ExLM: Rethinking the Impact of `[MASK]` Tokens in Masked Language Models

Kangjie Zheng [1]   **Junwei Yang** [1]   **Siyue Liang** [1]   **Bin Feng** [2 1]   **Zequn Liu** [1]
Wei Ju [3]   **Zhiping Xiao** [# 4]   **Ming Zhang** [# 1]

Abstract

Masked Language Models (MLMs) have achieved remarkable success in many self-supervised representation learning tasks. MLMs are trained by randomly masking portions of the input sequences with `[MASK]` tokens and learning to reconstruct the original content based on the remaining context. This paper explores the impact of `[MASK]` tokens on MLMs. Analytical studies show that masking tokens can introduce the ***corrupted semantics*** problem, wherein the corrupted context may convey multiple, ambiguous meanings. This problem is also a key factor affecting the performance of MLMs on downstream tasks. Based on these findings, we propose a novel enhanced-context MLM, ExLM. Our approach expands `[MASK]` tokens in the input context and models the dependencies between these expanded states. This enhancement increases context capacity and enables the model to capture richer semantic information, effectively mitigating the corrupted semantics problem during pre-training. Experimental results demonstrate that ExLM achieves significant performance improvements in both text modeling and SMILES modeling tasks. Further analysis confirms that ExLM enriches semantic representations through context enhancement, and effectively reduces the semantic multimodality commonly observed in MLMs.
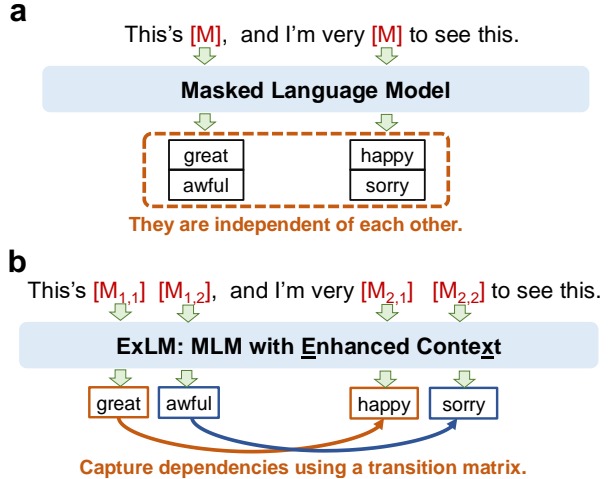
---

[1]State Key Laboratory for Multimedia Information Processing, School of Computer Science, PKU-Anker LLM Lab, Peking University, China. [2]International Digital Economy Academy (IDEA), Shenzhen, China. [3]College of Computer Science, Sichuan University, Chengdu, China. [4]Paul G. Allen School of Computer Science and Engineering, University of Washington, U.S.A. Correspondence to: Ming Zhang <mzhang_cs@pku.edu.cn>, Zhiping Xiao <patxiao@uw.edu>.

Figure 1. Illustrations of the vanilla MLM (**a**) and ExLM (**b**). MLM can be affected by the multimodality problem. In ExLM, the model creates multiple hidden states for each `[MASK]` token (e.g., $[M_{1,1}]$, $[M_{1,2}]$, $[M_{2,1}]$, $[M_{2,2}]$). By leveraging a larger semantic space and explicitly modeling the dependencies between these states, the model can capture richer semantic information in the enhanced context while mitigating the effects of multimodality.

1. Introduction

Pre-trained masked language models (MLMs) have achieved significant success across various types of sequence data, including text (Devlin, 2018; Liu, 2019; Lan, 2019; He et al., 2020; Joshi et al., 2020; Meng et al., 2023), small molecules (Wang et al., 2019; Ross et al., 2022; Pan, 2023; Zheng et al., 2024a), and proteins (Lin et al., 2022; 2023; Su et al., 2023; Zheng et al., 2024b; Hayes et al., 2025), establishing themselves as a foundational approach for sequence representation learning tasks. To enable the effective extraction of useful semantic information, MLM employs a mask-then-predict training strategy. During the pre-training process, a certain proportion (typically 15%) of tokens in the input sequence are randomly replaced with a special symbol, `[MASK]`. The model is then trained to predict the original tokens based on the corrupted context. This process allows MLMs to learn meaningful semantic information in a self-supervised manner from large-scale unlabeled data. The learned knowledge can be effectively transferred to a

wide range of downstream tasks, such as text classification and molecular property prediction, significantly improving the performance of deep learning models in these domains.

Although MLMs have achieved significant success in numerous tasks and applications, their effectiveness remains an important research problem. During the MLM training process, parts of the input are masked, and unreal tokens (`[MASK]`) are introduced into the context. This process impacts the input context of MLMs in two critical aspects:

- Introducing *Unreal Tokens*: The context provided to MLM during pre-training contains a large number of unreal tokens (`[MASK]`) that are absent from real-world text, potentially distorting the learning process.

- Resulting in *Corrupted Semantics*: The replacement of tokens with `[MASK]` results in incomplete context semantics, which can negatively affect the model's ability to learn accurate semantic representations.

These two aspects of impact are closely tied to the mask ratio used during pre-training. Higher mask ratios exacerbate the problems of unreal tokens and corrupted semantics at the same time, leading to a noticeable performance drop when the mask ratio is too high. While previous studies have investigated the unreal token problem and its impact on MLM performance (Clark, 2020; Meng et al., 2023), they have largely overlooked the corrupted semantics problem. Consequently, there is little work systematically exploring the impact of both problems on MLM performance, or evaluating the magnitude of their effect independently. This gap arises because these two factors are interdependent and both rely on the mask ratio, making it challenging to design experiments that disentangle their respective effects.

To fill this research gap, we designed and conducted the *Repeated MLM* analytical experiments (shown in Figure 2) to separately evaluate the relative impact of these two factors. The experimental results demonstrate that **the corrupted semantics problem has a significantly greater impact on MLM performance than the unreal tokens problem**. This is further reflected in a stronger semantic multimodality (Gu et al., 2017), where multiple plausible predictions exist for the original tokens due to the ambiguous or context-dependent meanings. Based on these findings, we propose a novel pre-trained model, ExLM, which enhances context representation in MLMs. By expanding each `[MASK]` token in the input context into multiple hidden states, the model is provided with a larger semantic space. This allows it to capture richer semantic information associated with each `[MASK]` token, thereby reducing the semantic multimodality in token prediction. Furthermore, we introduce a transition matrix between these expanded states, enabling the model to directly capture semantic relationships among
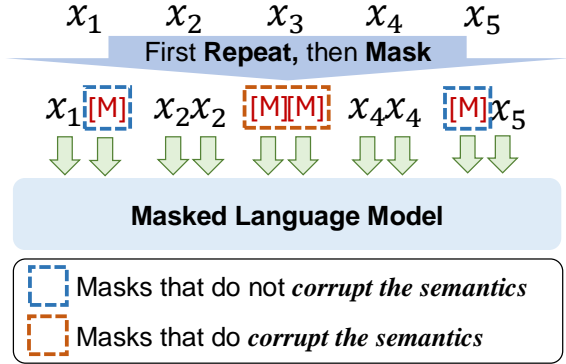

Figure 2. Illustrations of the *Repeated MLM* experiment. Each token in the model input is repeated before being masked. The artificial redundancy introduced into the input ensures that replacing a token with `[MASK]` (`[M]`) does not necessarily lead to semantic corruption in the context, as the repeated tokens provide additional information to preserve the original semantics. A token is regarded as having corrupted semantics only when all its copies are masked.

different states. To efficiently train the ExLM, we propose a state alignment algorithm based on dynamic programming, which aligns target tokens with expanded states in a data-driven manner, significantly improving model performance.

In summary, the contributions of this work are as follows:

- We conduct the first systematic analysis of MLM behavior from the dual perspectives of both *unreal tokens* and *corrupted semantics*. Through a series of carefully designed experiments, we reveal that the corrupted semantics problem has a greater impact on MLM performance, providing new insights into MLM studies.

- Based on the analysis, we propose ExLM, a novel pre-trained model that enhances context representation for MLMs. By expanding `[MASK]` tokens into multiple states and employing a state alignment algorithm, ExLM effectively improves semantic modeling and reducing the semantic multimodality in the context.

- Extensive experimental results on text and molecular property prediction tasks demonstrate the superior performance of ExLM. Further analysis confirms its effectiveness and highlights its potential for addressing challenges in MLM pre-training.

2. Background of MLM

The pre-training objective of MLMs is to predict missing tokens from a partially masked input sequence. In this process, a portion of the input tokens is randomly selected and masked (i.e., replaced with `[MASK]`), forming a corrupted sequence. The model's task is to recover the original tokens using the remaining unmasked context as input.

k \ P	2.25%	15%	38.7%	62.2%	78.9%
1	81.3	83.6	83.3	81.3	79.6
2		81.4	83.1	83.3	81.3
4			81.0	83.2	82.6
8				79.2	82.8

Same mask ratio

Figure 3. Results of the *Repeated MLM* experiment. These are the evaluation results (i.e., accuracy) of MLMs with different repetition times k and mask ratios p on the MNLI task (Williams et al., 2018), with results of similar performance highlighted in similar colors.

k \ P	2.25%	15%	38.7%	62.2%	78.9%
1	2.25%	15%	38.7%	62.2%	78.9%
2		2.25%	15%	38.7%	62.2%
4			2.25%	15%	38.7%
8				2.25%	15%

Figure 4. The corrupted semantics proportions corresponding to each set of *Repeated MLM* experiments are reported. When the repetition times are k and the mask ratio is p, the proportion of corrupted semantics is p^k (see Appendix C for a detailed proof).

Specifically, let's define a sequence of tokens as $\mathbf{X} = [x_1, x_2, \ldots, x_n]$. In the MLM pre-training process, some of the tokens in $\mathbf{X}$ are randomly replaced with a special token `[MASK]`, producing a partially masked input sequence $\tilde{\mathbf{X}} = [\tilde{x}_1, \tilde{x}_2, \ldots, \tilde{x}_n]$, where:

$$\tilde{x}_i = \begin{cases} [\text{MASK}] & \text{if token } x_i \text{ is selected to be masked}, \\ x_i & \text{otherwise}. \end{cases}$$

The model is trained to predict the original tokens x_i corresponding to the masked positions i in $\tilde{\mathbf{X}}$. Typically, a fixed percentage of the input tokens (e.g., 15%) are randomly selected to be masked during the pre-training process. The objective of MLM pre-training is to minimize the discrepancy between the predicted tokens and the true tokens at the masked positions. This is achieved by maximizing the likelihood of the true tokens given the context of the unmasked tokens. The objective function for MLM pre-training can be formulated as:

$$\mathcal{L}_{\text{MLM}} = - \sum_{i \in M} \log P(x_i \mid \tilde{x}_1, \ldots, \tilde{x}_n),$$

where M denotes the set of indices corresponding to the masked tokens, and $P(x_i \mid \tilde{x}_1, \ldots, \tilde{x}_n)$ represents the model's predicted probability for a certain token x_i at the masked position i, conditioned on a corrupted context formed by the unmasked tokens in $\tilde{\mathbf{X}}$.

3. Understanding the Impact of `[MASK]`

In this section, we design analytical experiments to explore the impact of **unreal tokens** and **corrupted semantics** on MLM performance. However, since these two factors are strongly interrelated in MLMs, we first designed a *Repeated MLM* experiment to decouple unreal tokens and corrupted semantics (Section 3.1). This allows us to separately explore the impact of each factor and demonstrate that corrupted

semantics has a significantly greater effect on MLM performance (Section 3.2). Additionally, our analysis further shows that more severe corrupted semantics lead to the loss of critical context semantics, which exacerbates the multi-modality in token predictions (Section 3.3).

3.1. Decoupling *Corrupted Semantics* and *Unreal Tokens*

In traditional MLM training, the mask ratio plays a crucial role in determining both the proportion of `[MASK]` tokens introduced into the context and the proportion of semantic information discarded from the original context. This interplay makes it challenging to isolate and compare the impact of these two factors–the proportion of `[MASK]` tokens (i.e., **unreal tokens**) and the proportion of **corrupted semantics**–on MLM performance. To address this challenge, we designed the *Repeated MLM* experiment.

The **core idea** of this experiment is to artificially introduce redundancy into the MLM's input, ensuring that replacing a token with `[MASK]` does not inevitably lead to semantic corruption in the context.

Specifically, as illustrated in Figure 2, before feeding the sequence into the MLM, we first repeat each token k times, where $k \in \mathbb{N}$ is a hyperparameter set based on experimental requirements. We then randomly mask the repeated sequence at a certain ratio p, which means replacing $p \in (0.0\%, 100.0\%)$ of the tokens with `[MASK]`. This setup creates an interesting phenomenon: while the proportion of unreal `[MASK]` tokens in the context remains p due to the masking ratio, the proportion of corrupted semantics changes. Since each token has k copies and the probability of each copy being masked is p, the probability of the semantic information carried by a token being completely corrupted becomes p^k. We also have provided a detailed proof about the results in Appendix C.

By keeping p fixed and varying k, we ensure that the pro-

portion of `[MASK]` tokens in the context remains constant while adjusting the degree of corrupted semantics. This enables us to control variables effectively, and systematically compare the impact of these two factors on MLM performance by measuring each factor separately.

3.2. What Matters More: *Corrupted Semantics* or *Unreal Tokens*

Under the experimental setup designed in Section 3.1, we have trained a series of MLMs with different repetition times k and mask ratios p to analyze MLM's behavior. For consistency, all training hyperparameters of the MLMs in the experiments are kept the same except for p and k. Additionally, during downstream fine-tuning, the input is repeated with the same repetition times k as in pre-training. More detailed training configurations and hyperparameters can be found in Appendix B. The results of the MNLI task, evaluated using accuracy as the primary metric (Williams et al., 2018), are presented in Figure 3. We also provide the results of this experiments on more tasks in Appendix F. Some entries are blank (gray areas) due to overly low proportions of corrupted semantics (less than 0.05%), causing very low loss during pre-training and unstable model training problem. For comparison, the proportions of corrupted semantics for each experiment are provided in Figure 4.

From these results, we can observe that both excessively large and excessively small corrupted semantics lead to significant performance degradation in the model. Besides, **when the proportions of corrupted semantics remain constant and the mask ratio varies, the performance of the MLM changes only slightly**. Although there is a minor decline in performance as the mask ratio increases (red cells in Figure 3, from 83.6 to 82.8), the overall performance remains relatively similar. As long as the proportions of corrupted semantics are not excessively high, the model can still maintain relatively good performance even if the context in the MLM pre-training process contains a large number of `[MASK]` tokens (e.g., $p = 78.9\%$, $k = 8$).

In contrast, when the mask ratio remains fixed and the proportions of corrupted semantics increase, the model's performance exhibits more significant changes (from 82.8 to 79.6). This demonstrates that **the corrupted semantics problem has a more pronounced impact on performance compared to the unreal tokens problem**.

3.3. Core Impact of *Corrupted Semantics*: Multimodality

We conducted a deeper analysis to investigate how *corrupted semantics* influence the performance of MLMs. As illustrated in Figure 1(a), the reconstruction of a `[MASK]` token relies on the semantic information provided by its context. When a corrupted context is given, it may imply multiple different potential semantics, resulting in signifi-

k \ P	2.25%	15%	38.7%	62.2%	78.9%
1	1.62	2.78	3.95	5.79	7.26
2		1.37	2.30	3.44	5.48
4			1.23	1.77	3.35
8				1.22	1.76

Larger semantics corruption

Figure 5. Entropy analysis in the *Repeated MLM* experiment. We visualize the entropy (in bits) of different models during mask prediction. Larger semantics corruption significantly leads to an increase in the entropy of the model's prediction distribution.

cantly different reconstruction outcomes. This phenomenon is referred to as multimodality, a concept initially proposed in NAT (Gu et al., 2017). Multimodality causes MLMs to produce more mixed and uncertain predictions during pre-training, thereby significantly affecting downstream performance. To gain a more straightforward understanding of multimodality, we analyzed the prediction entropy of MLMs with different repetition times k and mask ratios p. We also have provided more details on the entropy calculation process in Appendix E. As illustrated in Figure 5, with an increase in semantic corruption, the prediction entropy also rises, indicating a more severe semantic multimodality. Such multimodal phenomenon also significantly impacts the performance of MLMs.

On the other hand, when corruption is too low (e.g., 2.25%), the context still contains plenty of semantic clues, making token prediction too easy. At this point, the model's predictions also exhibit very low uncertainty. The training curve analysis in Appendix G also shows that when the corruption is too low, the training task becomes overly simplistic. This simplicity prevents the model from learning deeper knowledge and ultimately reduces performance.

Therefore, an optimal approach would be to design the input context with a reasonable degree of semantic corruption, ensuring the model can correctly handle the missing semantics and learn meaningful knowledge. This balance would avoid the negative impacts of the semantic multimodality problem while maintaining the model's performance.

4. Proposed Method: ExLM

In this section, we will first introduce the core design concepts and overall architecture of ExLM (Section 4.1). Then, we will elaborate on two key components of context enhancement, **states expansion** and **dependency capture**, by detailing the model design and training algorithm for each
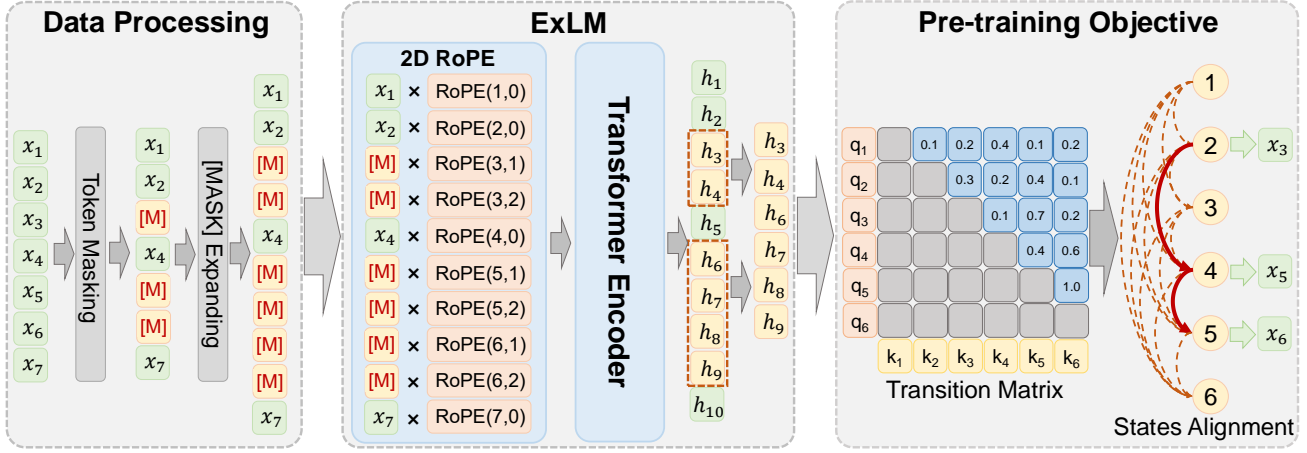
Figure 6. Overview of our proposed ExLM. ExLM creates multiple expanded states for each [MASK] token, providing a larger semantic space and a stronger ability to capture the missing semantics in the context. Additionally, it explicitly models the semantic dependencies between the expanded states using a transition matrix, which is computed based on the representation of each state $\mathbf{h}_i$.

(Section 4.2 and Section 4.3).

4.1. Overview

Through the analysis in Section 3, we have identified that corrupted semantics is the main factor affecting MLM's performance. When semantics of the context are severely corrupted, the resulting multimodality makes it increasingly difficult for the MLM to restore the original tokens.

Building on this, an intuitive approach to improving the MLM is: **how can we enhance the model's ability to better handle semantic multimodality?** More specifically, the semantic multimodality can be divided into two aspects:

- **Intra-token Multimodality**: The potential choices for each missing token become more diverse, and the significant semantic differences among these choices increase the semantic diversity and ambiguity.

- **Inter-token Multimodality**: The meaning of one token is intricately linked to and influenced by the meanings of other tokens, resulting in complex semantic interactions and dependencies between missing tokens.

Therefore, we need to enhance the model's capabilities in two areas: one is improving its ability to model diverse and ambiguous semantics, and the other is enhancing its ability to capture semantic dependencies between missing tokens. Following this core idea, we propose a novel enhanced-context MLM (ExLM) with two main improvements:

- **States Expansion**: For each [MASK] token in the input context, multiple hidden states are created, enlarging the semantic space for the model. A larger

semantic space enables the model to capture richer and more diverse semantic information, better handling the semantic diversity introduced by multimodality.

- **Dependency Capture**: A transition matrix is used to explicitly model the semantic dependencies between different states, and the States Alignment algorithm based on dynamic programming efficiently provides supervision signals to these hidden states.

Therefore, we will first introduce the details of context enhancement, which includes States Expansion and Dependency Capture (Section 4.2), and then present the States Alignment algorithm for ExLM pre-training (Section 4.3).

4.2. Modeling Semantics with Enhanced Context

The core of context enhancement process is to create multiple hidden states for each [MASK] token and capture the dependencies among these states, enhancing the model's ability to capture richer semantic information. This process consists of three main steps. First, the model expands each [MASK] token in the input into multiple hidden states, with the number of states determined by a hyperparameter k. Secondly, to enable the model to distinguish between these expanded states, it further employs 2D Rotary Position Embedding (RoPE) to differentiate the states. Finally, ExLM uses a transition matrix to explicitly capture the semantic dependencies between these hidden states, enhancing the model's ability to model the missing semantic dependencies.

Expanding [MASK] tokens in the inputs. A key idea in ExLM is to duplicate the embeddings of [MASK] tokens, effectively creating multiple "clones" for each [MASK]

before feeding them into the model. More formally, for each [MASK] token in the original input sequence $\tilde{\mathbf{X}} = [x_1, x_2, \ldots, x_i = [\text{MASK}], \ldots, x_n]$, we take its embedding $\mathbf{e}_{[\text{MASK}]}$ and make k copies. We then form an expanded input sequence by replacing the single [MASK] token with its k duplicated embeddings:

$$\mathbf{X}' = [\mathbf{e}_{x_1}, \mathbf{e}_{x_2}, \ldots, \mathbf{e}^{(1)}_{[\text{MASK}]}, \ldots, \mathbf{e}^{(k)}_{[\text{MASK}]}, \ldots, \mathbf{e}_{x_n}],$$

where $\mathbf{e}_{x_i}$ is the embedding of token x_i, $\mathbf{e}^{(i)}_{[\text{MASK}]}$ is the i-th copy of [MASK] embedding $\mathbf{e}_{[\text{MASK}]}$, and k is a hyperparameter controlling the number of cloned embeddings.

This expanded sequence $\mathbf{X}'$, consisting of embeddings for both the original tokens and the duplicated [MASK] tokens, is then passed into a Transformer Encoder $\boldsymbol{\theta}$ for contextual encoding $\mathbf{H}$ (Vaswani et al., 2017):

$$\mathbf{H} = [\mathbf{h}_{x_1}, \mathbf{h}_{x_2}, \ldots, \mathbf{h}^{(1)}_{[\text{MASK}]}, \ldots, \mathbf{h}^{(k)}_{[\text{MASK}]}, \ldots, \mathbf{h}_{x_n}],$$

where each $\mathbf{h}_t$ represents the hidden state corresponding to the input embedding $\mathbf{e}_t$ in $\mathbf{X}'$. By explicitly expanding [MASK] tokens, the model is equipped to learn richer and more diverse representations for the missing information, leveraging these enriched embeddings to better capture semantic information and reconstruct the original context.

Using 2D RoPE to distinguish expanded states. In the context enhancement process, each [MASK] token is duplicated multiple times, creating several "clones" that the model may find difficult to distinguish. To address this challenge, we introduce a 2D Rotary Position Embedding (RoPE) mechanism (Su et al., 2021), which leverages a second dimension in the positional information to differentiate these clones. Specifically, if the original [MASK] token is located at position i in the sequence, its k duplicates are assigned unique 2D positions: $(i, 1), (i, 2), \ldots, (i, k)$. Meanwhile, all original (non-[MASK]) tokens retain their original positions, represented as $(j, 0)$, where j denotes their index in the sequence. Here, the first coordinate captures the token's position in the original sequence, while the second coordinate differentiates the clones of a [MASK] token. Using this two-dimensional positional structure, the 2D RoPE mechanism applies rotational position embeddings to encode both the sequence position and the clone index.

Using a transition matrix to capture dependencies between expanded states. The semantic dependencies between expanded states can be modeled as a directed acyclic graph (DAG), where each node represents a state, and the edge weights indicate the strength of the semantic dependencies between pairs of states. Similar to previous work (Huang et al., 2022), we adopt a DAG to effectively capture these semantic dependencies.

Specifically, the representation of each state $\mathbf{h}_i$ extracted by the Transformer Encoder undergoes an attention-like computation to derive the transition matrix $\mathbf{E}$. This transition matrix $\mathbf{E}$ serves as the adjacency matrix of the DAG, quantifying the semantic association strength between different states. The computation is defined as follows:

$$\mathbf{E} = \text{softmax}\left(\frac{\mathbf{Q}\mathbf{K}^\top}{\sqrt{d}} + \mathbf{M}\right),$$

$$\mathbf{Q} = \mathbf{H}\mathbf{W}_{\text{Q}}, \quad \mathbf{K} = \mathbf{H}\mathbf{W}_{\text{K}},$$

where d is the hidden size, $\mathbf{W}_{\text{Q}}$ and $\mathbf{W}_{\text{K}}$ are learnable weight matrices, $\mathbf{M}$ is an upper triangular mask matrix ensuring that $\mathbf{E}$ remains an upper triangular matrix, thereby enforcing the DAG structure by preventing backward edges.

Additionally, each state representation $\mathbf{h}_i$ is passed through the model's token prediction head to compute the probability distribution over possible tokens for that state:

$$\mathbf{P} = \text{softmax}(\mathbf{H}\mathbf{W}_{\text{P}}^\top),$$

where $\mathbf{P}$ represents the probability distributions for each state, $\mathbf{W}_{\text{P}}$ is a learnable weight matrix for token prediction.

By using the transition matrix $\mathbf{E}$, the model explicitly captures the semantic dependencies between expanded states, enhancing its ability to reconstruct the missing semantic information effectively.

4.3. Pre-training of ExLM: States Alignment

A key challenge in training ExLM lies in the fact that each [MASK] token is expanded into multiple hidden states. This expansion means that there are more hidden states than the target tokens to predict. Consequently, we must determine an alignment between these states and the target tokens—that is, deciding which hidden state should be responsible for predicting which token. This alignment process is at the heart of our States Alignment algorithm.

Formulating States Alignment as a DAG decoding problem. Our goal is to maximize the probability of the DAG decoding all target tokens $\mathbf{Y}$ (i.e., all masked tokens). Formally, let Γ denote the set of all possible paths (i.e., all possible ways to align states to the target tokens). Each path $\mathbf{A} \in \Gamma$ represents one particular alignment of states to the target tokens in $\mathbf{Y}$. The training objective can thus be written as a marginalization over all possible alignments:

$$\mathcal{L}_{\text{SA}} = -\log P_\theta(\mathbf{Y} \mid \mathbf{X}') = -\log \sum_{\mathbf{A} \in \Gamma} P_\theta(\mathbf{Y}, \mathbf{A} \mid \mathbf{X}'),$$

where $\mathbf{X}'$ is the input sequence (with enhanced context), θ denotes the model parameters, and $P_\theta(\mathbf{Y}, \mathbf{A} \mid \mathbf{X}')$ represents the probability of generating the target sequence $\mathbf{Y}$ through a specific alignment $\mathbf{A}$.

Table 1. The overall results on 7 molecule property classification datasets. We report ROC-AUC score (higher is better) under scaffold splitting. The best results are **bold**. The second-best results are underlined. * indicates that the model uses the same training data, model architecture, and training hyperparameters as ExLM. For more detailed information about the dataset, please refer to Table 9.

Datasets	BACE↑	BBBP↑	Tox21↑	SIDER↑	MUV↑	ClinTox↑	ToxCast↑	Mean↑
# Molecules	1531	2039	7831	1427	93087	1478	8575	-
D-MPNN	80.9	71.0	75.9	57.0	<u>78.6</u>	<u>90.6</u>	<u>65.5</u>	<u>74.2</u>
Attentive FP	78.4	64.3	76.1	60.6	76.6	84.7	63.7	72.1
N-Gram$_{RF}$	77.9	69.7	74.3	**66.8**	76.9	77.5	-	-
GROVER	**82.6**	70.0	74.3	<u>64.8</u>	62.5	81.2	65.4	71.5
GraphMVP	<u>81.2</u>	<u>72.4</u>	75.9	63.9	77.7	79.1	63.1	73.3
Mole-BERT	80.8	71.9	<u>76.8</u>	62.8	<u>78.6</u>	78.9	64.3	73.4
3D InfoMax	79.7	69.1	74.5	60.6	74.4	79.9	64.4	71.8
SMILES-BERT*	77.8	68.6	75.1	61.2	75.1	89.8	64.9	73.2
ExLM	79.6	**72.8**	**78.2**	64.5	**78.8**	**91.6**	**66.9**	**76.1**

This objective requires summing over all possible alignment paths $\mathbf{A} \in \Gamma$, which can be computationally expensive. To address this, we employ a dynamic programming (DP) algorithm that efficiently computes this sum with a complexity of $\mathcal{O}(M)$, where M is the number of all masked tokens.

More specifically, we adopted a dynamic programming algorithm similar to that used in DA-Transformer (Huang et al., 2022). In this DP scheme, we define $f_{i,u}$ as the cumulative probability of all partial paths ending at state u (a node in the DAG) that have generated the first i tokens of $\mathbf{Y}$. Formally, u indexes the states in our DAG in a manner that respects the acyclic property (i.e., we only move forward in the state sequence), and i ranges over the positions of the target sequence $\mathbf{Y}$.

The DP recursion works by summing over all valid predecessors v of u (where $v < u$ in the DAG):

$$f_{i,u} = \sum_{v<u} f_{i-1,v} \times \mathbf{E}_{v,u} \times \mathbf{P}_u(y_i),$$

where $\mathbf{E}_{v,u}$ is the transition score from state v to state u (derived from our transition matrix), $\mathbf{P}_u(y_i)$ is the probability of state u predicting the i-th target token y_i.

By computing $f_{i,u}$ across all i and u, we eventually obtain $f_{M,L}$, where M is the number of masked tokens in the target sequence $\mathbf{Y}$, and L refers to the final state index. The final training objective is:

$$\mathcal{L}_{\text{SA}} = -\log f_{M,L}.$$

This dynamic programming approach reduces the time complexity of the alignment problem to $\mathcal{O}(M \times L^2)$, offering a significant improvement over exhaustive path enumeration. In practice, the computation can be further optimized to $\mathcal{O}(M)$ by leveraging parallelized operations provided by PyTorch (Paszke, 2019), making the method highly efficient

and suitable for large-scale training. A detailed analysis of the DP algorithm's efficiency is provided in Appendix H.

Through this approach, ExLM effectively explores over all possible alignments between the expanded states and target tokens, leveraging both the token probability distributions $\mathbf{P}$ and the transition matrix (the edges in DAG) $\mathbf{E}$ to reconstruct the missing semantics during pre-training.

5. Experimental Results

We evaluate the performance of the ExLM model in both text modeling and SMILES modeling tasks. SMILES is a sequential representation of molecular information, and a more detailed explanation can be found in Appendix D.

We pre-train ExLM[1] on data of text and SMILES separately. Then we fine-tune and evaluate ExLM across diverse benchmarks and verify the contribution of each component through ablation studies. Finally, a visualization analysis is included to explain the advantages of unified modeling.

5.1. Results on SMILES Representation Learning

Pre-training. For SMILES pre-training, we use the large-scale molecular dataset provided by Zhou et al. (2023), which includes SMILES information for 19 million molecules. We tokenize SMILES sequences with the regular expression from Schwaller et al. (2018). The pre-training hyperparameters can be found in Appendix I.

Fine-tuning. For fine-tuning, we employ the widely-recognized MoleculeNet benchmark (Wu et al., 2018). We follow the same data split as used by Zhou et al. (2023). Details of the fine-tuning datasets and baselines can be found in Appendix K. We fine-tune the ExLM model on downstream task datasets using three different random seeds and reported the average performance of the model.

---

[1]Code is released at https://github.com/zhengkangjie/ExLM.

Table 2. The overall results on the GLUE and SQuAD 2.0 development sets (medians over five random seeds). Results not available in pervious research are marked with "–". The "MEAN" column contains the averaged results across the eight GLUE tasks.

Model	GLUE									SQuAD 2.0	
	MNLI-(m/mm)	QQP	QNLI	SST-2	CoLA	RTE	MRPC	STS-B	MEAN	EM	F1
BERT	84.5/-	91.3	91.7	93.2	58.9	68.6	87.3	<u>89.5</u>	83.1	73.7	76.3
ALBERT	81.6/-	–	–	90.3	–	–	–	–	–	77.1	80.0
XLNet	85.8/85.4	–	–	92.7	–	–	–	–	–	78.5	81.3
UniLMv2	86.1/86.1	–	–	93.2	–	–	–	–	–	<u>80.9</u>	<u>83.6</u>
TUPE	<u>86.2</u>/<u>86.2</u>	91.3	92.2	93.3	63.6	73.6	**89.9**	89.2	84.9	–	–
RoBERTa[*]	85.9/85.8	<u>91.6</u>	<u>92.3</u>	<u>93.7</u>	<u>64.3</u>	<u>75.5</u>	88.7	<u>89.5</u>	<u>85.2</u>	78.3	81.5
ExLM	**86.9/86.7**	**92.0**	**93.1**	**93.9**	**64.6**	**78.8**	<u>89.6</u>	**90.5**	**86.2**	**82.0**	**84.6**
ExLM$_{\text{LARGE}}$	87.8/87.5	92.2	93.8	94.5	65.3	79.1	90.4	91.2	86.9	82.6	85.0

Results. As show in Table 1, ExLM achieves the best performance among all baseline models on 5 out of 7 molecular property classification tasks and closely matches the best baseline models on the remaining two tasks. A noteworthy result is that ExLM significantly outperforms the MLM pre-trained with the same model architecture, pre-training hyperparameters, and pre-training data—namely, the SMILES-BERT model. This further underscores the advantages of ExLM over traditional MLM pre-training tasks. ExLM also demonstrates the strongest average performance across all 7 tasks, indicating that it outperforms other baseline models in these prediction tasks overall.

5.2. Results on Textual Representation Learning

Pre-training. For textual pre-training, we adopt the English Wikipedia and BookCorpus datasets (Devlin, 2018) as the pre-training dataset. The model size of ExLM is consistent with BERT base (Devlin, 2018). We also train ExLM$_{\text{LARGE}}$ with the same size as BERT large. For more details about pre-training settings, please see Appendix M.

Fine-tuning. We evaluate the ExLM model using the the GLUE (Wang et al., 2018) and SQuAD 2.0 (Rajpurkar et al., 2018) benchmarks. Detailed of the GLUE benchmark are provided in Appendix L. For fine-tuning, we follow the standard procedures used in BERT (Devlin, 2018) and RoBERTa (Liu, 2019). We also provide the hyperparameter search space for fine-tuning in Appendix N. Consistent with previous studies (Liu, 2019), all reported fine-tuning results represent the median values obtained from five different random seeds across both GLUE and SQuAD benchmarks.

Results. As shown in Table 2, we evaluate ExLM on the GLUE and SQuAD 2.0 development sets. ExLM achieves the best performance on SQuAD 2.0 and 7 GLUE tasks, and also performs very closely to the best baseline model on the remaining one GLUE task. Furthermore, ExLM demonstrates significantly higher average performance compared to other baseline models. Notably, RoBERTa in the

Table 3. **Ablation studies.** We assess the effectiveness of 2D RoPE and the transition matrix in ExLM and verify its efficiency.

Method	MNLI ↑	QNLI↑	QQP↑	RTE↑	Avg ↑
ExLM w/o 2D RoPE	84.6	91.1	91.3	56.7	80.9
ExLM w/o Transitions	83.8	90.9	91.1	55.6	80.4
Vanilla MLM	83.6	90.0	90.3	54.7	79.6
Vanilla MLM++	84.4	91.2	90.6	56.3	80.7
ExLM	**85.1**	**91.4**	**91.3**	**57.6**	**81.4**

table uses the same pre-training data and settings as ExLM, yet ExLM shows significant improvements. This demonstrates the superior performance and effectiveness of ExLM. ExLM$_{\text{LARGE}}$ also achieves obvious performance improvements compared to ExLM due to the increased scale.

5.3. Analytical Experiments of ExLM

We validate ExLM's effectiveness through ablation studies and explore the impact of expanded states and mask ratios on its performance. A case study also demonstrates its good ability to capture rich semantic information. Efficiency and entropy analysis of ExLM are provided in Appendix P.

Ablation studies on ExLM. We perform ablation studies on the 2D RoPE and transition matrix in ExLM, as shown in Table 3. The results reveal that removing the 2D RoPE (ExLM w/o 2D RoPE) or the transition matrix (ExLM w/o Transitions) causes a significant performance drop, with the transition matrix having a larger impact. This demonstrates the importance of the transition matrix in capturing semantic dependencies. Despite the performance decline from removing the transition matrix or the 2D RoPE, the expanded states still enable the model to capture semantic information more effectively, outperforming Vanilla MLM. To assess ExLM's efficiency, we train an MLM with the same training cost, Vanilla MLM++, and show in Table 3 that ExLM performs better, highlighting that ExLM can capture semantic information more efficiently than MLM.

Table 4. **Performance comparison of MLM and ExLM with different mask ratios p and numbers of expanded states k.** ExLM demonstrates stronger modeling capability for contexts with a high mask ratios compared with MLM, and increasing k further improves its performance under high mask ratios inputs.

p	Method	MNLI $\uparrow$	QNLI $\uparrow$	QQP $\uparrow$	RTE $\uparrow$	Avg
15%	Vanilla MLM	83.6	90.0	90.3	54.7	79.6
15%	ExLM-k=2	84.6	91.3	91.1	56.7	80.9
	ExLM-k=4	**85.1**	**91.4**	**91.3**	**57.6**	**81.4**
	ExLM-k=8	84.4	91.0	90.9	56.9	80.8
38.7%	Vanilla MLM	83.3	90.0	90.2	53.3	79.2
38.7%	ExLM-k=2	84.1	91.4	91.0	55.4	80.5
	ExLM-k=4	84.9	91.6	91.2	57.0	81.2
	ExLM-k=8	84.3	90.9	90.7	56.8	80.7

Figure 7. **Case study.** We visualize the model's predictions when the input is "*This is `[MASK]`, and I'm very `[MASK]` to see this.*" ($k = 4$). The **yellow nodes** correspond to the expanded states of the first `[MASK]` token, while the **brown nodes** correspond to those of the second `[MASK]` token. The weights in the figure represent the transition probability between two nodes in the DAG, and the axis labels show the top-1 predicted word for each node.

The impact of k and p. We explore the impact of the number of expanded states k and mask ratios p on ExLM's performance, with results shown in Table 4. As k increases, performance improves due to enhanced semantic modeling ability. However, when k becomes too large (e.g., $k = 8$), performance slightly declines due to the excessive expanded states of `[MASK]` tokens in the input context and redundant nodes in the DAG. Thus, with a mask ratio of 15%, $k = 4$ proves to be the optimal choice. We further verify the impact of the mask ratio p on the performance of ExLM, with the results shown in Table 4. These results show that a higher mask ratio p has less impact on ExLM's performance compared to MLM, highlighting the enhanced semantic modeling capability of ExLM. Furthermore, ExLM needs more expanded states to model the multimodal and ambiguous context induced by a higher mask ratio. Thus, increasing k appropriately at higher mask ratios enhances its performance. However, an excessively high mask ratio and too many expanded states (a higher k) can result in a large number of expanded states of `[MASK]` tokens in the input context, limiting further performance gains of ExLM.

Case Study. We conduct a case study to evaluate the model's performance on specific inputs, as shown in Figure 7. The results demonstrate that the model effectively handles multimodal semantic information, capturing four possible semantic choices for each `[MASK]` token. The transition matrix models the relationships between these choices, for example, when the first `[MASK]` token is *amazing*, the second is more likely to be *glad*, and when the first is *terrible*, the second tends to be *sorry*. A more detailed analysis can be found in Appendix O. These results show that ExLM can capture rich contextual information.

6. Conclusions

In this paper, we analyze the impact of the semantics corruption caused by `[MASK]` tokens on MLMs, showing that it has a more signi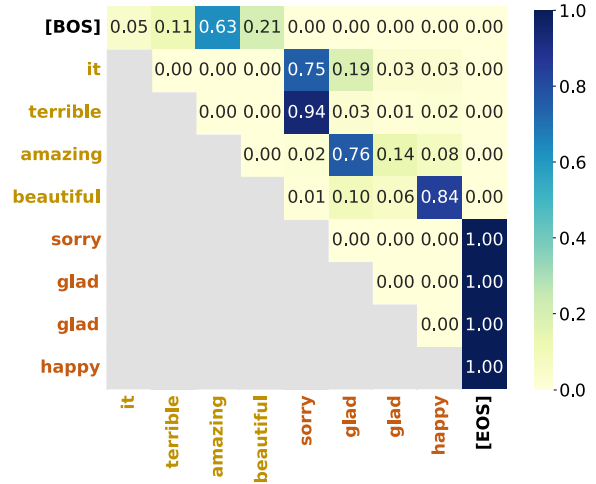ficant impact than the unreal token problem, offering a new perspective on understanding MLMs. Based on our analysis, we propose ExLM, which enhances the model's semantic modeling ability through two key designs: States Expansion and Dependency Capture. These designs reduce the negative impact of the semantic multimodality on the model. We demonstrate ExLM's strong performance in both text and SMILES modeling scenarios. Ablation studies and case study also validate its effectiveness and efficiency.

Acknowledgements

Impact Statement

This paper is the first work to systematically study the impact of `[MASK]` tokens on MLM from the perspectives of unreal tokens and semantics corruption, and proposes effective solutions to these problems. These analytical results provide valuable insights for designing better lan-

guage representation learning models. Furthermore, MLM-based sequence representation learning models have already achieved significant success in various domains, including text, images, video, and scientific data. These models have also shown promising results in applications such as recommendation systems and protein design. Therefore, the methods proposed in this paper have the potential to drive further advancements in these fields, and the approach also holds significant application potential. However, we also acknowledge that this work inherits the potential negative impacts of existing MLMs, such as the possibility of being used to generate false information or design and manufacture molecules with biological hazards.

References

Bao, H., Dong, L., Wei, F., Wang, W., Yang, N., Liu, X., Wang, Y., Gao, J., Piao, S., Zhou, M., et al. Unilmv2: Pseudo-masked language models for unified language model pre-training. In *International conference on machine learning*, pp. 642–652. PMLR, 2020.

Bao, H., Dong, L., Piao, S., and Wei, F. Beit: Bert pre-training of image transformers. *arXiv preprint arXiv:2106.08254*, 2021.

Bentivogli, L., Clark, P., Dagan, I., and Giampiccolo, D. The fifth pascal recognizing textual entailment challenge. In *TAC*, 2009.

Cer, D., Diab, M., Agirre, E., Lopez-Gazpio, I., and Specia, L. Semeval-2017 task 1: Semantic textual similarity multilingual and crosslingual focused evaluation. In *International Workshop on Semantic Evaluation (SemEval)*, 2017.

Chithrananda, S., Grand, G., and Ramsundar, B. Chemberta: large-scale self-supervised pretraining for molecular property prediction. *arXiv preprint arXiv:2010.09885*, 2020.

Clark, K. Electra: Pre-training text encoders as discriminators rather than generators. *arXiv preprint arXiv:2003.10555*, 2020.

Dagan, I., Glickman, O., and Magnini, B. The pascal recognising textual entailment challenge. In *Machine Learning Challenges Workshop*, 2005.

Dai, Y., Li, L., Zhou, C., Feng, Z., Zhao, E., Qiu, X., Li, P., and Tang, D. ” is whole word masking always better for chinese bert?”: Probing on chinese grammatical error correction. *arXiv preprint arXiv:2203.00286*, 2022.

Devlin, J. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

Dolan, W. B. and Brockett, C. Automatically constructing a corpus of sentential paraphrases. In *International Workshop on Paraphrasing (IWP)*, 2005.

Dong, L., Yang, N., Wang, W., Wei, F., Liu, X., Wang, Y., Gao, J., Zhou, M., and Hon, H.-W. Unified language model pre-training for natural language understanding and generation. *Advances in neural information processing systems*, 32, 2019.

Du, Z., Qian, Y., Liu, X., Ding, M., Qiu, J., Yang, Z., and Tang, J. Glm: General language model pretraining with autoregressive blank infilling. *arXiv preprint arXiv:2103.10360*, 2021.

Elnaggar, A., Heinzinger, M., Dallago, C., Rehawi, G., Wang, Y., Jones, L., Gibbs, T., Feher, T., Angerer, C., Steinegger, M., et al. Prottrans: Toward understanding the language of life through self-supervised learning. *IEEE transactions on pattern analysis and machine intelligence*, 44(10):7112–7127, 2021.

Feng, B., Liu, Z., Huang, N., Xiao, Z., Zhang, H., Mirzoyan, S., Xu, H., Hao, J., Xu, Y., Zhang, M., et al. A bioactivity foundation model using pairwise meta-learning. *Nature Machine Intelligence*, 6(8):962–974, 2024.

Feng, Z., Guo, D., Tang, D., Duan, N., Feng, X., Gong, M., Shou, L., Qin, B., Liu, T., Jiang, D., et al. Codebert: A pre-trained model for programming and natural languages. *arXiv preprint arXiv:2002.08155*, 2020.

Fu, Z., Zhou, W., Xu, J., Zhou, H., and Li, L. Contextual representation learning beyond masked language modeling. *arXiv preprint arXiv:2204.04163*, 2022.

Giampiccolo, D., Magnini, B., Dagan, I., and Dolan, B. The third pascal recognizing textual entailment challenge. In *ACL-PASCAL workshop on textual entailment and paraphrasing*, 2007.

Graves, A., Fernández, S., Gomez, F., and Schmidhuber, J. Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks. In *Proceedings of the 23rd international conference on Machine learning*, pp. 369–376, 2006.

Gu, J., Bradbury, J., Xiong, C., Li, V. O., and Socher, R. Non-autoregressive neural machine translation. *arXiv preprint arXiv:1711.02281*, 2017.

Guo, D., Ren, S., Lu, S., Feng, Z., Tang, D., Liu, S., Zhou, L., Duan, N., Svyatkovskiy, A., Fu, S., et al. Graphcodebert: Pre-training code representations with data flow. *arXiv preprint arXiv:2009.08366*, 2020.

Haim, R. B., Dagan, I., Dolan, B., Ferro, L., Giampiccolo, D., Magnini, B., and Szpektor, I. The second pascal

recognising textual entailment challenge. In *PASCAL Challenges Workshop on Recognising Textual Entailment*, 2006.

Hayes, T., Rao, R., Akin, H., Sofroniew, N. J., Oktay, D., Lin, Z., Verkuil, R., Tran, V. Q., Deaton, J., Wiggert, M., et al. Simulating 500 million years of evolution with a language model. *Science*, pp. eads0018, 2025.

He, K., Chen, X., Xie, S., Li, Y., Dollár, P., and Girshick, R. Masked autoencoders are scalable vision learners. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 16000–16009, 2022a.

He, P., Liu, X., Gao, J., and Chen, W. Deberta: Decoding-enhanced bert with disentangled attention. *arXiv preprint arXiv:2006.03654*, 2020.

He, P., Gao, J., and Chen, W. Debertav3: Improving deberta using electra-style pre-training with gradient-disentangled embedding sharing. *arXiv preprint arXiv:2111.09543*, 2021.

He, Z., Sun, T., Wang, K., Huang, X., and Qiu, X. Diffusionbert: Improving generative masked language models with diffusion models. *arXiv preprint arXiv:2211.15029*, 2022b.

Hochreiter, S. Long short-term memory. *Neural Computation MIT-Press*, 1997.

Huang, C., Zhou, H., Jen, C., Zheng, K., Zaiane, O., and Mou, L. A decoding algorithm for length-control summarization based on directed acyclic transformers. In Al-Onaizan, Y., Bansal, M., and Chen, Y.-N. (eds.), *Findings of the Association for Computational Linguistics: EMNLP 2024*, pp. 11572–11583, Miami, Florida, USA, November 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.findings-emnlp.677. URL https://aclanthology.org/2024.findings-emnlp.677/.

Huang, F., Zhou, H., Liu, Y., Li, H., and Huang, M. Directed acyclic transformer for non-autoregressive machine translation. In *International Conference on Machine Learning*, pp. 9410–9428. PMLR, 2022.

Huang, F., Ke, P., and Huang, M. Directed acyclic transformer pre-training for high-quality non-autoregressive text generation. *Transactions of the Association for Computational Linguistics*, 2023.

Jiang, T., Huang, S., Luan, Z., Wang, D., and Zhuang, F. Scaling sentence embeddings with large language models. *arXiv preprint arXiv:2307.16645*, 2023.

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.

Ke, G., He, D., and Liu, T.-Y. Rethinking positional encoding in language pre-training. *arXiv preprint arXiv:2006.15595*, 2020.

Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Lan, Z. Albert: A lite bert for self-supervised learning of language representations. *arXiv preprint arXiv:1909.11942*, 2019.

Li, X. and Li, J. Bellm: Backward dependency enhanced large language model for sentence embeddings. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pp. 792–804, 2024.

Liao, B., Thulke, D., Hewavitharana, S., Ney, H., and Monz, C. Mask more and mask later: Efficient pre-training of masked language models by disentangling the [mask] token. *arXiv preprint arXiv:2211.04898*, 2022.

Lin, Z., Akin, H., Rao, R., Hie, B., Zhu, Z., Lu, W., dos Santos Costa, A., Fazel-Zarandi, M., Sercu, T., Candido, S., et al. Language models of protein sequences at the scale of evolution enable accurate structure prediction. *BioRxiv*, 2022:500902, 2022.

Lin, Z., Akin, H., Rao, R., Hie, B., Zhu, Z., Lu, W., Smetanin, N., Verkuil, R., Kabeli, O., Shmueli, Y., et al. Evolutionary-scale prediction of atomic-level protein structure with a language model. *Science*, 379(6637): 1123–1130, 2023.

Liu, S., Demirel, M. F., and Liang, Y. N-gram graph: Simple unsupervised representation for graphs, with applications to molecules. *Advances in neural information processing systems*, 32, 2019.

Liu, S., Wang, H., Liu, W., Lasenby, J., Guo, H., and Tang, J. Pre-training molecular graph representation with 3d geometry. *arXiv preprint arXiv:2110.07728*, 2021.

Liu, Y. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*, 364, 2019.

Meng, Y., Xiong, C., Bajaj, P., Tiwary, S., Bennett, P., Han, J., and Song, X. Pretraining text encoders with adversarial mixture of training signal generators. *arXiv preprint arXiv:2204.03243*, 2022.

Meng, Y., Krishnan, J., Wang, S., Wang, Q., Mao, Y., Fang, H., Ghazvininejad, M., Han, J., and Zettlemoyer, L. Representation deficiency in masked language modeling. *arXiv preprint arXiv:2302.02060*, 2023.

Namazifar, M., Tur, G., and Hakkani-T"ur, D. Warped language models for noise robust language understanding. In *2021 IEEE spoken language technology workshop (SLT)*, pp. 981–988. IEEE, 2021.

Pan, J. Large language model for molecular chemistry. *Nature Computational Science*, 3(1):5–5, 2023.

Paszke, A. Pytorch: An imperative style, high-performance deep learning library. *arXiv preprint arXiv:1912.01703*, 2019.

Peters, M. E., Neumann, M., Iyyer, M., Gardner, M., Clark, C., Lee, K., and Zettlemoyer, L. Deep contextualized word representations. In Walker, M., Ji, H., and Stent, A. (eds.), *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pp. 2227–2237, New Orleans, Louisiana, June 2018. Association for Computational Linguistics. doi: 10.18653/v1/N18-1202. URL https://aclanthology.org/N18-1202/.

Rajpurkar, P., Jia, R., and Liang, P. Know what you don't know: Unanswerable questions for SQuAD. In *ACL*, 2018.

Ren, X., Wei, W., Xia, L., Su, L., Cheng, S., Wang, J., Yin, D., and Huang, C. Representation learning with large language models for recommendation. In *Proceedings of the ACM on Web Conference 2024*, pp. 3464–3475, 2024.

Rong, Y., Bian, Y., Xu, T., Xie, W., Wei, Y., Huang, W., and Huang, J. Self-supervised graph transformer on large-scale molecular data. *Advances in Neural Information Processing Systems*, 33:12559–12571, 2020.

Ross, J., Belgodere, B., Chenthamarakshan, V., Padhi, I., Mroueh, Y., and Das, P. Large-scale chemical language representations capture molecular structure and properties. *Nature Machine Intelligence*, 4(12):1256–1264, 2022.

Saharia, C., Chan, W., Saxena, S., and Norouzi, M. Non-autoregressive machine translation with latent alignments. In Webber, B., Cohn, T., He, Y., and Liu, Y. (eds.), *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 1098–1108, Online, November 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.emnlp-main.83. URL https://aclanthology.org/2020.emnlp-main.83/.

Schwaller, P., Gaudin, T., Lanyi, D., Bekas, C., and Laino, T. "found in translation": predicting outcomes of complex organic chemistry reactions using neural sequence-to-sequence models. *Chemical science*, 9(28):6091–6098, 2018.

Shankar, I., Nikhil, D., and Kornél, C. First Quora dataset release: Question pairs, 2017. URL https://www.quora.com/q/quoradata/First-Quora-Dataset-Release-Question-Pairs.

Shao, C., Ma, Z., and Feng, Y. Viterbi decoding of directed acyclic transformer for non-autoregressive machine translation. In *Findings of EMNLP 2022*, 2022.

Shin, J., Lee, Y., Yoon, S., and Jung, K. Fast and accurate deep bidirectional language representations for unsupervised learning. *arXiv preprint arXiv:2004.08097*, 2020.

Socher, R., Perelygin, A., Wu, J., Chuang, J., Manning, C. D., Ng, A. Y., and Potts, C. Recursive deep models for semantic compositionality over a sentiment treebank. In *EMNLP*, 2013.

Springer, J. M., Kotha, S., Fried, D., Neubig, G., and Raghunathan, A. Repetition improves language model embeddings. *arXiv preprint arXiv:2402.15449*, 2024.

Stärk, H., Beaini, D., Corso, G., Tossou, P., Dallago, C., Günnemann, S., and Liò, P. 3d infomax improves gnns for molecular property prediction. In *International Conference on Machine Learning*, pp. 20479–20502. PMLR, 2022.

Su, J., Lu, Y., Pan, S., Murtadha, A., Wen, B., and Liu, Y. Roformer: Enhanced transformer with rotary position embedding. *arXiv preprint arXiv:2104.09864*, 2021.

Su, J., Han, C., Zhou, Y., Shan, J., Zhou, X., and Yuan, F. Saprot: Protein language modeling with structure-aware vocabulary. *bioRxiv*, pp. 2023–10, 2023.

Theodoris, C. V., Xiao, L., Chopra, A., Chaffin, M. D., Al Sayed, Z. R., Hill, M. C., Mantineo, H., Brydon, E. M., Zeng, Z., Liu, X. S., et al. Transfer learning enables predictions in network biology. *Nature*, 618(7965):616–624, 2023.

Tong, Z., Song, Y., Wang, J., and Wang, L. Videomae: Masked autoencoders are data-efficient learners for self-supervised video pre-training. *Advances in neural information processing systems*, 35:10078–10093, 2022.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

Wang, A., Singh, A., Michael, J., Hill, F., Levy, O., and Bowman, S. R. GLUE: A multi-task benchmark and analysis platform for natural language understanding. In *EMNLP Workshop BlackboxNLP*, 2018.

Wang, L., Yang, N., Huang, X., Yang, L., Majumder, R., and Wei, F. Improving text embeddings with large language models. *arXiv preprint arXiv:2401.00368*, 2023.

Wang, R., Chen, D., Wu, Z., Chen, Y., Dai, X., Liu, M., Jiang, Y.-G., Zhou, L., and Yuan, L. Bevt: Bert pre-training of video transformers. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 14733–14743, 2022.

Wang, S., Guo, Y., Wang, Y., Sun, H., and Huang, J. Smiles-bert: large scale unsupervised pre-training for molecular property prediction. In *Proceedings of the 10th ACM international conference on bioinformatics, computational biology and health informatics*, pp. 429–436, 2019.

Wang, X., Zheng, Z., Ye, F., Xue, D., Huang, S., and Gu, Q. Diffusion language models are versatile protein learners. *arXiv preprint arXiv:2402.18567*, 2024a.

Wang, X., Zheng, Z., Ye, F., Xue, D., Huang, S., and Gu, Q. Dplm-2: A multimodal diffusion protein language model. *arXiv preprint arXiv:2410.13782*, 2024b.

Warstadt, A., Singh, A., and Bowman, S. R. Neural network acceptability judgments. In *TACL*, 2019.

Weininger, D. Smiles, a chemical language and information system. 1. introduction to methodology and encoding rules. *Journal of chemical information and computer sciences*, 28(1):31–36, 1988.

Wettig, A., Gao, T., Zhong, Z., and Chen, D. Should you mask 15% in masked language modeling? *arXiv preprint arXiv:2202.08005*, 2022.

Williams, A., Nangia, N., and Bowman, S. A broad-coverage challenge corpus for sentence understanding through inference. In *NAACL-HLT*, 2018.

Wu, Z., Ramsundar, B., Feinberg, E. N., Gomes, J., Geniesse, C., Pappu, A. S., Leswing, K., and Pande, V. Moleculenet: a benchmark for molecular machine learning. *Chemical science*, 9:513–530, 2018.

Xia, J., Zhao, C., Hu, B., Gao, Z., Tan, C., Liu, Y., Li, S., and Li, S. Z. Mole-bert: Rethinking pre-training graph neural networks for molecules. In *The Eleventh International Conference on Learning Representations*, 2023.

Xie, Z., Zhang, Z., Cao, Y., Lin, Y., Bao, J., Yao, Z., Dai, Q., and Hu, H. Simmim: A simple framework for masked image modeling. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 9653–9663, 2022.

Xiong, Z., Wang, D., Liu, X., Zhong, F., Wan, X., Li, X., Li, Z., Luo, X., Chen, K., Jiang, H., et al. Pushing the boundaries of molecular representation for drug discovery with the graph attention mechanism. *Journal of medicinal chemistry*, 63(16):8749–8760, 2019.

Yang, F., Wang, W., Wang, F., Fang, Y., Tang, D., Huang, J., Lu, H., and Yao, J. scbert as a large-scale pretrained deep language model for cell type annotation of single-cell rna-seq data. *Nature Machine Intelligence*, 4(10):852–866, 2022.

Yang, J., Zheng, K., Long, S., Nie, Z., Zhang, M., Dai, X., Ma, W.-Y., and Zhou, H. Mol-ae: Auto-encoder based molecular representation learning with 3d cloze test objective. *bioRxiv*, pp. 2024–04, 2024.

Yang, K., Swanson, K., Jin, W., Coley, C., Eiden, P., Gao, H., Guzman-Perez, A., Hopper, T., Kelley, B., Mathea, M., et al. Analyzing learned molecular representations for property prediction. *Journal of chemical information and modeling*, 59(8):3370–3388, 2019.

Zhang, S., Huang, H., Liu, J., and Li, H. Spelling error correction with soft-masked bert. *arXiv preprint arXiv:2005.07421*, 2020.

Zheng, K., Liang, S., Yang, J., Feng, B., Liu, Z., Ju, W., Xiao, Z., and Zhang, M. Smi-editor: Edit-based smiles language model with fragment-level supervision. *arXiv preprint arXiv:2412.05569*, 2024a.

Zheng, K., Long, S., Lu, T., Yang, J., Dai, X., Zhang, M., Nie, Z., Ma, W.-Y., and Zhou, H. Esm all-atom: Multi-scale protein language model for unified molecular modeling. In *Forty-first International Conference on Machine Learning*, 2024b.

Zhong, Q., Ding, L., Liu, J., Du, B., and Tao, D. Can chatgpt understand too? a comparative study on chatgpt and fine-tuned bert. *arXiv preprint arXiv:2302.10198*, 2023a.

Zhong, Q., Ding, L., Liu, J., Liu, X., Zhang, M., Du, B., and Tao, D. Revisiting token dropping strategy in efficient bert pretraining. *arXiv preprint arXiv:2305.15273*, 2023b.

Zhou, G., Gao, Z., Ding, Q., Zheng, H., Xu, H., Wei, Z., Zhang, L., and Ke, G. Uni-mol: A universal 3d molecular representation learning framework. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=6K2RM6wVqKu.

Appendix

A. Related Works

A.1. Language Models for Representation Learning.

MLM for Representation Learning.   Currently, language models serve as an important role in the self-supervised representation learning field and achieve excellent performance across a wide range of tasks. The ELMo model (Peters et al., 2018) first introduces a self-supervised language representation learning model based on bidirectional LSTM (Hochreiter, 1997). Later, the pre-trained masked language model based on the transformer architecture, BERT (Devlin, 2018), is proposed in the field of text modeling. BERT uses a bidirectional transformer encoder to extract meaningful semantic representations from large amounts of text data and applies them to various downstream tasks such as text understanding (Devlin, 2018) and text correction (Zhang et al., 2020; Dai et al., 2022). With the help of the transformer architecture (Vaswani et al., 2017) and the MLM training approach, this model achieves great success in the text domain. Following this, a large number of follow-up works aim to improve these bidirectional encoder-based masked language models (Liu, 2019; Ke et al., 2020; He et al., 2020; Shin et al., 2020; Namazifar et al., 2021; Du et al., 2021; Fu et al., 2022; Wettig et al., 2022; Meng et al., 2022; 2023; He et al., 2022b). Moreover, MLM also succeeds in various representation learning tasks, including images (Bao et al., 2021; Xie et al., 2022; He et al., 2022a), videos (Tong et al., 2022; Wang et al., 2022), codes (Feng et al., 2020; Guo et al., 2020; He et al., 2021), small molecules(Wang et al., 2019; Chithrananda et al., 2020; Ross et al., 2022; Pan, 2023; Feng et al., 2024; Yang et al., 2024; Zheng et al., 2024a), proteins (Elnaggar et al., 2021; Lin et al., 2022; 2023; Zheng et al., 2024b; Hayes et al., 2025), and single-cell data (Yang et al., 2022; Theodoris et al., 2023).

Other Methods & Summary.   In addition to language models trained with the MLM pre-training approach, there are other types of language models used for representation learning tasks, such as autoregressive language models (Wang et al., 2023; Jiang et al., 2023; Springer et al., 2024; Ren et al., 2024; Li & Li, 2024), diffusion language models (Wang et al., 2024a;b), and unified language models (Dong et al., 2019; Bao et al., 2020; Du et al., 2021). These models have demonstrated their effectiveness in representation learning across various tasks. However, overall, MLM is one of the most commonly used approaches for constructing language models for representation learning tasks and is a typical representative of this type of model. Zhong et al. (2023a) also points out that the MLM performs better than LLM in handling paraphrase and similarity tasks. Given the widespread impact and application of MLM, studying the corrupted semantics problem in MLM is of great significance, and it may also play a positive role in advancing the future development of MLM in different fields.

A.2. Studies of `[MASK]` in MLM pre-training.

Due to the widespread use of MLM and the important role of the `[MASK]` tokens in the MLM training process, several previous works have studied the impact of the `[MASK]` tokens on MLM. Clark (2020) proposed the ELECTRA model, an improvement on MLM, where the `[MASK]` token is not included in the context during pre-training. This approach helps address the inconsistency between pre-training and downstream task fine-tuning caused by `[MASK]` unreal tokens. Liao et al. (2022) further studied the impact of the `[MASK]` unreal tokens on MLM models and pointed out that removing `[MASK]` tokens from the model's shallow representations during training does not affect MLM performance. Meanwhile, Meng et al. (2023) explored the representation deficiency problem caused by `[MASK]` unreal tokens from a theoretical perspective and proposed a MAE-based language model training approach. This approach improves the model's representation learning ability by removing the `[MASK]` tokens from the model input to avoid the unreal tokens problem. Additionally, Wettig et al. (2022) and Liao et al. (2022) also explored the impact of higher mask ratio on MLM performance. While these works have studied the effects of the `[MASK]` tokens, they focus only on the unreal tokens aspect and lack an investigation into the model's behavior from the perspective of semantic corruption. In contrast, Zhong et al. (2023b) points out that dropping `[MASK]` tokens causes semantic loss, affecting performance on semantic-intense tasks like RTE (Dagan et al., 2005). This finding further highlights the negative impact of token dropping on the model's semantic modeling capability, emphasizing the importance of addressing the semantic loss problem. However, this work only analyzes the additional semantic loss caused by the token dropping process compared to the token masking process, and does not analyze the semantic loss caused by the token masking process itself. Therefore, its conclusion can not generalize to MLMs. This work aims to fill this gap by analyzing MLM's behavior from the perspective of both the semantic corruption problem and the unreal tokens problem.

B. Hyper-Parameter Configuration for *Repeated MLM* Experiments

B.1. Pre-training Configuration

In the *Repeated MLM* experiment, we train a series of MLMs with different p and k parameters (a total of 14 sets of models). When k is large, the input sequence length of the model increases significantly, which also results in a high training cost. To accelerate the training process, we use a larger batch size and a larger learning rate to help the model converge more quickly, and we also reduce the number of training steps to lower the overall training cost, ensuring that the training expense is manageable. The model size used in this part of the experiment is the same as BERT base (Devlin, 2018). Specifically, the model has 12 stacked Transformer layers, each with 12 attention heads. The model dimension and feedforward dimension of each Transformer layer are 768 and 3,072, respectively. The total number of parameters in the model is 128M. Furthermore, to ensure that the increase in k does not result in an excessively long sequence that the model cannot process, the model's *Max sequence length* parameter (the maximum length of the position embedding) increases with k, with a specific value of $512 * k$. And we set the learning rate as 0.002 and warmup steps as 5K. The total training steps are 50K and each batch has 4096 samples at maximum.

To ensure the comparability of the results, all other pre-training hyperparameters in the MLM across different experimental groups are exactly the same, except for the *Max sequence length* parameter. In addition to the *Repeated MLM* experiment, we also used the same pre-training parameters in the ablation experiments to reduce pre-training costs. For more pre-training hyper-parameters, please refer to Table 5.

Table 5. Hyper-parameters for the *Repeated MLM* experiments.

Hyper-parameters	Value
Learning rate	2e-3
LR scheduler	polynomial_decay
Warmup updates	5K
Max updates	50K
Batch size	$4,096$
FFN dropout	0.1
Attention dropout	0.1
Activation dropout	0
Num of layers	12
Num of attention heads	12
Encoder embedding dim	768
Encoder FFN dim	$3,072$
Adam (β_1, β_2)	$(0.9, 0.98)$
Mask ratio	0.15
Activation function	GELU
Weight Decay	0.01
Clip Norm	0.0
Max sequence length	$512*k$

B.2. Fine-tuning Configuration

To ensure the comparability of the experiments, the MLMs in different sets of experiments use exactly the same training parameters during the fine-tuning phase (except for the *Max sequence length* parameter, which is set the same as in the pre-training phase). Moreover, to ensure consistency between the pre-training and fine-tuning processes, we repeat each token in the input sequences k times during fine-tuning, using the same method as in the pre-training phase. Each MLM is trained three times with different random seeds on each downstream task, and the average of these three results is taken as the final outcome. The detailed training parameters are provided in Table 6.

C. Expectation and Variance of the Proportion of Corrupted Semantics

To quantify the proportion of corrupted semantics, consider a sequence of N tokens, and each token is duplicated by k times, resulting in a total of $N \times k$ tokens. Each duplicated copy is independently masked with probability p. A token is deemed to have corrupted semantics if all k of its copies are masked. Formally, the corrupted semantics proportion s is defined as:

Hyperparameter	MNLI, QNLI, QQP, RTE
Peak learning rate	5e-5
Batch size	32
Max epochs	3
Warm-Up Proportion	6%
Max sequence length	512*k
Fine-tuning Seeds	$\{1, 2, 3\}$

Table 6. Fine-tuning Hyper-Parameter Configuration for *Repeated MLM* Experiments.

$$s = \frac{1}{N} \sum_{i=1}^{N} X_i$$

where X_i is an indicator variable defined by:

$$X_i = \begin{cases} 1, & \text{if all } k \text{ copies of token } i \text{ are masked,} \\ 0, & \text{otherwise.} \end{cases}$$

Each of these k copies is independently masked with probability p. To determine if the semantics of token i are corrupted, we need all k copies of the token to be masked simultaneously. Thus the probability that all k copies of token i are masked is simply the product of the individual masking probabilities. Therefore, the probability that $X_i = 1$, meaning that all k copies are masked and the token's semantics are corrupted, is given by:

$$\mathbb{P}(X_i = 1) = p^k.$$

Since X_i is an indicator variable that takes the value 1 if the token's semantics are corrupted, and 0 otherwise, it follows a Bernoulli distribution with parameter p^k, i.e., $X_i \sim \text{Bernoulli}(p^k)$. Thus the expectation of X_i is p^k and the variance of X_i is $p^k(1 - p^k)$.

Therefore the expectation and variance of s can be derived as follows:

$$\mathbb{E}[s] = \mathbb{E}\Big[\frac{1}{N} \sum_{i=1}^{N} X_i\Big] = \frac{1}{N} \sum_{i=1}^{N} \mathbb{E}[X_i] = p^k, \quad \text{Var}(s) = \text{Var}\Big(\frac{1}{N} \sum_{i=1}^{N} X_i\Big) = \frac{1}{N^2} \sum_{i=1}^{N} \text{Var}(X_i) = \frac{p^k(1 - p^k)}{N}.$$

This analysis leverages the independence of masking each token, leading to the conclusion that since the length of the input sequence N is typically large, the variance of s is very small. It also proves that when the repetition times are k and the mask ratio is p, the proportion of corrupted semantics is p^k.

D. A Brief Introduction to SMILES

SMILES (Simplified Molecular Input Line Entry System) (Weininger, 1988) is a notation system used to represent chemical structures in a text format. It encodes molecular information using a series of characters, where atoms are represented by their chemical symbols (e.g., C for carbon, O for oxygen), bonds by symbols like "-" (single bond), "=" (double bond), and "#" (triple bond), and branches are indicated with parentheses. For example, the SMILES representation of ethanol is "CCO", where "C" stands for carbon atoms and "O" represents an oxygen atom, with a single bond between them. The notation captures the connectivity between atoms, bond types, and sometimes stereochemistry, making it a compact and easily readable way to represent molecular structures. The principle behind SMILES is that it provides a linear representation of the molecule that can be easily interpreted by both humans and computational systems, facilitating tasks like database searching, molecular simulations, and chemical informatics.

E. Calculation of Entropy in *Repeated MLM*

To analyze the uncertainty of MLM predictions, we calculate the entropy of the model's predicted distribution P for missing tokens. Entropy is a measure of uncertainty, calculated as:

$$H[P] = - \sum_{x \in \text{voc}} P(x) \log P(x),$$

where $P(x)$ is the predicted probability of token x at a given masked position, voc is the vocabulary.

A larger entropy indicates a more dispersed prediction distribution P, which also suggests that the model has higher uncertainty.

E.1. Adjustments for Repeated Tokens

In the *Repeated MLM* experiment, some tokens are repeated multiple times, which introduces additional considerations for calculating entropy. Without proper filtering, the calculated entropy would be influenced by tokens that are not fully masked. This means that the entropy would reflect the uncertainty associated with the model's predictions for partially visible tokens, resulting in an artificially lower entropy that fails to represent the true uncertainty under full masking conditions. To ensures the entropy calculation only considers tokens where all k copies are masked, we introduce a filtering criterion. Any token with at least one unmasked copy is excluded from the computation.

For a token that appears k times and is fully masked, the entropy is determined by averaging the individual entropies of the prediction distributions for each of its masked copies. This approach gives a more consistent and meaningful representation of the model's uncertainty. Formally, the entropy of a token with k masked positions is defined as:

$$H_{\text{token}} = \frac{1}{k} \sum_{j=1}^{k} H[P(x_j)],$$

where $H[P(x_j)]$ is the entropy of the prediction distribution at the j-th repetition of the fully masked token, $P(x_j)$ represents the model's predicted probability distribution over the vocabulary at that position. The summation iterates over all k masked positions, and the resulting average provides the token's overall entropy.

This method ensures that the entropy metric accurately reflects the model's uncertainty in the fully masked scenario, without interference from partially visible tokens or the added complexity of repeated instances.

E.2. Data and Experimental Setup

This entropy analysis is conducted on the validation set to ensure that the results reflect the model's behavior on unseen data and avoid any training bias. By focusing on fully masked tokens and calculating their entropy, this approach provides a precise measure of the model's uncertainty in predicting missing semantic information.

F. More Results of *Repeated MLM* Experiments

k \ P	2.25%	15%	38.7%	62.2%	78.9%
1	81.8	84.1	83.9	81.4	80.3
2		81.6	83.5	83.1	81.9
4			81.8	83.6	82.7
8				79.8	82.5

(a) MNLI-m

k \ P	2.25%	15%	38.7%	62.2%	78.9%
1	88.1	90.0	90.0	88.6	86.9
2		88.2	89.7	89.5	88.7
4			87.8	89.0	88.7
8				86.4	88.8

(b) QNLI

k \ P	2.25%	15%	38.7%	62.2%	78.9%
1	89.2	90.3	90.2	89.5	89.2
2		89.9	90.6	90.3	89.9
4			90.0	90.6	90.2
8				89.7	90.4

(c) QQP

k \ P	2.25%	15%	38.7%	62.2%	78.9%
1	50.4	54.7	53.3	50.9	48.0
2		51.7	54.6	54.3	54.4
4			52.4	56.9	56.2
8				48.6	56.7

(d) RTE

Figure 8. The results of the *Repeated MLM* experiments on MNLI-m, QNLI, QQP, and RTE tasks. We use accuracy in these four tasks as the metrics. And for all tasks, higher values are better. Similar performance are marked with similar colors.

We extend the *Repeated MLM* experiment to a broader range of tasks, with the results presented in Figure 8. From these results, we observe that the conclusions drawn in Section 3.2 still hold. Although the exact performance trends vary from task to task, the overall pattern remains clear and consistent: **the severity of corrupted semantics directly correlates with performance degradation**.

Specifically, as the intensity of corrupted semantics changes, the model's performance on these tasks exhibits significant fluctuations. Notably, the magnitude of these changes is substantially greater than the performance variations observed when the mask ratio is altered. This suggests that corrupted semantics plays a more critical role in influencing model performance than the presence of unreal tokens.
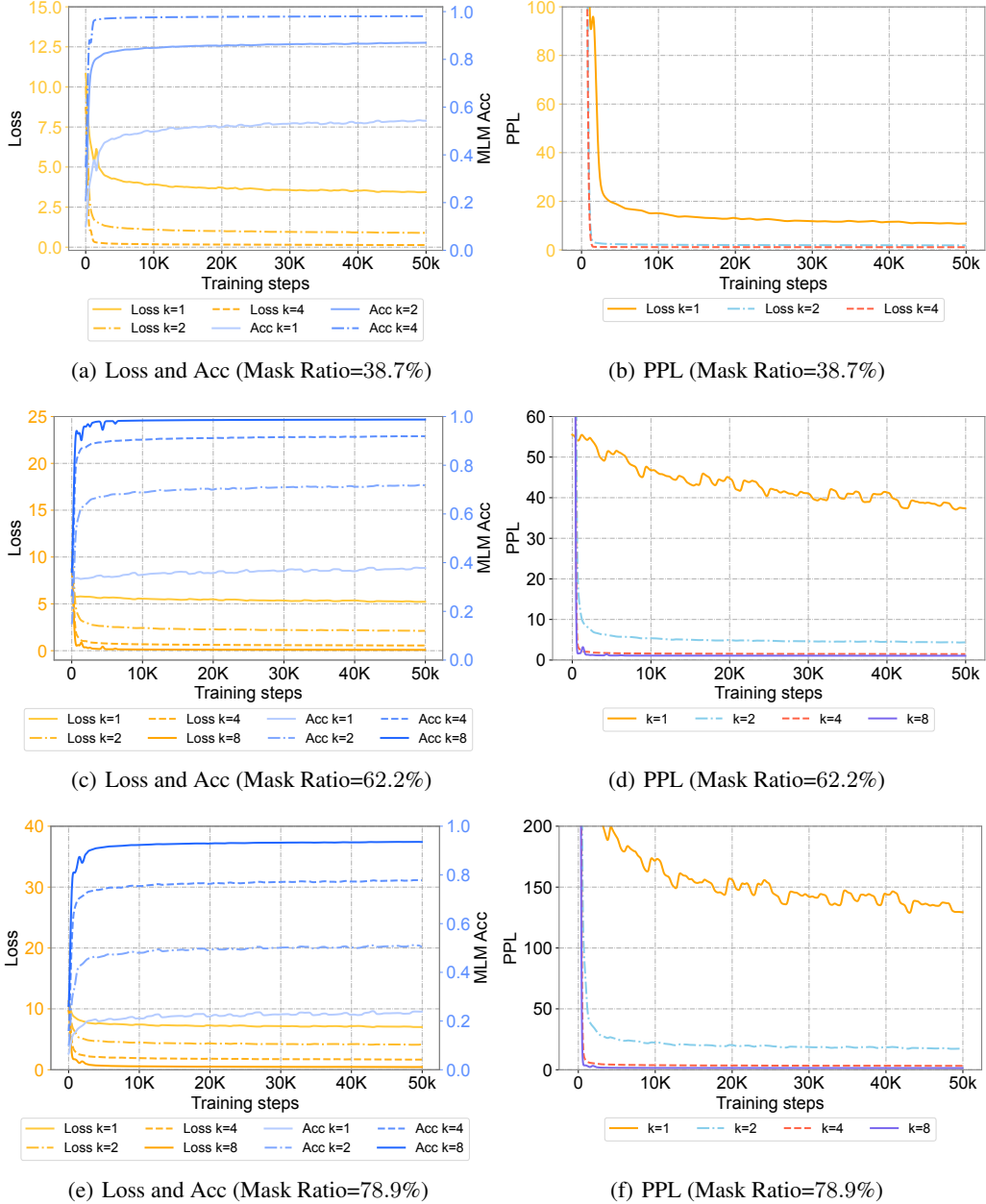
G. Training Curves in the *Repeated MLM* Experiments

(a) Loss and Acc (Mask Ratio=38.7%)

(b) PPL (Mask Ratio=38.7%)

(c) Loss and Acc (Mask Ratio=62.2%)

(d) PPL (Mask Ratio=62.2%)

(e) Loss and Acc (Mask Ratio=78.9%)

(f) PPL (Mask Ratio=78.9%)

Figure 9. The training loss, mask prediction accuracy and perplexity (PPL) curves of MLMs in the *Repeated MLM* experiments.

We plot the training curves of the MLMs with different repetition times k and mask ratios p in the *Repeated MLM* experiments, as shown in Figure 9. From this figure, we can observe that when we fix the mask ratio p and change the parameter k, as k increases (meaning more redundancy in the input), the intensity of semantics corruption in the context received by the model decreases. As a result, the model exhibits faster convergence, lower loss, lower perplexity and higher mask prediction accuracy during pre-training. This is also a key reason why model performance drops significantly when the semantics corruption in the input context are too low; at this point, the pre-training task becomes too simple, and the model converges quickly without learning meaningful information.

H. CUDA-Accelerated Dynamic Programming Framework: Details and Efficiency Analysis

Naively summing over all paths $\mathbf{A} \in \Gamma$ can be prohibitively expensive, since the number of possible paths grows exponentially. To solve this, we use a dynamic programming (DP) approach, which is also adopted by the training algorithms for connectionist temporal classification (Graves et al., 2006; Saharia et al., 2020) and DA-Transformer (Huang et al., 2022; Shao et al., 2022; Huang et al., 2023; 2024). We further adopted a CUDA-accelerated dynamic programming algorithm from DA-Transformer (Huang et al., 2022) to efficiently perform state alignment during pre-training. The core of the algorithm utilizes CUDA (Compute Unified Device Architecture) to parallelize the computations involved in dynamic programming for sequence modeling. The primary objective is to efficiently calculate forward (α) and backward (β) probabilities, which are crucial for evaluating sequence likelihoods and performing gradient-based optimization during model training. By leveraging the parallel processing capabilities of GPUs, the algorithm distributes computations across numerous threads and blocks, significantly reducing runtime compared to traditional CPU-based implementations.

H.1. Dynamic Programming Framework: Forward Probability (α) and Backward Probability (β)

The forward algorithm computes the forward probability matrix α, where each element $\alpha_{i,u}$ represents the probability of reaching state u at position i in the target sequence. The computation follows the recursive relation:

$$\alpha_{i,u} = \sum_{v < u} \left(\alpha_{i-1,v} \times \mathbf{E}_{v,u} \times \mathbf{P}_u(y_i) \right),$$

where v and u index the states, $\mathbf{E}_{v,u}$ is the transition score from state v to state u, $\mathbf{P}_u(y_i)$ is the emission probability of state u emitting the i-th token y_i.

CUDA Parallelization Strategy for the Forward Algorithm. Each GPU thread is assigned to compute $\alpha_{i,u}$ for specific states and positions, allowing multiple $\alpha_{i,u}$ values to be calculated concurrently. Warp-level optimizations, are employed to efficiently perform reductions when summing contributions from multiple states. Additionally, shared memory is utilized to store intermediate results, and synchronization queues coordinate computations across different segments of the sequence. This parallel approach enables the simultaneous computation of numerous $\alpha_{i,u}$ values, thereby accelerating the forward pass across all sequence positions and states.

The backward algorithm calculates the backward probability matrix β, where each element $\beta_{i,u}$ signifies the probability of transitioning from state u at position i to the end of the sequence. The recursive relation for β is defined as:

$$\beta_{i,u} = \sum_{v > u} \left(\beta_{i+1,v} \times \mathbf{E}_{u,v} \times \mathbf{P}_v(y_{i+1}) \right),$$

where v and u are state indices, $\mathbf{E}_{u,v}$ is the transition score from state u to state v, $\mathbf{P}_v(y_{i+1})$ is the emission probability of state v emitting the $(i+1)$-th token y_{i+1}.

CUDA Parallelization Strategy for the Backward Algorithm. Similar to the forward pass, GPU threads are allocated to compute $\beta_{i,u}$ for specific states and positions. The backward pass processes the sequence in reverse order, ensuring that $\beta_{i+1,v}$ values are computed before $\beta_{i,u}$ values that depend on them. Warp-level primitives facilitate efficient summation of probabilities from future states, while shared memory and synchronization mechanisms maintain data consistency across different sequence segments. This parallelization allows for the rapid aggregation of probabilities from multiple future states, effectively computing the entire β matrix.

H.2. Gradient Calculations

Gradient computation is essential for optimizing model parameters during training. The algorithm calculates gradients with respect to both emission probabilities $\mathbf{P}_u(y_i)$ and transition scores $\mathbf{E}_{v,u}$. In the following formulas, M denotes the final position in the target sequence (i.e., the last index along the target length dimension).

$$\nabla \mathbf{P}_u(y_i) = \frac{\alpha_{i,u} \times \beta_{i,u}}{\sum_u \alpha_{M,u}}$$

where $\sum_u \alpha_{M,u}$ represents the likelihood of the entire sequence, computed by summing the forward probabilities $\alpha_{M,u'}$ over all possible final states u' at the final position M. This gradient indicates how changes in the emission probability for state u at position i affect the log-likelihood of the sequence.

The transition gradient is given by:

$$\nabla \mathbf{E}_{v,u} = \frac{\alpha_{i-1,v} \times \beta_{i,u}}{\sum_u \alpha_{M,u}}$$

This gradient measures the effect of changes in the transition score between states v and u on the overall log-likelihood of the sequence.

CUDA Parallelization Strategy. For emission gradients, each thread independently computes $\nabla \mathbf{P}_u(y_i)$ for assigned states and positions, leveraging parallel execution to handle multiple computations simultaneously. For transition gradients, threads collaboratively compute $\nabla \mathbf{E}_{v,u}$ by aggregating contributions from various sequence positions using parallel reduction techniques. This parallel approach ensures efficient backpropagation through the dynamic programming steps, facilitating rapid parameter updates during training.

H.3. Time Complexity Analysis

Serial (Naïve) Time Complexity. In a sequential implementation, computing the forward and backward probabilities involves iterating over all batches, sequence positions, states, and transitions. The total number of operations scales as:

$$\mathcal{O} \left(\text{bsz} \times \text{tarlen} \times \text{prelen} \times \text{translen} \right),$$

where bsz is batch size, tarlen is target sequence length, prelen is the number of states, translen is the maximum number of transitions per state (usually equals to prelen).

CUDA Parallel Time Complexity. Under ideal parallel conditions, assuming the GPU can handle an effectively infinite number of parallel threads, which means that each sequence position can be processed in parallel during the forward and backward passes, reducing the time complexity to $\mathcal{O} \left(\text{tarlen} \right)$, since each position i can be computed independently once the necessary dependencies are met. The gradient computations similarly benefit from parallel execution, allowing gradients for different states and transitions to be calculated concurrently.

While the theoretical parallel time complexity suggests $\mathcal{O}(\text{tarlen})$, actual performance is influenced by GPU resource constraints, such as the number of available threads, memory bandwidth, and synchronization overhead. In practice, the wall-clock time is significantly reduced compared to the serial case, though it may scale slightly worse than $\mathcal{O}(\text{tarlen})$ due to these hardware and implementation factors.

H.4. Space Complexity Analysis

The algorithm's space requirements are primarily determined by the storage of the α and β matrices, along with auxiliary tensors used for transition scores. The overall space complexity is:

$$\mathcal{O} \left(\text{bsz} \times \text{prelen} \times \left(\text{tarlen} + \text{translen} \right) \right),$$

where bsz is batch size, tarlen is target sequence length, prelen is the number of states, translen is the maximum number of transitions per state (usually equals to prelen).

This accounts for the forward and backward probability matrices and the transition scores necessary for computations.

Real-World Analysis of Algorithm Space Usage. In practical scenarios, the space requirements of the CUDA-accelerated dynamic programming algorithm remain manageable, even for large-scale tasks such as masked sequence modeling for text. For instance, consider a typical setup with a batch size of 64 per GPU, a maximum sequence length of 512 tokens, and a masking ratio of 0.15, resulting in approximately 77 `[MASK]` tokens per sequence. If each `[MASK]` token can be expanded

to up to 4 hidden states ($k = 4$), the prelen (number of states) becomes approximately $512 \times 0.15 \times 4 \approx 308$. Similarly, tarlen (the target sequence length) is approximately 77, while translen (the number of transitions per state) is set to 308, matching prelen. Under these conditions, tensors such as α and β, each of size [bsz, tarlen, prelen], would require around $64 \times 77 \times 308 \approx 1.52 \times 10^6$ entries, equivalent to 5.8 MB per tensor for 32-bit floats. The transition links tensor, of size [bsz, prelen, translen], would require approximately $64 \times 308 \times 308 \approx 6.07 \times 10^6$ entries, or about 23.2 MB of memory for 32-bit floats. Altogether, these tensors occupy approximately 34.8 MB of memory. Even with these allocations, the memory usage is well within the 80 GB available on an NVIDIA A100 GPU, leaving ample room for model parameters, activations, and framework overheads. **This demonstrates that the algorithm's $\mathcal{O}(\mathbf{bsz} \times \mathbf{prelen} \times (\mathbf{tarlen} + \mathbf{translen}))$ space complexity poses no significant constraint in real-world training setups**.

States alignment algorithm does not affect the scalability of the model. The analysis of the time and space complexity of the states alignment algorithm reveals that its complexity is independent of the model size (e.g., the number of layers or the embedding dimension) and depends only on the input sequence length. Therefore, with the training data unchanged, states alignment algorithm does not introduce additional training overhead as the model size increases.

I. Hyper-Parameter Configuration for Molecular Pre-training

We implement ExLM using 9 stacked Transformer layers, each with 12 attention heads. The model dimension and feedforward dimension of each Transformer layer are 768 and 2,048, respectively. The total number of ExLM's parameters achieves 50.5M. We use Adam (Kingma & Ba, 2014) optimizer and polynomial learning rate scheduler to train ExLM, and we set the learning rate as 5e-4 and warmup stesp as 10K. The total training steps are 120K and each batch has 64k tokens at maximum. We implement the ExLM model using the Fairseq library [2] and train ExLM on two RTX3090 GPUs for about 24 hours.

For more pre-training hyper-parameters, please refer to Table 7.

Table 7. ExLM hyper-parameters for molecular pre-training.

Hyper-parameters	Value
Learning rate	5e-4
LR scheduler	polynomial_decay
Num of expanded states	2
Warmup updates	10K
Max updates	120K
Max tokens	64K
FFN dropout	0.1
Attention dropout	0.1
Activation dropout	0
Num of layers	9
Num of attention heads	12
Encoder embedding dim	768
Encoder FFN dim	$2,048$
Adam (β_1, β_2)	$(0.9, 0.98)$
Fragments Drop ratio	0.15
Vocabulary size	369
Activation function	GELU
Weight Decay	0.0
Clip Norm	1.0

J. Hyper-Parameter Configuration for Molecular Fine-tuning

In different downstream task, we use different hyper-parameters. We run each task three times using three different random seeds, and take the average performance of these three runs as the final result. For detailed fine-tuning hyper-parameters, please refer to Table 8.

Table 8. ExLM hyper-parameters for molecular fine-tuning.

Tasks	Epochs	Batch size	Learning rate	Warmup Ratio	Dropout	Pooler-dropout
BACE	60	64	1e-4	0.06	0.1	0.2
BBBP	40	128	4e-4	0.06	0.1	0.1
TOX21	80	128	1e-4	0.06	0.1	0.1
SIDER	100	32	5e-4	0.4	0.1	0
MUV	40	128	2e-5	0.2	0.1	0.1
ClinTox	100	256	5e-5	0.1	0.1	0.5
ToxCast	80	64	1e-4	0.06	0.1	0.1

---

[2]https://fairseq.readthedocs.io/en/latest/

K. Details of Molecular Fine-tuning Datasets and Baselines

Datasets. We perform a comprehensive set of experiments on the MoleculeNet(Wu et al., 2018) benchmark, focusing on the molecular property prediction task. MoleculeNet has emerged as one of the most widely recognized and utilized benchmarks in the field of molecular property prediction, providing a standardized platform for evaluating machine learning models' performances on evaluating molecular properties. Its datasets encompass a broad range of molecular tasks, and address diverse and practical scientific problems such as drug discovery, toxicity prediction and so on.

In this section, we provide a detailed summary of the statistics and fundamental characteristics of the MoleculeNet datasets we use in Table 9. This table offers information about the dataset sizes, task types, and compositions, providing readers with essential background information to better understand the experimental setup and subsequent analysis.

Baselines. We evaluate our approach against various supervised learning and pre-training baselines, including both SMILES-based and 3D molecular pre-trained models. The supervised methods include D-MPNN (Yang et al., 2019) and AttentiveFP (Xiong et al., 2019), both of which are based on graph neural networks (GNNs). For 2D and 3D molecular pre-training, we consider baseline methods: N-gram (Liu et al., 2019), GROVER (Rong et al., 2020), GraphMVP (Liu et al., 2021), Mole-BERT (Xia et al., 2023), and 3D InfoMax (Stärk et al., 2022). For a fair comparison, we train a SMILES model based on MLM pre-training, referred to as SMILES-BERT, using the same training data, model architecture, and training hyperparameters as ExLM.

Table 9. Summary information of the MoleculeNet benchmark datasets.

Dataset	Tasks	Task type	Molecules (train/valid/test)	Describe
BACE	1	Classification	1,210/151/151	Binding results of human BACE-1 inhibitors
BBBP	1	Classification	1,631/204/204	Blood-brain barrier penetration
ClinTox	2	Multi-label classification	1,182/148/148	Clinical trial toxicity and FDA approval status
Tox21	12	Multi-label classification	6,264/783/783	Qualitative toxicity measurements
ToxCast	617	Multi-label classification	6,860/858/858	Toxicology data based on in vitro screening
SIDER	27	Multi-label classification	1,141/143/143	Adverse drug reactions to the 27 systemic organs
MUV	17	Multi-label classification	74,469/9,309/9,309	A subset of PubChem BioAssay

L. Details of the GLUE Benchmark

Below are detailed descriptions of all the GLUE tasks:

MNLI (Multi-Genre Natural Language Inference). The MNLI dataset (Williams et al., 2018) consists of 393K training examples gathered through crowdsourcing from various genres. The task requires predicting whether a premise sentence entails, contradicts, or is neutral with respect to a given hypothesis sentence.

QQP (Quora Question Pairs). QQP (Shankar et al., 2017) includes 364K training examples sourced from the Quora question-answering platform. The objective is to determine if two provided questions are semantically equivalent.

QNLI (Question Natural Language Inference). Derived from the Stanford Question Answering Dataset (SQuAD) (Rajpurkar et al., 2018), QNLI comprises 108K training examples. The task involves predicting whether a sentence contains the answer to a specific question.

SST-2 (Stanford Sentiment Treebank). SST-2 (Socher et al., 2013) contains 67K training examples based on movie reviews with human-annotated sentiments. The goal is to classify each sentence as expressing either positive or negative sentiment.

CoLA (Corpus of Linguistic Acceptability). The CoLA dataset (Warstadt et al., 2019) includes 8.K training examples extracted from books and journal articles focused on linguistic theory. The task is to assess whether a given sentence is linguistically acceptable.

RTE (Recognizing Textual Entailment). RTE (Bentivogli et al., 2009; Dagan et al., 2005; Haim et al., 2006; Giampiccolo et al., 2007) encompasses 2.5K training examples derived from textual entailment challenges. The objective is to determine if a premise sentence entails a provided hypothesis sentence.

MRPC (Microsoft Research Paraphrase Corpus). MRPC (Dolan & Brockett, 2005) consists of 3.7K training examples collected from various news sources. The task is to predict whether two given sentences are semantically equivalent.

STS-B (Semantic Textual Similarity Benchmark). STS-B (Cer et al., 2017) includes 5.8K training examples sourced from multiple origins, annotated by humans for sentence pair semantic similarity. The task requires predicting the degree of semantic similarity between two sentences on a scale from 1 to 5.

We use Spearman correlation for STS, Matthews correlation for CoLA, and accuracy for MNLI, QNLI, RTE and SST-2 as the metrics on GLUE.

M. Hyper-Parameter Configuration for Textual Pre-training

We implement EXLM in two configurations: a base model and a large model. The base EXLM consists of 12 stacked Transformer layers, each with 12 attention heads. The model dimension and feedforward dimension of each Transformer layer are 768 and 3,072, respectively, resulting in a total of 128M parameters. The large EXLM model uses 24 Transformer layers with 16 attention heads per layer. The model dimension and feedforward dimension are increased to $1,024$ and $4,096$, respectively, with a total parameter count of 361M. We use Adam (Kingma & Ba, 2014) optimizer and polynomial learning rate scheduler to train EXLM, and we set the learning rate as 5e-4 and warmup steps as 10K. The total training steps are 125K and each batch has 2048 samples at maximum. We also implement the EXLM model using the Fairseq library. For more pre-training hyper-parameters, please refer to Table 10.

Table 10. EXLM hyper-parameters for textual pre-training.

Hyper-parameters	Value
Learning rate	5e-4
LR scheduler	polynomial_decay
Num of expanded states	4
Warmup updates	10K
Max updates	125K
Batch size	$2,048$
FFN dropout	0.1
Attention dropout	0.1
Activation dropout	0
Num of layers	*base*: 12, *large*: 24
Num of attention heads	*base*: 12, *large*: 16
Encoder embedding dim	*base*: 768, *large*: $1,024$
Encoder FFN dim	*base*: $3,072$, *large*: $4,096$
Adam (β_1, β_2)	$(0.9, 0.98)$
Mask ratio	0.15
Activation function	GELU
Weight Decay	0.01
Clip Norm	0.0

N. Hyper-Parameter Configuration for Textual Fine-tuning

We apply grid search for both the GLUE and SQuAD 2.0 datasets, and the grid search hyperparameters are shown in Table 11.

Table 11. Grid search hyperparameters for the GLUE and SQuAD 2.0 tasks.

Hyperparameter	MNLI, QNLI, QQP, SST-2
Peak learning rate	{1e-5, 2e-5, 3e-5, 4e-5}
Batch size	32
Max epochs	{2, 3, 5}
Warm-Up Proportion	6%
	RTE, MRPC, CoLA, STS-B
Peak learning rate	{2e-5, 3e-5, 4e-5, 5e-5}
Batch size	{16, 32}
Max epochs	{2, 3, 5, 10}
Warm-Up Proportion	{6%, 10%}
	SQuAD 2.0
Peak learning rate	{2e-5, 3e-5, 4e-5, 5e-5}
Batch size	{16, 32}
Max epochs	{2, 3}
Warm-Up Proportion	{6%, 10%}

O. Detailed Case Study

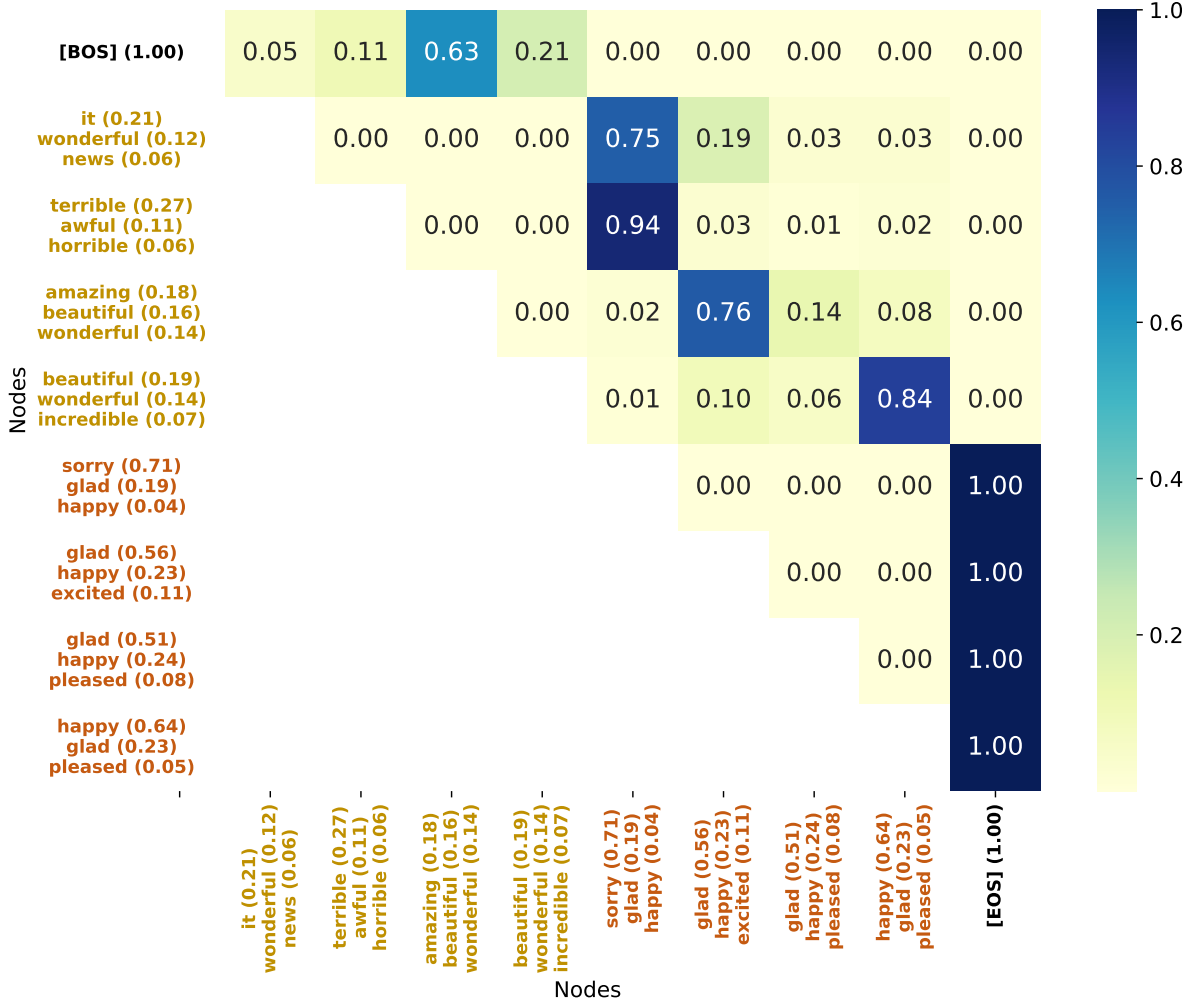| Nodes	it (0.21) / wonderful (0.12) / news (0.06)	terrible (0.27) / awful (0.11) / horrible (0.06)	amazing (0.18) / beautiful (0.16) / wonderful (0.14)	beautiful (0.19) / wonderful (0.14) / incredible (0.07)	sorry (0.71) / glad (0.19) / happy (0.04)	glad (0.56) / happy (0.23) / excited (0.11)	glad (0.51) / happy (0.24) / pleased (0.08)	happy (0.64) / glad (0.23) / pleased (0.05)	[EOS] (1.00)
[BOS] (1.00)	0.05	0.11	0.63	0.21	0.00	0.00	0.00	0.00	0.00
it (0.21) / wonderful (0.12) / news (0.06)		0.00	0.00	0.00	0.75	0.19	0.03	0.03	0.00
terrible (0.27) / awful (0.11) / horrible (0.06)			0.00	0.00	0.94	0.03	0.01	0.02	0.00
amazing (0.18) / beautiful (0.16) / wonderful (0.14)				0.00	0.02	0.76	0.14	0.08	0.00
beautiful (0.19) / wonderful (0.14) / incredible (0.07)					0.01	0.10	0.06	0.84	0.00
sorry (0.71) / glad (0.19) / happy (0.04)						0.00	0.00	0.00	1.00
glad (0.56) / happy (0.23) / excited (0.11)							0.00	0.00	1.00
glad (0.51) / happy (0.24) / pleased (0.08)								0.00	1.00
happy (0.64) / glad (0.23) / pleased (0.05)									1.00

Figure 10. We visualize the model's predictions when the input is "*This is* [MASK]*, and I'm very* [MASK] *to see this.*" ($k = 4$). The **yellow nodes** represent the expanded states corresponding to the first [MASK] token, while the **brown nodes** represent the expanded states corresponding to the second [MASK] token. The x-axis and y-axis show the top-3 predicted word for each graph node.

We further provide a visualization of the model's predictions when the input is *This is* [MASK]*, and I'm very* [MASK] *to see this.*, as shown in Figure 10. Specifically, since the input contains two [MASK] tokens and $k = 4$, there are 8 nodes in the graph. For implementation convenience, we introduce two special nodes, [BOS] and [EOS], which correspond to the starting and ending nodes of the graph. We require that all paths in the graph must start from the [BOS] node and terminate at the [EOS] node, so when searching for a valid path in the graph, we can directly start from the [BOS] node without enumerating all possible starting nodes. In this graph, besides the heatmap representing the transition probability (i.e., the edge weight between two nodes), we also display the top 3 most probable words decoded from each graph node. From these results, we observe the following characteristics of ExLM:

- **ExLM successfully represents the different choices of each [MASK] token using distinct expanded states.** Since $k = 4$, the model predicts four different choices for both [MASK] tokens in the input. These four choices typically represent different aspects of the semantics. For example, the first [MASK] token's choices include three categories: a noun (e.g., it, news), negative emotion words (e.g., terrible, awful), and positive emotion words (e.g., amazing, beautiful, wonderful), each represented by a different expanded state. This effectively avoids semantic ambiguity in the

context, i.e., the semantic multimodality. Similarly, the second [MASK] token's choices include positive emotions (e.g., glad, happy) and negative emotions (e.g., sorry), and these different types of semantic information are also represented by distinct expanded states.

- **ExLM successfully captures the semantic relationships between different `[MASK]` tokens.** The transition matrix displayed in Figure 10 shows an important phenomenon: the degree of dependency between different states in ExLM (i.e., the edge weight between two nodes) is strongly correlated with the semantic similarity between them. For instance, when the first [MASK] token is "terrible," the second [MASK] token has a higher probability of choosing "sorry." Conversely, when the first [MASK] token is a more positive emotion word, the second [MASK] token is more likely to choose "glad," which also represents positive emotion. This demonstrates that ExLM can effectively learn the semantic dependencies between [MASK] tokens. We further visualize this as a directed acyclic graph (DAG) in Figure 11, where the edge weight between nodes is directly related to the semantic dependency between them, further validating the effectiveness of ExLM.

- **ExLM typically has lower uncertainty for `[MASK]` tokens that appear later**. As shown in Figure 10, the word probability distributions of the four expanded states corresponding to the **first `[MASK]` token** are often more evenly distributed, meaning that the differences of word probabilities across different words in each state are not very large. However, for the four expanded states corresponding to the **second `[MASK]` token**, the model usually predicts one word with a probability significantly higher than the others. This phenomenon occurs due to the directionality of the DAG. Specifically, in ExLM's DAG, transitions only occur from earlier states to later states, and the choices of earlier states have a greater impact on the final result, which means that the earlier states tend to have higher uncertainty.
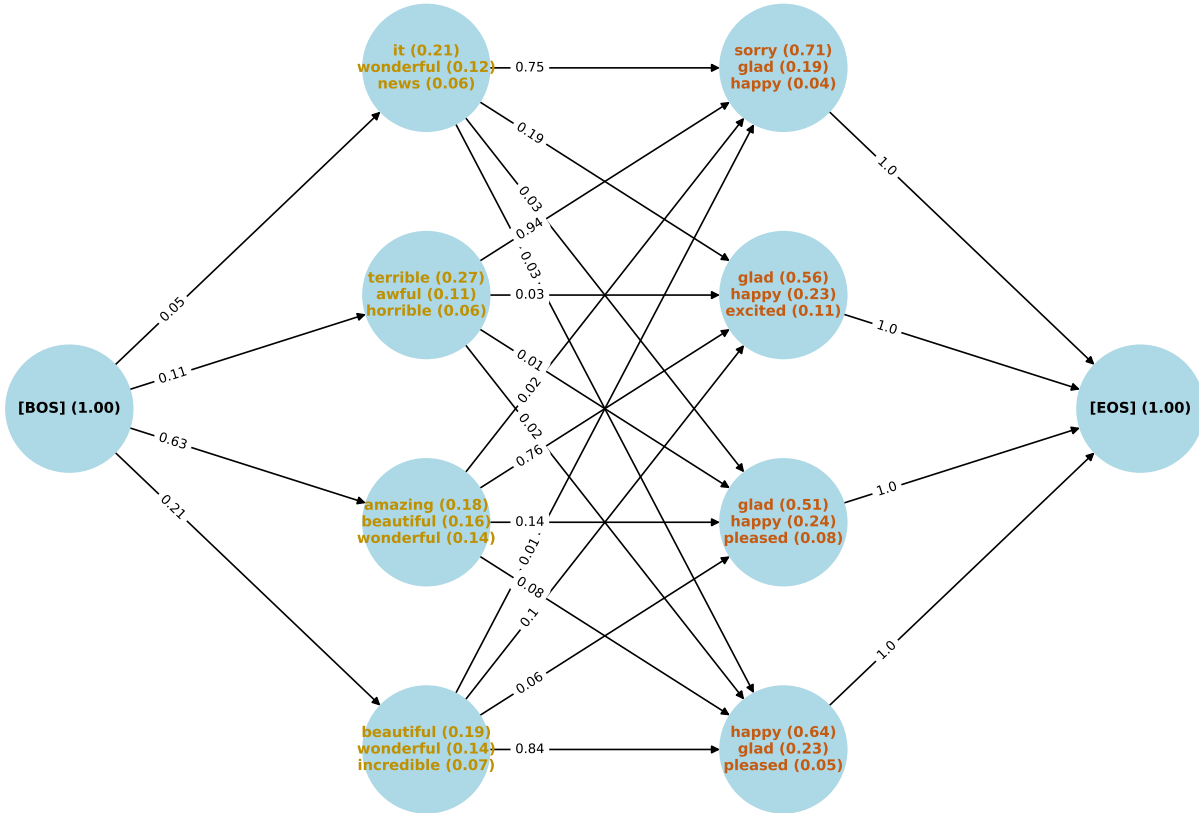

Figure 11. The visualization of the DAG from ExLM. It shows that the edge weight between different nodes is directly related to the semantic dependency between those nodes.

P. More Analytical Experiments of ExLM

P.1. Entropy Analysis on ExLM

We conduct an entropy analysis to compare the severity of the semantic multimodality under different mask ratios p in the input context for ExLM and MLM. The results are shown in Figure 12. The results reveal that ExLM has significantly lower entropy in its prediction distribution than MLM, indicating lower uncertainty in dealing with ambiguous contexts. This further proves that ExLM significantly mitigates the negative impact of semantic multimodality and can still produce relatively confident predictions even when contextual semantic corruption is severe. Additionally, as k increases, the uncertainty of ExLM decreases, suggesting that increasing k expands the model's semantic space, helping to alleviate the negative effects of the semantic multimodality. Moreover, under higher mask ratios, ExLM shows a more noticeable reduction in uncertainty compared to MLM, explaining why ExLM still maintains relatively strong performance under high mask ratios.
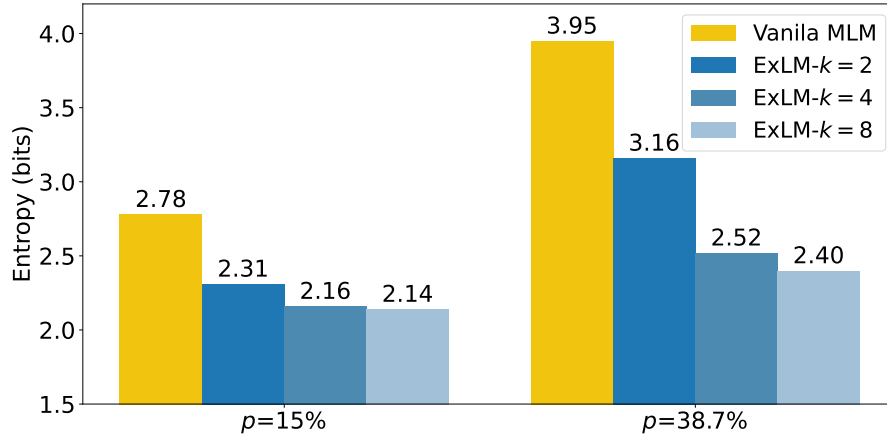

Figure 12. Entropy analysis of ExLM with different mask ratios p and numbers of expanded states k. ExLM demonstrates significantly lower uncertainty than MLM, and a larger k further reduces the ExLM's uncertainty, enhancing its better modeling capability.

P.2. Training Efficiency of ExLM

We compare the training efficiency differences between MLM and ExLM ($k = 4$) with the same pre-training configuration. Both models are trained on two Tesla A100 80G GPUs under the hyperparameters from Table 5, and their training cost statistics are shown in Table 12. From this table, we can know that the training cost of ExLM ($k = 4$) is approximately 1.9 times that of the MLM with the same pre-training configuration. To compare the performance differences between MLM and ExLM ($k = 4$) under equal training costs, we increase the training steps of MLM from 50,000 to 100,000, and refer to this model as Vanilla MLM++. The performance of Vanilla MLM++ is shown in Table 3, where we can see that although increasing the training steps significantly improves MLM performance, it still remains lower than that of ExLM. This further demonstrates that ExLM achieves stronger performance under the same training cost.

Table 12. Training time comparison (Tesla A100 80G GPUs) between ExLM ($k = 4$) and MLM.

	MLM	ExLM ($k = 4$)
GPU Time (Hours)	54.7	104.2

P.3. Performance of Sparse ExLM

We also implement a version of ExLM using a Sparse DAG (Sparse ExLM) and evaluate its performance on various downstream tasks, with the results shown in Table 13. Specifically, the key difference between the Sparse DAG and the original DAG in ExLM is that, in the Sparse DAG, we do not allow transitions between expanded states that belong to the same `[MASK]` token. This significantly reduces the number of edges in the DAG, making the graph more sparse.

However, as shown in Table 13, ExLM with the Sparse DAG experiences some performance degradation. While it still outperforms the version of ExLM without a transition matrix (ExLM w/o Transitions), it falls behind the original ExLM. The reason for this decline is that, in the Sparse DAG, only one expanded state can be selected for each `[MASK]` token, which prevents the model from capturing longer but possible outputs. For example, in the sentence "*This is `[MASK]`, and I'm very `[MASK]` to see this*," both "*so good*" and "*happy*" are reasonable interpretations for the first and second `[MASK]` tokens, respectively. However, the Sparse DAG cannot model this ambiguity, as it does not allow multiple expanded states being chosen for a single `[MASK]` token. This limitation significantly restricts the model's ability to capture rich semantic variations, leading to a performance drop.

Table 13. **Performance of Sparse ExLM.** Using a sparser DAG in ExLM leads to a performance decline.

Method	MNLI ↑	QNLI↑	QQP↑	RTE↑	Avg ↑
ExLM w/o Transitions	83.8	90.9	91.1	55.6	80.4
ExLM w/ Sparse DAG	84.4	91.2	91.3	56.9	81.0
Vanilla MLM	83.6	90.0	90.3	54.7	79.6
ExLM	**85.1**	**91.4**	**91.3**	**57.6**	**81.4**