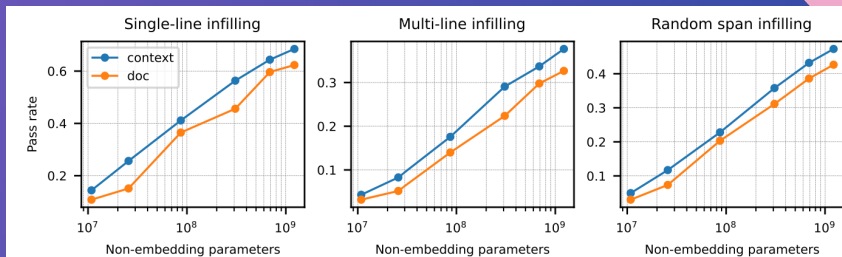


Figure 7: **Applying FIM at the context level consistently outperforms document level FIM.** All benchmarks are evaluated with temperature 0.2 and 200 samples/task.

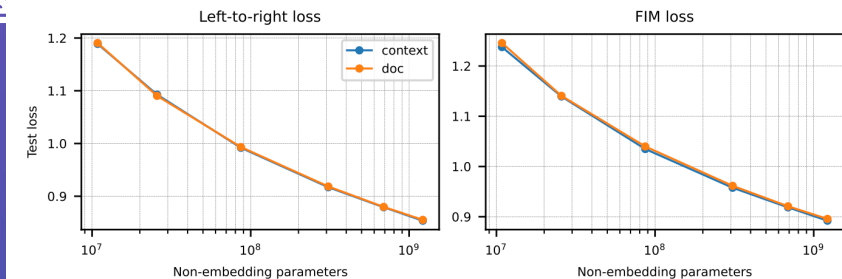


Figure 8: **Comparison of losses with different FIM implementations.** While document level FIM introduces partially broken data into training, it does not hurt the autoregressive loss (left). We also find that the reduction in FIM perplexity (right) is not commensurate to the gain in pass rate shown in Figure 7.

# Pretraining results

## Middle span selection

- important consideration in FIM training is the **choice of middle span**.
- Character level이 좋다 (no train-test mismatch)

Training middle span	Single-line infilling	Multi-line infilling	Random span infilling
Line-level random span	0.586	0.269	0.015
Token-level random span	0.548	0.242	0.102
<b>Character-level random span</b>	0.557	0.250	0.321

Table 2: Pass rates of medium models pretrained with **various middle span selection strategies**. Training on line-based spans improves the single- and multi-line infilling metrics reported in InCoder, but line- and token-level spans used in previous works can not robustly handle real life use cases where the span starts or ends in subtokens. Overall, character-level random span run dominates in random span benchmark while it is also not far behind in single and multi line infilling.





5

Finetuning results





# Finetuning results

Pretraining > finetuning

- finetuning도 50B tokens & 90% FIM로 해야 그나마 비슷해진다 (데이터의 반정도를 다시)

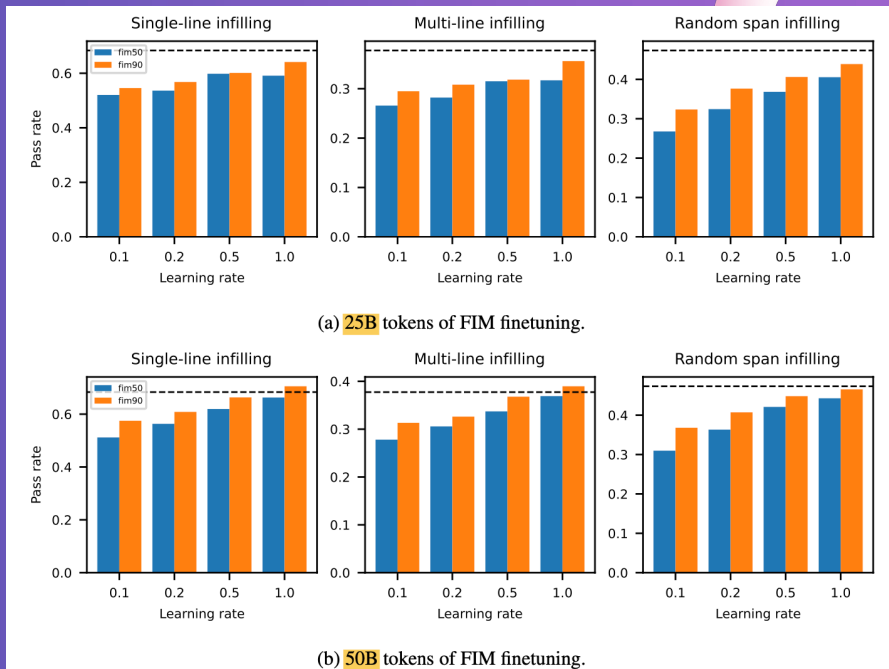


Figure 9: Evaluation of the final snapshots of models pretrained for 100B tokens without FIM and then finetuned for 25B (row a) and 50B (row b) tokens with FIM. The x-axis shows the learning rate multiplier relative to the pretraining learning rate. The dashed line indicates the baseline performance of the model pretrained for 100B tokens with a FIM rate of 50% with no additional finetuning. Only the most aggressive combination of 90% FIM rate and a learning rate multiplier of 1.0 with 50B tokens of finetuning catches up to the performance of the baseline. Reported results are with temperature 0.2 and 100 sampler per task.



6

Discussion

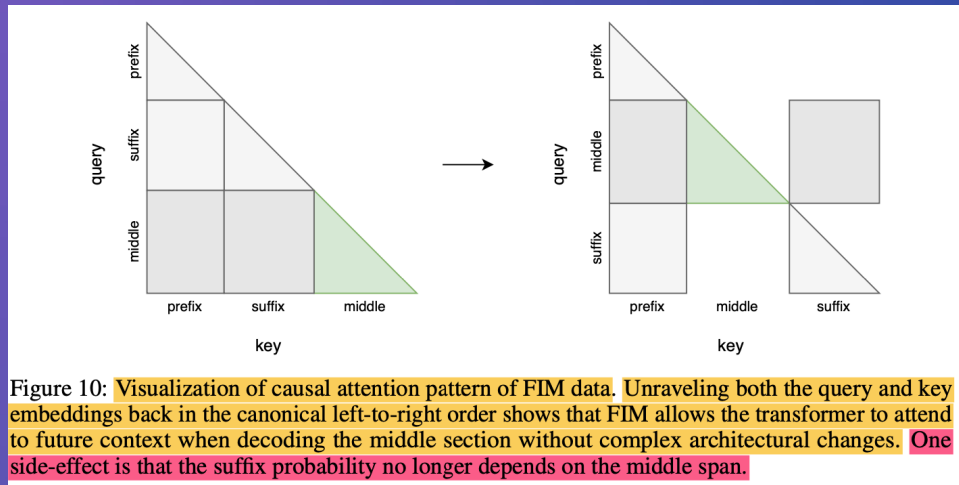


# Discussion



## Pretraining vs finetuning

- even though FIM data is locally identical to autoregressive data, FIM does impose a **different global attention pattern over the whole document**.
- show the causal attention mask of a FIM document in Figure 10. These new attention pattern could be the reason why it takes a relatively long token horizon and a high learning rate to learn FIM in finetuning



FIM과 기존 AR을 보면 attention pattern이 다르니 학습이 더 걸리는거 아닐까 라는 주장 부작용은 suffix가 middle을 보지 못한다 정도..?!

# Discussion



## FIM loss, AR loss, and the difficulty of FIM task

- FIM이 left-to-right generation보다 더 어려울 수 있다
- prefix뿐만 아니라 특정 suffix에 맞는 middle을 generation하는게 쉽지 않고 후처리하기도 어렵다 (문맥상 trim하면 어색함)

When I was young, I only liked to play video games. Over time, I started thinking if it'd be possible to make bots to play better than any human can ever play these games. I eventually decided I liked working on the latter more than playing the games themselves and that's how first I got interested in AI research.

When I was young, I only liked to play video games. I would play sometimes more than 13 hours per day. The rush, novelty, and variety were beyond anything real life could offer. I loved the challenge and I excelled at it. I would often skip classes and go to and that's how first I got interested in AI research.



# Discussion



## FIM loss, AR loss, and the difficulty of FIM task

### ● 성공케이스

#### H.1 Successful infilling examples

FIM enables a model to process information from both before and after the point of generation. This unlocks new capabilities that previously required specialized models finetuned on specific tasks. For example, unlike Codex [Chen et al., 2021] that trained a separate docstring model, we now have a single model that can infer the import modules, function names, arguments, docstrings, definitions, and many more. We show one such example below that is impossible to complete unless the model can read the entire source code. This example is also interesting in that the prefix “from sym” and the suffix both contain subtokens, which are known to cause traditional language models trained without techniques like stochastic BPE [Provilkov et al., 2019] to fail.

```
from sympy import isprime

def largest_prime_factor(n):
    """
    Return the largest prime factor of n.
    """
    ans = 1
    for num in range(2, n + 1):
        if n % num == 0 and isprime(num):
            ans = num
    return ans
```

The benefits are not limited to coding. The model can adapt to the existing writing style and complete the passage in a natural way that takes the ending into consideration.

Dolphins are very intelligent animals. They are mammals and breathe air. They live in the sea and are related to whales and porpoises. Dolphins are very playful animals.

The commercial diver finally thought he'd snagged a big catch when he saw something white. But then he quickly realized it wasn't a fish — he was wrangling an alligator.

Wikipedia is a free, web-based, collaborative, multilingual encyclopedia. It is overseen by the nonprofit Wikimedia Foundation. Wikipedia uses a collaborative software known as wiki that facilitates the creation and development of articles.



# Discussion



## FIM loss, AR loss, and the difficulty of FIM task

### ● 실패케이스

#### H.2 Limitations

**Difficult prompts.** Unlike completing text from the end, infilling needs to infer the missing span that connects the prefix to the suffix. When the suffix is completely unrelated, the model can generate very long middle sections. We consider this behavior as the model's attempt at coming up with a plausible trajectory that joins the ending. Because the context size is limited, the model usually fails to join. However, given that even people have trouble infilling some of these prompts in a short passage, this failure demonstrates how challenging of a task FIM can be.

Below, we show one such difficult prompt where the model typically fails to connect entirely or join in a seamless way. Even when the model writes a seemingly plausible middle section, the quality can often vary.

The dentist looked me in the eyes and said, "I'm going to have to take all of your teeth out."  
I was stunned. I said, "All my teeth? Isn't there something else we could do?" He said, "No  
, I'm afraid not."

No one can predict the future.

The Ottomans were defeated in World War I and the French were defeated at Waterloo.

**Deciding when to stop.** The model is trained to predict the <EOT> token when it thinks it has joined the suffix. Even when the prompts are seemingly straightforward, deciding when to end can still be a challenge in practice. Because there are many equally valid completions with varying lengths, the probability of outputting the <EOT> is discounted by other longer candidates and is often smaller than expected. This is further exacerbated by the fact that the terminal symbol can simply be missed due to sampling. This results in a behavior where the model does not seem to end in a timely manner and generates a valid, but spurious content in the middle. In the process, the model can choose to write its own ending to the prefix, effectively ignoring the given suffix.

Dogs are friendly animals.  
Koalas are pleasant animals.  
Monkeys are playful animals.  
Whales are enormous animals.  
Owls are wise animals.  
Penguins are graceful animals.  
Crocodiles are ferocious animals.

While the general problem of not knowing when to stop applies to normal left-to-right completion as well, this has not been as big a problem as infilling because there is no constraint to join the suffix.

**Repetition.** When the model fails to generate an <EOT> and copies the suffix, the model's ability to match patterns leads it to lock on and repeat the prompt indefinitely. Surprisingly, even large models are susceptible to this mode of failure. The example below ends with "the the heart," because the model has failed to generate the terminal symbol and is still in the middle of filling in the missing span which unfortunately will not stop.

The way is not in the sky. The way is in the heart.  
The way is not in the sky. The way is in the heart.  
The way is not in the sky. The way is in the heart.  
The way is not in the sky. The way is in the heart.





# Discussion



## FIM loss, AR loss, and the difficulty of FIM task

### ● 중간 케이스

#### - numbered items로 힌트주기

#### H.3 Mitigations

Like GPT-3 [Brown et al., 2020] where the performance depends on the quality of prompts, some of the failures in the earlier sections can be alleviated with prompt engineering. Namely, providing hints to constrain the output can dramatically improve the model's ability to generate the <EOT> token and connect to the suffix within a reasonable token budget as the model has a more concrete understanding of how long the middle section should be.

One such idea is to provide examples both in the beginning and the end with numbered items. This makes the model internally keep track of the position, pay attention to the desired prefix and suffix, and generally abstain from generating spurious content as shown below. Providing leading examples alone without any explicit cues can often worsen the problem because it does not resolve the ambiguity in whether the model should join to the beginning of the suffix or consider it as part of a new example.

1. Dogs are friendly animals.
2. Koalas are sleepy animals.
3. Lions are regal animals.

Section 1:

1. The way is not in the sky. The way is in the heart.
2. Peace comes from within. Do not seek it without.

Section 2:

It is important to note that the numbered few-shot prompting helps considerably but does not completely fix the problem, as the model can still accidentally start a new list of items.

In general, as the model can simply miss sampling the <EOT> token, we recommend generating multiple samples and preferring samples that end with <EOT>, as this increases the chance of choosing a sample that actually joins the ending. When multiple samples end in <EOT>, they can be reranked by the likelihood or other heuristics of interest. We call this EOT-aware best-of-n sampling.





# Discussion



## FIM loss, AR loss, and the difficulty of FIM task

### ● PPL도 AR보다 FIM이 더 높다(어렵)

- model은 모두 FIM이고 test loss만 변경
- FIM과 left-to-right 섞어서 학습했지만 left-to-right를 더 잘하고 FIM을 더 못함 (전체적 섹션에 대해서)
- middle section은 FIM이 더 잘함

결과 해석이 좀 애매한 느낌도..



The difficulty of FIM task compared to AR task is also reflected in the loss associated with each task. To see this, in Figure 11, we compare the FIM loss with the AR loss over a suite of FIM models all with 50% FIM rate. To remove confounders, we ensure the documents that underlie the AR test set are the same documents that are transformed through FIM to make up the FIM test set. We find the FIM perplexity is consistently higher than the AR perplexity across scale. That is, on average

$$P_{\text{FIM}}([\text{prefix}, \text{suffix}, \text{middle}]) > P_{\text{AR}}([\text{prefix}, \text{middle}, \text{suffix}]),$$

which means the models have a harder time modelling the same document in FIM format than AR format.

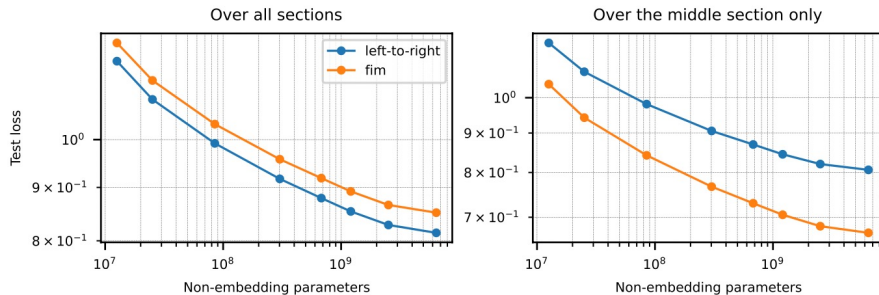


Figure 11: Comparison of the overall (left) and middle span (right) loss of 50% FIM code models. In the left plot, we see that the AR loss is consistently lower than the FIM loss suggesting that next-token prediction is inherently more compressible than infilling in the middle. The right figure evaluates the conditional loss of the middle span given the surrounding context showing that  $P_{\text{FIM}}(\text{middle}|\text{prefix}, \text{suffix}) > P_{\text{AR}}(\text{middle}|\text{prefix})$ . Here, FIM attains a lower loss because it can attend to the suffix. We emphasize that left-to-right and FIM here do not refer to model type, as all models in this figure are FIM models. They refer rather to the type of test loss used in evaluation.



7

Related work



# Related work



similar to this work,

- utilize left-to-right autoregressive modeling **by moving the infill regions** to the end of context, with regions separated by sentinels
  - GLM: General Language Model Pretraining with Autoregressive Blank Infilling
  - CM3: A CAUSAL MASKED MULTIMODAL MODEL OF THE INTERNET (Facebook AI Research)
  - InCoder: A Generative Model for Code Infilling and Synthesis





8

Conclusion



# Conclusion

- left-to-right, FIM 데이터 섞어서 causal decoder도 학습하면 document 중간 채울 수 있다
- FIM-for-free property를 발견했다 (pretrain시 데이터 변경만하면 FIM을 공짜로 할 수 있다)
- best FIM performance 위해서는 finetuning보단 pretraining 추천
- perplexity는 true infilling performance 반영하지 못한다
- FIM rate는 100% 이하로 하자 (50~90%)
- FIM은 character level로 하자

## Future directions

- Smarter span selection
- Steerable generation
- Further examination of the FIM-for-free property
- Multiple infilling slots
- Improving natural language FIM performance
- Role of bidirectionality and attention



Thank you