On a Benefit of Masked Language Model Pretraining:
Robustness to Simplicity Bias

Anonymous ACL submission

Abstract

Despite the success of pretrained masked language models (MLM), why MLM pretraining is useful is still a question not fully answered. In this work we theoretically and empirically show that MLM pretraining makes models robust to lexicon-level spurious features, partly answering the question. Our explanation is that MLM pretraining may alleviate problems brought by simplicity bias (Shah et al., 2020), which refers to the phenomenon that a deep model tends to rely excessively on simple features. In NLP tasks, those simple features could be token-level features whose spurious association with the label can be learned easily. We show that MLM pretraining makes learning from the context easier. Thus, pretrained models are less likely to rely excessively on a single token. We also explore the theoretical explanations of MLM’s efficacy in causal settings. Compared with Wei et al. (2021), we achieve similar results with milder assumption. Finally, we close the gap between our theories and real-world practices by conducting experiments on real-world tasks.

1 Introduction

The question ‘why is masked language model (MLM) pretraining (Devlin et al., 2019; Liu et al., 2019) useful?’ has not been totally answered. In this work, as an initial step toward the answer, we show and explain that MLM pretraining makes the model robust to lexicon-level features that are spuriously associated with the target label. It gives the model a better generalization capability under distribution shift.

Previous studies have empirically shown the robustness of MLM pretrained models. Hao et al. (2019) show that MLM pretraining leads to wider optima and better generalization capability. Hendrycks et al. (2020) and Tu et al. (2020) show that pretrained models are more robust to out-of-distribution data and spurious features. However, it remains unanswered why pretrained models are more robust.

We conjecture that models trained from scratch suffer from the pitfall of simplicity bias (Shah et al., 2020) (Figure 1). Shah et al. (2020) and Kalimeris et al. (2019) showed that deep networks tend to converge to a simple decision boundary that involves only a few features. The networks may not utilize all the features and thus may not maximize the margin, which results in worse robustness. A consequence of this could be that a model may excessively rely on a feature that has spurious association with the label and ignore the other features that are more robust. In the studies of Shah et al. (2020) and Kalimeris et al. (2019), they investigated networks with continuous input. Lovering et al. (2021) discovered similar results on synthetic NLP tasks, where the inputs are discrete. We will further explore this discrete setting in this work.

We start the exploration with the following assumptions: Let the sentence, label pair be \(X, Y\).

Assumption 1. We assume that from \(X\), we can extract two features \(X_1\) and \(X_2\).

Assumption 2. \(X_1\) is a spurious feature that has strong association with \(Y\). Specifically, it means that, solely relying on \(X_1\), one can predict with high accuracy, but cannot be 100% correctly.

Assumption 3. \(X_2\) is a robust feature based
on which $Y$ can be predicted with 100% accuracy. Namely, there exists a deterministic mapping $f_{X_2 \rightarrow Y}$ that maps $X_2$ to $Y$.

The assumptions above are realistic in some NLP tasks. In NLP tasks, the input $X$ is a sequence of tokens. Some tasks satisfy Assumption 1: $X$ can be decomposed into $X_1$ and $X_2$, where $X_1$ is the presence of certain tokens, and $X_2$ is the context of the token. Thus, $X_2$ has a much higher dimensionality than $X_1$. As shown by the analysis of Gardner et al. (2021), there are indeed datasets where Assumption 2 is true. However, if Assumption 3 is true, we would desire the model to rely on $X_2$, which contains the semantic of the input $X$.

With these assumptions, in Section 2 we empirically demonstrate that spurious features in discrete inputs can cause problems as in the continuous cases (Shah et al., 2020; Kalimeris et al., 2019). We show that, possibly due to the simplicity bias, a deep model is likely to excessively rely on $X_1$ and to rely on $X_2$ less. In Section 3.1 and Section 3.2 we provide a theoretical explanation of how MLM pretraining makes a model robust to spurious features. Let $\Pi^{(1)}$ be the conditional probability $P(X_1|X_2)$. We show (1) the relation between the mutual information $I(\Pi^{(1)}; Y) \geq I(X_1; Y)$ and (2) the convergence rate of learning from $\Pi^{(1)}$ is of the same order as learning from $X_1$. That is, when the MLM model can perfectly model the probability $P(X_1|X_2)$ and thus generate perfect $\Pi^{(1)}$, learning from $\Pi^{(1)}$ is as easy as learning from $X_1$. As a result, the model will be more likely to rely on $\Pi^{(1)}$. Since $\Pi^{(1)}$ is estimated based on $X_2$, higher reliance on $\Pi^{(1)}$ also implies higher reliance on the robust feature $X_2$. This avoids the pitfall of simplicity bias that the model relies excessively on $X_2$. To relax Assumption 3, we make one step further by considering causal settings in Section 3.3.

The above results partly explain why MLM pretraining is useful for NLP. Denote a sequence of tokens as $X = \langle X_1, X_2, \ldots, X_L \rangle$. During the MLM pretraining process, each token is masked randomly at certain probability, and the training objective is to predict the masked tokens with the maximum likelihood loss. As a result, the model is capable of estimating the conditional probability $P(X_i|X_i \backslash X_2)$ for all $i = 1, 2, \ldots, L$. Even though which of the tokens is spurious is unknown, as long as the spurious token has non-zero probability to be masked during pretraining, MLM can estimate its distribution conditioned on the context and thus can reduce the reliance on it.

Finally, we close the gap between our theories and reality. One major gap is that, in reality, we do not use the conditional probability for downstream tasks. Instead, we feed the input $X$ without masking any token and fine-tune the model along with a shallow layer over its output. Regardless of that, we hypothesize that the robustness brought by MLM pretraining still exists. To prove that, in Section 4 we use the toy example and verify the effect of MLM pretraining when using the common practice for fine-tuning. In Section 5, we validate our theories with two real-world NLP tasks.

To sum up, our study leads to new research directions. Firstly, we provide a new explanation of MLM pretraining’s efficacy. Unlike the previous purely theoretical studies (Saunshi et al., 2021; Wei et al., 2021), our assumptions are milder and more realistic. Secondly, we study NLP robustness from the perspective of self-supervised model. Since self-supervised trained embeddings have been widely used since Word2vec (Mikolov et al., 2013), and it is indispensable to the generalization to unseen data, a better understanding of self-supervised embeddings should be important for future research.

## 2 A Toy Example

To show that spurious association can cause difficulty of convergence, we construct a toy example with variables $X_1, X_2, Y$ that satisfy the assumptions. We make $X_1$ depends only on $X_2$, so it is not a caustic feature of $Y$. Let the dimension of the random variables $X_1$ and $X_2$ be $2$ and $d_2$ respectively. Their value $x_1 \in X_1 = \{e_1, e_2\}$ and $x_2 \in X_2 = \{e_1, \ldots, e_{d_2}\}$, where $e_i$ is the one-hot vector whose $i$th element is $1$. We control the strength of the association between $X_1$ and $Y$ with $\nu < 0.5$, making $X_1 = Y$ with probability $1 - \nu$. Specifically, denote with $X_2$ the middle $2\nu d_2$ dimensions of $X_2$, i.e. the $[d_2/2 - \nu d_2]$th to the $[d_2/2 + \nu d_2]$th elements in $X_2$. We consider the following random process:

\[
\begin{align*}
X_2 &= e_i, i \sim \text{Uniform}(1, d_2) \\
Y &= \begin{cases} -1 & \text{if } X_2 = e_i \text{ for some } i < d_2/2 \\
+1 & \text{otherwise} \end{cases} \\
X_1 &= \begin{cases} e_i, i \sim \text{Uniform}(1, 2) & \text{If } \hat{x}_2 \neq 0 \\
f(X_2) & \text{Otherwise} \end{cases}
\end{align*}
\]
Table 1: The number of iterations a model w/ or w/o pretraining requires to converge. The number is the average of 25 runs with different random seeds, and the number in parentheses is the standard deviation.

<table>
<thead>
<tr>
<th>$d_2$</th>
<th>$\nu$</th>
<th>1 layer</th>
<th>2 layers</th>
<th>3 layers</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>w/o</td>
<td>w/o pre</td>
<td>w/ pre</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>50</td>
<td>0.04</td>
<td>3680 (189.5)</td>
<td>691 (55.8)</td>
<td>614 (169.1)</td>
</tr>
<tr>
<td></td>
<td>0.10</td>
<td>2664 (121.2)</td>
<td>530 (30.6)</td>
<td>441 (134.9)</td>
</tr>
<tr>
<td></td>
<td>0.25</td>
<td>1420 (96.0)</td>
<td>352 (23.8)</td>
<td>300 (62.0)</td>
</tr>
<tr>
<td></td>
<td>0.50</td>
<td>306 (79.8)</td>
<td>141 (40.7)</td>
<td>118 (33.4)</td>
</tr>
<tr>
<td>100</td>
<td>0.04</td>
<td>5466 (170.1)</td>
<td>945 (57.2)</td>
<td>689 (225.3)</td>
</tr>
<tr>
<td></td>
<td>0.10</td>
<td>3789 (99.2)</td>
<td>677 (32.2)</td>
<td>478 (142.9)</td>
</tr>
<tr>
<td></td>
<td>0.25</td>
<td>1952 (64.9)</td>
<td>428 (13.1)</td>
<td>330 (85.0)</td>
</tr>
<tr>
<td></td>
<td>0.50</td>
<td>330 (78.0)</td>
<td>156 (34.0)</td>
<td>133 (41.2)</td>
</tr>
<tr>
<td>500</td>
<td>0.04</td>
<td>11127 (265.9)</td>
<td>1953 (112.5)</td>
<td>857 (442.6)</td>
</tr>
<tr>
<td></td>
<td>0.10</td>
<td>7912 (169.2)</td>
<td>1279 (67.5)</td>
<td>657 (234.9)</td>
</tr>
<tr>
<td></td>
<td>0.25</td>
<td>4321 (152.3)</td>
<td>772 (35.5)</td>
<td>501 (133.5)</td>
</tr>
<tr>
<td></td>
<td>0.50</td>
<td>576 (150.0)</td>
<td>392 (70.2)</td>
<td>407 (81.1)</td>
</tr>
</tbody>
</table>

where $f(X_2) = e_1$ if $X_2 = e_i$ for some $i < d_2/2$, and $f(X_2) = e_2$ otherwise. In this way, predicting $Y$ solely based on the spurious feature $X_1$ can achieve accuracy $1 - \nu$.

We conduct experiments to inspect the effect of the strength of spurious association between $X_1$ and $Y$. We train linear networks by drawing batches of i.i.d. $([X_1; X_2], Y)$ pairs from the random process defined in 1. We use Adam optimization with learning rate 0.001 and the cross-entropy loss. In addition to single-layer linear networks, we also try over-parameterized 2-layer and 3-layer linear networks. The hidden size is $[10, 32]$. Since it is a linear separable problem, we can check whether the learned weight can lead to 100% accuracy in the defined distribution. We check it every 25 iterations. We say a model has converged if it is 100% accurate for 5 consecutive checks. We report the number of the iterations required before it converges for different $\nu$ and $d_2$.

Even though it is a linear-separable convex optimization problem, our results in Table 1 show that the strength of the spurious association can impact the number of iterations required to converge. We observe that when $\nu < 0.5$, the models tend to be trapped by the spurious feature, starting at accuracy $1 - \nu$ for iterations. When the spurious relation between $X_1$ and $Y$ is stronger, i.e. $\nu$ is smaller, the number of iterations required to converge is larger. In addition, the number of iterations is also larger when the $d_2$ is larger. An intuitive explanation is that the learning signal from $X_2$ is more sparse when $d_2$ is larger.

3 A Theoretical Explanation of the Efficacy of MLM Pretraining

3.1 $P(X_1|X_2)$ is More Informative Than $X_2$

The toy example above motivates us to consider the information contained in $P(X_1|X_2)$. In the toy example, when predicting $P(Y = 0|X)$, if we simply output $P(X_1 = e_1|X_2)$, then the accuracy of our prediction of $Y$ will be as high as predicting $Y$ solely based on $X_1$. It motivates us to inspect the reliability of the estimated $P(X_1|X_2)$ as a feature for the prediction of $Y$ compared to $X_1$. Let $\Pi_1 = P(X_1|X_2)|_{X_2}$ be a $|X_1|$-dimensional random variable. We can prove that when $P(X_1|X_2)$ is estimated perfectly, $\Pi_1$ is at least as informative as $X_1$. We will omit the subscript of $\Pi_1$ when there is no ambiguity.

Lemma 1. When $X_1$, $X_2$ are discrete, if $\Pi_1$ perfect, namely $\Pi_1 = P(\cdot|X_2)$, then the mutual information $I(X_1; \Pi_1) = I(X_1; X_2)$. (Proof: Appendix A.1)

Compared to previous works (Hjelm et al., 2019; Belghazi et al., 2018; Oord et al., 2018) that show some self-supervised training objectives are lower bounds of the mutual information $I(X_1; X_2)$, we directly show that the output of the MLM, $\Pi$, maximizes the mutual information, since $I(X_1; f(X_2)) \leq I(X_1; X_2)$ for any $f$. Moreover, instead of explaining the efficacy of pretraining.

\(1\)Uniform($a$, $b$) is the uniform distribution over $\{n\}^a_{n=a}$.
with the infomax principle (Linsker, 1988; Bell and Sejnowski, 1995), our theories below provide a different perspective.

**Theorem 1.** If $\Pi$ is perfect,

$$I(\Pi; Y) \geq I(X_1; Y)$$  

(2)

**Proof.** Since $\Pi$ is perfect, by Lemma 1, we have

$$I(X_1; X_2) = I(X_1; \Pi).$$  

(3)

By data processing inequality, Equation 3 implies $I(X_1; X_2) = 0$. By Assumption 3, a deterministic mapping $f_{X_2 \rightarrow Y}$ from $X_2$ to $Y$ exists. Applying data processing inequality again, we have

$$I(X_1, X_2|\Pi) \geq I(X_1, f_{X_2 \rightarrow Y}(X_2)|\Pi) = I(X_1, Y|\Pi) \geq 0,$$  

(4)

which implies $I(Y, X_1|\Pi) = 0$. Accordingly,

$$H(Y|\Pi) = H(Y|X_1, \Pi) \leq H(Y|X_1)$$  

(5)

Theorem 1 shows that $\Pi$ is a more informative feature than $X_1$. However, a model does not necessarily rely more on a more informative feature. We will discuss more in the next section.

**3.2 Learning from $\Pi$ is Easy**

It is important that learning from $\Pi$ is easy. Because of simplicity bias, a neural network model is likely to rely on the easy-to-learn features (Shah et al., 2020; Kalimeris et al., 2019). We conjecture that a model excessively relies on the spurious feature $X_1$ when learning from $X_1$ is easier than learning from the robust feature $X_2$. If learning from $\Pi$ is easy, then the model will rely on $\Pi$ more and thus will rely on $X_1$ less. However, features with higher mutual information to $Y$ are not necessarily easy to learn. For instance, although $X_2$ is more informative, models tend to rely on $X_1$ instead of $X_2$ at the beginning of the training process. To show that MLM can mitigate the issue brought by simplicity bias, we need to show learning from $\Pi$ is easy.

Therefore, we have the following theorem that implies learning from $\Pi$ is at least as easy as learning from $X_1$:

**Theorem 2.** Let $\tilde{h}_{X_1}^{(D_n)} : X_1 \rightarrow \mathcal{Y}$ be the classifier trained with MLE loss using $n$ data pairs $(x_1^{(1)}, y_1^{(1)}), (x_1^{(2)}, y_1^{(2)}), \cdots, (x_1^{(n)}, y_1^{(n)})$, and the converged classifier be $\tilde{h}_{X_1}^{\ast}$. There exists a learning algorithm, which generates $\tilde{h}_{\Pi}^{(D_n)} : \Pi \rightarrow \mathcal{Y}$ using $(\pi^{(1)}, y_1^{(1)}), (\pi^{(2)}, y_1^{(2)}), \cdots, (\pi^{(n)}, y_1^{(n)})$, such that the following three properties are satisfied: (1)\[\mathbb{E} \left[ D_{KL} \left[ \tilde{h}_{X_1}^{(D_n)} \left| \tilde{h}_{X_1}^{\ast} \right. \right] \right] = O \left( \frac{1}{n} \right), \quad (6)\]

which is asymptotically at the same rate as \[\mathbb{E} \left[ D_{KL} \left[ \tilde{h}_{\Pi}^{(D_n)} \left| \tilde{h}_{\Pi}^{\ast} \right. \right] \right] \]. (2) Over the distribution of $(X, Y)$, the expected loss of the converged classifier $\tilde{h}_{\Pi}^{\ast}$ is not greater than the expected loss of $\tilde{h}_{X_1}^{\ast}$. (3) $\tilde{h}_{\Pi}^{\ast}$ is a linear model, whose input is $\Pi$. (Proof: Appendix A.2)

The remaining question is whether deep learning models used in common practices can perform at least as well as the algorithm in Theorem 2. Indeed, without any knowledge of deep learning models, it is impossible to theoretically prove that a model will necessarily rely on $\Pi$ instead of $X_1$. Therefore, in Section 4 and Section 5 we will empirically validate that our theories are applicable in the real world scenarios.

**3.3 Extending with Causal Models**

Before we validate our theories empirically, we make a step further by relaxing Assumption 3. We do so by treating $X_2$ as a confounder, and then we can see how MLM pre-training is helpful in the causal and anticausal settings.
Table 2: Comparison between Theorem 4 in this work and Theorem 3.3 in Wei et al. (2021).
Specifically, the experimental design is as follows: We use the two-layer and three-layer MLP architectures same as in Section 2. When pretraining, we mask $X_1$ in $X$ by using $X' = [0, 0; X_2]$ as inputs. Let the output from the first linear layer as $Z = W'X'$. The loss function is the cross-entropy between $X_1$ and the softmax over $[z_1, z_2]$. After pretraining, we fine-tune the pretrained model with $(\{X_1; X_2\}, Y')$ pairs, and report the average number of iterations required to converge for 25 different random seeds.

We want to eliminate the possibility that the faster convergence of the pretrained model is because of larger initial weights over $X_1$. Therefore, after pretraining, we manually create a path from $X_1$ to $Z$. We do so by initializing the weights of the third and fourth row of $W$ with $[k, -k; 0, \cdots, 0]$ and $[-k, k; \cdots, 0, 0]$ respectively, where $k$ is the average of the absolute value of the weights in the pretrained part, i.e. the weights of the first two rows in $W$. In this way, the information from $X_1$ has the same scale as the pretrained representation $[Z_1, Z_2]$, and thus it can compete with $[Z_1, Z_2]$ fairly.

Table 1 shows that pretraining can always reduce the number of iterations required to converge when $\nu < 0.50$. The effect is more significant when $d_2$ is larger. It could be because of the higher sparsity of the learning signal from $X_2$ when $d_2$ is larger.

We further inspect how the importance over the inputs changes in the process of training. The importance can be inferred from the product of the weights over $X$ after pretraining, we manually create a path from $X_1$ to $Z_1, Z_2$, and $X_2$ (the middle part of $X_2$) respectively. The orange curve represents the accuracy.

![Figure 3: The average weights over the features during training a two-layer model. The upper and the lower rows are the curves of model without and with pretraining respectively. From left to right, $(\nu, d_2) = (50, 0.04), (500, 0.04)$. Blue, green, purple curves represent the average weights over features in $X_1, X_2$, and $X_2$ (the middle part of $X_2$) respectively. The orange curve represents the accuracy.](image)

5 Experiments

To further validate our theories, we experiment on real world NLP tasks. We facilitate datasets that have known spurious features. We first pretrain models on the training dataset with different masking policies. Different masking policies mask the spurious tokens in different ways. Afterward, we fine-tune the model using the target label. We show that the models will be less robust on downstream tasks if spurious tokens are not masked during pretraining, and always masking the spurious token during pretraining improving the robustness. The results validate our theories.

5.1 Dowstream Tasks

Hate Speech Detection Previous study has shown that hate speech detection datasets tend to have lexical bias (Dixon et al., 2018). That is, models rely excessively on the presence or the absence of certain words when predicting the label. Here we follow the formulation of lexical bias in hate speech detection proposed by Zhou et al. (2021). We focus on the effect of non-offensive minority identity (NOI) mentions, such as “woman”, “gay”, “black”. Those mentions are often highly associated with hateful instances. However, it is more desirable that a model does not rely on those mentions. Therefore, we can see the presence of NOI as a spurious feature.

Name Entity Recognition (NER) Lin et al. (2020) has shown that name entity recognition
(NER) models perform worse when the name entities are not seen in the training data. In this case, we can see the content of the name entities as a spurious feature. Models may learn to memorize the name entities when fitting the training data, while we may desire the model to recognize name entities according to the context.

5.1.1 Datasets

**Hate Speech Detection** We use a portion of the dataset proposed by Zhou et al. (2021). In their original dataset, only a small number of hateful instances contain NOI. Our preliminary experiments show that the model without pretraining does not suffer much from the bias of NOI when training with the full data. Therefore, we create a dataset, whose positive (hateful) instances are all the positive samples in the original dataset that contain NOI. As for negative instances, we sample them randomly from the original training set. We control the number of negative instances so the ratio of positive and negative instances is the same as the original dataset. We create both the training and the validation splits in this way, and use the original full testing set for evaluation. We also evaluate the models on a NOI subset where all the instances contain NOI.

**NER** We use the standard NER dataset Conll-2003 (Tjong Kim Sang and De Meulder, 2003). To create a testing set with unseen name entities, we replace the name entities in the original validation and testing splits with the entities from WNUT-17 (Derczynski et al., 2017). Specifically, we replace the LOC, ORG, PER entities with the corresponding type of entities in WNUT-17, while the MISC entities remain untouched.

5.2 Masking Policies

For each sentence with $n_s$ spurious tokens, we experiment with different masking policies: (1) **scratch**: We do not pretrain the model before fine-tuning. (2) **vanilla**: During pretraining, we mask each token with 15% probability, which is same as the original implementation in (Devlin et al., 2019). (3) **unmask random**: This is similar to vanilla MLM, but we uniformly randomly select $n_s$ tokens from the whole sentence and unmask them if they have been masked. (4) **unmask spurious**: This is similar to vanilla MLM, but we unmask all the spurious tokens. (5) **remove spurious**: We replace spurious tokens with a special “[tunk]” token, and we unmask them. Note that this setting can be seen as an oracle setting, since in most applications the spurious features are unknown.

Note that setting (3), (4), (5) have the same expected number of masked tokens. Therefore, it rules out the possibility that their downstream performance differs because of the number of masked tokens. Instead, the difference should be resulted from whether spurious tokens are masked during pretraining.

5.3 Implementation Details

For both of the tasks and all the MLM settings, we tokenize the input with the bert-base-uncased tokenizer. We use the bert-base-uncased architecture, and also the pretrained embedding layer, which is frozen through the pretraining process. We include more details in Appendix A.3.

5.4 Result and Discussion

Results in Table 3 validate our theorems. For both of the tasks, **unmask random** performs better than **unmask spurious** under distribution shift. Specifically, **unmask random** has higher F1 on the unseen set of the NER task, and **unmask random** has a lower false positive rate (FPR) on the NOI set. Also, **unmask random** performs similarly to **random**. This implies that modeling the condition distribution of spurious tokens in the original random masking pretraining can reduce models’ reliance on them. Note that **unmask random** and **unmask spurious** have similar in-distribution performance, so the performance difference is not due to better in-distribution generalization suggested by Miller et al. (2021).

We also compare **unmask random** with the oracle setting **remove spurious**. We notice that even though **remove spurious** performs as well as **random**, **remove spurious** hurts the performance in the seen set. It indicates that modeling the conditional distribution of spurious tokens has effects beyond simply removing them from the model. On the other hand, **remove spurious** performs better in the hate speech detection task. A possible explanation is that NOI mentions contain little useful information for the task.

6 Related Work

Recently, there are efforts attempting to explain the effectiveness of massive language modeling pretraining. Theoretically, Saunshi et al. (2021)
Table 3: The performance on downstream tasks. For the hate speech detection task, we also report false positive detection (FPR) on the NOI subset, which is a set of instances containing non-offensive minority identity mentions, e.g. "women", "black".

<table>
<thead>
<tr>
<th>Mask Policy</th>
<th>NER</th>
<th></th>
<th>Hate Speech Detection</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>All (12893)</td>
<td>NOI (602)</td>
</tr>
<tr>
<td></td>
<td>F1</td>
<td>F1</td>
<td>Accuracy ↑</td>
<td>F1</td>
</tr>
<tr>
<td>Origin</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>vanilla</td>
<td>74.0</td>
<td>5.4</td>
<td>65.6</td>
<td>1.3</td>
</tr>
<tr>
<td>unmask random</td>
<td>72.7</td>
<td>6.5</td>
<td>56.5</td>
<td>0.8</td>
</tr>
<tr>
<td>unmask spurious</td>
<td>72.9</td>
<td>5.0</td>
<td>53.2</td>
<td>0.8</td>
</tr>
<tr>
<td>remove spurious</td>
<td>69.8</td>
<td>5.5</td>
<td>56.7</td>
<td>0.8</td>
</tr>
</tbody>
</table>

Explore why auto-regressive language models help solve downstream tasks. However, their explanation is based on the assumption that the downstream tasks are natural tasks, i.e. tasks that can be reformulated as sentence completion tasks. Their explanation also requires the pretrained language model to perform well for any sentence completion tasks, which is not likely to be true in the real world. Wei et al. (2021) analyzes the effect of fine-tuning a pretrained MLM model. Nonetheless, they have stronger assumptions as described in Section A.3. Aghajanyan et al. (2020) show that pretrained models have lower intrinsic dimension, and provide a generalization bound based on Arora et al. (2018). However, why pretrained models have lower intrinsic dimension is still unknown. Merrill et al. (2021) show that the parameter norm growth during training makes transformer a saturated model, which can be described in terms of formal languages. Empirically, Zhang and Hashimoto (2021) show that the effectiveness of MLM pretraining cannot be explained by formulating the downstream tasks as sentence completion problems. Sinha et al. (2021) find evidence supporting the hypothesis that masked language models benefit from modeling high-order word co-occurrence instead of word order. There are also some theories explaining the efficacy of non-MLM pretraining Lee et al. (2020); Saunshi et al. (2019); Zhang and Stratos (2021).

Many of the previous studies on robust NLP focus on supervised learning (Wang et al., 2021; Utama et al., 2020b,a; Karimi Mahabadi et al., 2020; Chang et al., 2020; He et al., 2019; Sagawa* et al., 2020; Kennedy et al., 2020; Chiang et al., 2020). However, without self-supervised learning, a model cannot possibly extrapolate to out-of-distribution data when the domain shifts. Our work also complements previous studies that focus on the bias or robustness of a model generated by the pretraining process (Kumar et al., 2020; Hawkins et al., 2020; Vargas and Cotterell, 2020; Liu et al., 2020; Gonen and Goldberg, 2019; Kurita et al., 2019; Zhao et al., 2019). In this work we investigate the pretraining process itself.

7 Implication

Our results provide possible explanations for some common practices found effective empirically. First, it could explain why continuing pretraining on target dataset is useful (Gururangan et al., 2020). It may be because continuing pretrained models model the distribution of spurious features in the target dataset better. Thus the model can better avoid the simplicity pitfall. Second, it provides reasons for more complex masking policies, such as masking continuous random spans (Joshi et al., 2019). It may improve the robustness to spurious features that contain more than one token. Third, if MLM can alleviate the simplicity bias and help the model to achieve a greater margin, it may also imply that the model has wider optima, explaining the finding in Hao et al. (2019).

8 Conclusion

In this work, we show a benefit of MLM pretraining, which partly explains its efficacy. We first empirically demonstrate the presence of simplicity bias when the input is discrete. We then theoretically and empirically explain how MLM pretraining can alleviate the problem brought by it. Finally, we close the gap between our theories and real-world practices with experiments on real-world NLP tasks. Our results shed light on future research on robustness and self-supervised learning.
References


Tomas Mikolov, Kai Chen, Gregory S. Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. In ICLR.


Koustuv Sinha, Robin Jia, Dieuwke Hupkes, Joelle Pineau, Adina Williams, and Douwe Kiela. 2021. Masked language modeling and the distributional


A Appendix

A.1 Proof of Lemma 1

Proof. Since $X_2$ is discrete, $\Pi_1$ is discrete too.

$$H(X_1|\Pi_1) = \sum_{x_1,\pi_1} P(x_1,\pi_1) \log P(x_1|\pi_1)$$

$$= \sum_{x_1,\pi_1} \sum_{x_2} P(x_1, x_2) \log P(x_1|x_2)$$

$$= H(X_1|X_2)$$

A.2 Proof of Theorem 2

To proof Theorem 2, we need a lemma from Gibbs and Su (2002); Paninski (2003) for the convergence rate of empirical measures.

Lemma 2. Given $n$ samples $x_1, x_2, \ldots, x_n$ of a random variable $X \in \{1, 2, \ldots, m\}$. Let

$$q_i^{(n)} = \frac{1}{n} \sum_{j=1}^{n} \mathbb{1}[x_j = i]. \quad (8)$$

The expected convergence rate

$$\mathbb{E} \left[ D_{KL} \left[ q^{(n)} \parallel p \right] \right] = O \left( \frac{1}{n} \right), \quad (9)$$

where $p_i = P(X = i)$. 

11
Figure 4: The average weights over the features during training a two-layer model without pretraining. From left to right, $\nu = 0.04, 0.10, 0.25, 0.5$. From top to bottom, $d_2 = 50, 100, 500$. Blue, green, purple curves represent the average weights over features in $X_1$, $X_2$, and $\dot{X}_2$ (the middle part of $X_2$) respectively. The orange curve represents the accuracy.
Figure 5: The average weights over the features during training a two-layer model with pretraining. From left to right, $\nu = 0.04, 0.10, 0.25, 0.5$. From top to bottom, $d_2 = 50, 100, 500$. Blue, green, purple curves represent the average weights over features in $X_1$, $X_2$, and $\dot{X}_2$ (the middle part of $X_2$) respectively. The orange curve represents the accuracy.
Proof.

\[
\sum_{i=1}^{m} q_i^{(n)} \log \frac{q_i^{(n)}}{p_i} \\
\leq \log \left( \sum_{i=1}^{m} \frac{q_i^{(n)} \log q_i^{(n)}}{p_i} \right) \quad \text{(By concavity of log)} \\
= \log \left( \sum_{i=1}^{m} \frac{(q_i^{(n)} - p_i)^2}{p_i} + 1 \right) \\
\leq \sum_{i=1}^{m} \frac{(q_i^{(n)} - p_i)^2}{p_i} \\
\mathbb{E} \left[ \sum_{i=1}^{m} \frac{(q_i^{(n)} - p_i)^2}{p_i} \right] = O\left( \frac{1}{n} \right)
\]

\(\square\)

**Lemma 3.** Let \(q^{(a)}, q^{(b)}\) be the empirical distribution estimated by counting \(n\) samples following \(p^{(a)}, p^{(b)}\). If \(D_{KL}(p^{(a)} \parallel q^{(a)}) = O(f(n))\) and \(D_{KL}(p^{(b)} \parallel q^{(b)}) = O(f(n))\) for some function \(f(n)\) (e.g. \(O(1/n)\)), then \(D_{KL}(p^{(a)}p^{(b)} \parallel q^{(a)}q^{(b)}) = O(f)\).

With these two lemmas, we can prove Theorem 2:

**Proof.** Proof sketch of Theorem 2: The classifier that maximizes the likelihood of \((x_1, y^{(1)}), (x_2, y^{(2)}), \ldots, (x_n, y^{(n)})\) can be attained by counting the co-occurrence of \(X_1\) and \(Y\).

\[
\tilde{h}^{(n)}_{X_1}(y|X_1 = x) = \sum_{i=1}^{n} \mathbb{1}[y_i = y] \sum_{i=1}^{n} \mathbb{1}[x_i = x] \\
(10)
\]

It converges to

\[
\tilde{h}^{*}_{X_1}(y|X_1 = x) = P(y|X_1 = x). \\
(11)
\]

Based on \(\Pi_1\), a classifier can be attained by first estimating \(P(Y)\) and \(P(x_1|y)\) for all \(x_1\) and \(y\):

\[
\rho^{(n)}_{y|x_1} = \frac{\sum_{i|y^{(i)}=y} \pi_{x_1}^{(i)}}{\sum_{i} \pi_{x_1}^{(i)}}, \\
(12)
\]

where \(\pi_{x_1}^{(i)} = \Pi(X_1 = x_1^{(i)}|X_2 = x_2^{(i)})\), and then we can construct a classifier

\[
\tilde{h}^{(n)}_{\Pi}(y|\pi) = \sum_{x_1} \rho^{(n)}_{y|x_1} \pi_{x_1}. \\
(13)
\]

It converges to

\[
\tilde{h}^{*}_{\Pi}(y|\pi) = \sum_{x_1} P(y|x_1) \pi_{x_1}. \\
(14)
\]

Based on Lemma 2 and Lemma 3, we have \(\mathbb{E}\left[ D_{KL} \left( \tilde{h}^{(n)}_{X_1} \parallel \tilde{h}^{*}_{X_1} \right) \right] = O\left( \frac{1}{n} \right)\) and \(\mathbb{E}\left[ D_{KL} \left( \tilde{h}^{(n)}_{\Pi} \parallel \tilde{h}^{*}_{\Pi} \right) \right] = O\left( \frac{1}{n} \right)\).

Then we show that \(\tilde{h}^{*}_{\Pi}(y|\pi)\) is at least as good as \(\tilde{h}^{*}_{X_1}(y|X_1)\) by showing \(D_{KL}(P(Y|X) \parallel \tilde{h}^{*}_{X_1}(Y|X)) \geq D_{KL}(P(Y|X) \parallel \tilde{h}^{*}_{\Pi}(Y|X))\) with convexity:

\[
\sum_{x_1} P(x_1|x_2)D_{KL}[P(Y|x_2) \parallel P(Y|x_1)] \geq D_{KL}(P(Y|x_2) \parallel \sum_{x_1} P(Y|x_1)P(x_1|x_2)). \\
(15)
\]

**A.3 Implementation Details of the Experiments**

We pretrain the models until they converge, and choose the checkpoint with the lowest MLM loss on the validation set. For the hate speech detection task, we use the implementation provided by Zhou et al. (2021). Except that we use bert-base-uncased instead of roberta-large, we use the other hyper parameters provided in their script. For the NER task, we use the implementation by Hugging Face.\(^2\)

\(^2\)https://github.com/huggingface/transformers/blob/master/examples/pytorch/token-classification/run_ner.py