N-gram Prediction and Word Difference Representations for Language Modeling

Anonymous ACL submission

Abstract

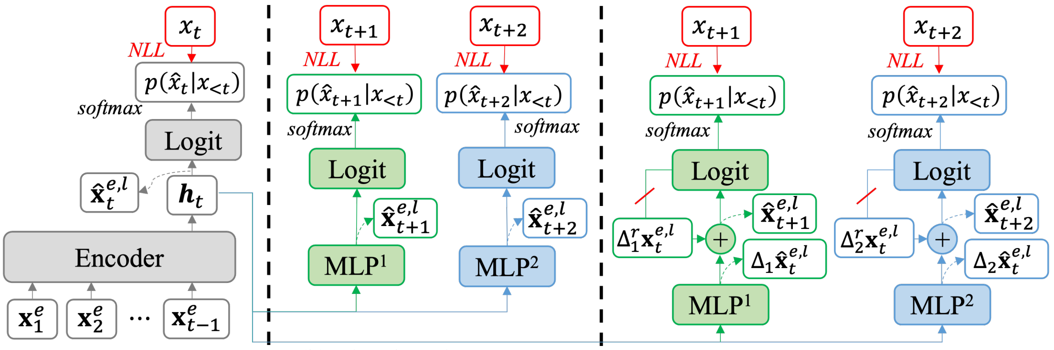
Causal language modeling (CLM) serves as the foundational framework underpinning remarkable successes of recent large language models (LLMs). Despite its success, the training approach for next word prediction poses a potential risk of causing the model to overly focus on local dependencies within a sentence. While prior studies have been introduced to predict future N words simultaneously, they were primarily applied to tasks such as masked language modeling (MLM) and neural machine translation (NMT). In this study, we introduce a simple N-gram prediction framework for the CLM task. Moreover, we introduce word difference representation (WDR) as a surrogate and contextualized target representation during model training on the basis of N-gram prediction framework. To further enhance the quality of next word prediction, we propose an ensemble method that incorporates the future N words' prediction results. Empirical evaluations across multiple benchmark datasets encompassing CLM and NMT tasks demonstrate the significant advantages of our proposed methods over the conventional CLM.

1 Introduction

With the remarkable advancements in deep learning techniques, neural language modeling has become a central component in modern natural language processing (NLP) tasks, such as natural language understanding (NLU), neural machine translation (NMT) and question answering. Among the approaches to language modeling, causal language modeling (CLM), which predicts the next word given the previous words, is a widely employed language modeling framework. For example, prominent large language models (LLMs) like GPT-2 (Radford et al., 2019) and GPT-3 (Brown et al., 2020) rely on CLM as their primary training framework. Despite their successful applications, the prevalent next word prediction manner can inadverently lead models to overfit to local dependencies rather than capturing long-term dependencies between words. This tendency arises from some phrases or paired words that have strong dependencies with each other, such as "*Barack Obama*" and "*Harry Potter*" (Qi et al., 2020).

A way of mitigating this problem involves predicting not solely the next word but also subsequent words in later time-steps such as N-gram prediction. Researchers (Sun et al., 2019; Joshi et al., 2020; Xiao et al., 2020; Qi et al., 2020) have adopted this N-gram prediction methodology for the masked language modeling (MLM) during the pre-training phase of LLMs (Devlin et al., 2018). Similar approaches have been applied to the NMT task (Shao et al., 2018; Ma et al., 2018; Shao et al., 2020). However, these methods often require significant modifications to the model architecture, a different loss function than the conventional cross-entropy loss, or an expansion of the vocabulary for N-grams.

This paper introduces a novel N-gram prediction framework designed specifically for CLM and proposes innovative methods aimed at fortifying this framework. The contributions of this work can be summarized as follows. *(1) A simple N-gram prediction for CLM*: we propose a simple N-gram prediction integrated to existing CLM models. Except for an additional multi-layer perceptron (MLP) layer, our method does not require other modifications to model architecture, loss function, and vocabulary. *(2) Word difference representation*: we propose to use the embedding vectors' difference between contiguous words, termed word difference representation (WDR), as a surrogate representation for individual words. Departing from the conventional approaches that employing a fixed word embedding as target representation, we provide diverse WDR as target representations in accordance with context. We discovered this method can vary backpropagated gradient during training so that it


x_t
NLL
$p(\hat{x}_t|x_{<t})$
softmax
Logit
$\hat{\mathbf{x}}_t^{e,l}$
$\boldsymbol{h}_t$
Encoder
$\mathbf{x}_1^e$
$\mathbf{x}_2^e$
$\cdots$
$\mathbf{x}_{t-1}^e$
x_{t+1}
NLL
$p(\hat{x}_{t+1}|x_{<t})$
softmax
Logit
$\hat{\mathbf{x}}_{t+1}^{e,l}$
MLP¹
x_{t+2}
NLL
$p(\hat{x}_{t+2}|x_{<t})$
softmax
Logit
$\hat{\mathbf{x}}_{t+2}^{e,l}$
MLP²
x_{t+1}
NLL
$p(\hat{x}_{t+1}|x_{<t})$
softmax
Logit
$\Delta_1^r\mathbf{x}_t^{e,l}$
$+$
$\hat{\mathbf{x}}_{t+1}^{e,l}$
$\Delta_1\hat{\mathbf{x}}_t^{e,l}$
MLP¹
x_{t+2}
NLL
$p(\hat{x}_{t+2}|x_{<t})$
softmax
Logit
$\Delta_2^r\mathbf{x}_t^{e,l}$
$+$
$\hat{\mathbf{x}}_{t+2}^{e,l}$
$\Delta_2\hat{\mathbf{x}}_t^{e,l}$
MLP²

(a) Convectional CLM (b) Simple Trigram CLM (c) WDR Trigram CLM

Figure 1: Model illustrations of (a) conventional CLM, (b) simple N-gram CLM, and (c) WDR N-gram CLM when $N = 3$. Note that all of the drawn logit layers above the MLP layers are the same function with the same parameter. Red diagonal lines in (c) on lines from logit layer to $\Delta_i^r \mathbf{x}_t^{e,l}$ indicate detaching operation.

can enhance generalizability. The algorithmic reversibility of WDR preserves the feasibility of the above simple N-gram prediction method. *(3) An ensemble method suitable for the CLM task*: we propose an ensemble method designed to refine the next word prediction by leveraging other multiple N predictions from the N-gram prediction.

Our preliminary and primary experimental results, conducted several CLM benchmark datasets, highlight the gradual improvements in perplexity achieved by our proposed simple N-gram framework, the WDR diverse target representations, and ensemble method when compared to several baseline models. Our qualitative analysis focusing on gradient elucidates the advantage of the WDR method from the perspective of optimization generalizability. In addition to the main CLM task, we demonstrate the applicability and advantages of our proposed approaches to the NMT task, which is a conditional form of the CLM task.

2 Background: Conventional CLM

Since the work of (Bengio et al., 2000), neural network-based language modeling has been developed and become mainstream in language modeling. As background knowledge, we describe the conventional training framework of CLM (the next word prediction) in this section.

A sentence consists with words, $X = \{x_1, x_2, \cdots, x_T\}, x \in \mathcal{V}$, where T means the sequence length of the sentence and $\mathcal{V}$ is the vocabulary set. Conventional CLM computes the likelihood of a word conditioned on its preceding words in the sentence, $p(x_t|x_{<t})$. For processing, words are mapped to embedding vectors (Mikolov et al., 2013), and the encoded hidden state at time-step t is formulated as follows:

$$\mathbf{h}_t = Enc_\theta(\{\mathbf{x}_1^e, \mathbf{x}_2^e, \cdots, \mathbf{x}_{t-1}^e\}) \in \mathbb{R}^d, \qquad (1)$$

where $\mathbf{x}_t^e \in \mathbb{R}^d$ means the embedded vector of x_t. Enc_θ is an encoder model with its parameter set θ. d is the dimension of the encoded hidden state and the embedding vector spaces. Recently, most language models use Transformer (Vaswani et al., 2017) as their encoder architecture. After encoding, the encoded hidden state is linearly transformed to a logit value of each word in a vocabulary set $\mathcal{V}$. Finally, the likelihood of the predicted word is formulated as follows:

$$\begin{aligned} p(\hat{x}_t|x_{<t};\theta) &= softmax(\hat{\mathbf{x}}_t^l), \\ \hat{\mathbf{x}}_t^l &= \mathbf{W}^l\mathbf{h}_t = \mathbf{W}^l\hat{\mathbf{x}}_t^{e,l}, \end{aligned} \qquad (2)$$

where $\mathbf{W}^l \in \mathbb{R}^{|\mathcal{V}|\times d}$ is the weight matrix of the logit layer.

To help the understanding of our idea, we note that a parameter vector of logit layer's weight is another word embedding set that is mapped to the target word, that is $\mathbf{W}^l = [\mathbf{x}_1^{e,l}, \mathbf{x}_2^{e,l}, \cdots, \mathbf{x}_{|\mathcal{V}|}^{e,l}]^\top$. In this point of view, the encoded hidden state, $\mathbf{h}_t$, is the predicted word embedding vector of the logit layer, $\hat{\mathbf{x}}_t^{e,l}$. Then, the inner product between $\mathbf{W}^l$ and $\hat{\mathbf{x}}_t^{e,l}$ outputs the predicted score of each embedding that indicates how the predicted word embedding is similar to the logit layer's original word embedding.

Finally, the model learns to minimize the negative log-likelihood (NLL) loss as follows:

$$\mathcal{L}(X,\theta) = -\sum_{t=1}^{T} \log p(\hat{x}_t = x_t|x_{<t};\theta). \qquad (3)$$

This loss becomes the minimum when the model exactly predicts the logit layer's embedding of the target word, that is $\hat{\mathbf{x}}_t^{e,l} = \mathbf{x}_t^{e,l}$. This process is illustrated in Fig.1(a).

3 Proposed Methods

In this section, we propose three ideas: (1) a simple N-gram CLM, (2) word difference representation N-gram CLM, and (3) an ensemble method over N-gram predictions.

3.1 Simple N-gram CLM

First, we propose a simple N-gram prediction on the conventional framework of CLM. The core idea is adding an MLP layer to predict a future word given the same hidden state of the conventional CLM. This process is formulated as follows:

$$\hat{\mathbf{x}}_{t+n}^{e,l} = MLP^n(\mathbf{h}_t). \tag{4}$$

For instance, assuming N is 3, two MLP layers, MLP^1 and MLP^2, are employed and predict $\hat{\mathbf{x}}_{t+1}^{e,l}$ and $\hat{\mathbf{x}}_{t+2}^{e,l}$, respectively, as shown in Fig.1(b). The limited capability of the MLP layer to learn an effective function from a large and complicated dataset may regularize the main encoder, Enc_θ, to encode a simultaneously informative hidden state for all N-gram predictions. This regularization might be beneficial to prevent the model to overly focus on local dependencies.

We compute the likelihoods of the future target words, $p(\hat{x}_{t+1}|x_{<t};\theta)$ and $p(\hat{x}_{t+2}|x_{<t};\theta)$ in the above example, following each logit layer and the softmax function. Instead of using individual logit layers for each future word prediction, we share the parameters of all logit layers, including the conventional CLM model's logit layer. Therefore, this approach increases just a small amount of parameters for each additional MLP layer. Furthermore, it re-uses the original (unigram) vocabulary set for the future word prediction, not an additional large vocabulary set of N-grams. The loss for n-th future word prediction is as follows:

$$\mathcal{L}_n(X,\theta) = -\sum_{t=1}^{T-n} \log p(\hat{x}_{t+n} = x_{t+n}|x_{<t};\theta). \tag{5}$$

As like Eq.(3), this loss becomes minimum when the model exactly predicts the future target word's embedding, i.e., $\hat{\mathbf{x}}_{t+n}^{e,l} = \mathbf{x}_{t+n}^{e,l}$. The total loss for the training of this simple N-gram CLM model is the mixture of Eq.(3) and Eq.(5) as follows:

$$\mathcal{L}_N^{tot}(X,\theta) = \frac{1}{2}\mathcal{L}(X,\theta) + \frac{1}{2(N-1)}\sum_{i=1}^{N-1}\mathcal{L}_i(X,\theta). \tag{6}$$

Notably, we do not equally take the average of the original loss, Eq.(3), with other losses, since the next word typically has stronger dependencies with the preceding words than other future words. In other words, averaging the entire set of loss terms together might introduce excessive regularization.

3.2 Word Difference Representation (WDR) N-gram CLM

To use a more informative target than simple N-gram CLM, we introduce the idea of WDR which is a contextualized surrogate representation of words within a sentence. Basically, it is based on a form of word embedding compositions: the difference vector, $\mathbf{x}_{t+1}^e - \mathbf{x}_t^e$. Since (Mikolov et al., 2013) demonstrated that arithmetic compositions of learned word embedding can convey semantic meanings, many researches have explored the word embedding compositionality (Xu et al., 2015; Hartung et al., 2017; Poliak et al., 2017; Scheepers et al., 2018; Li et al., 2018; Frandsen and Ge, 2019). Their studies utilized composed word embeddings as inputs to models, instead of original word embeddings, showcasing their advantages across various NLP tasks.

Unlike the prior research, we provide WDR to the model as the target to predict, rather than utilizing it as input. The difference vector of contiguous words offers a different representation for the word depending on its adjacent words. Therefore, by leveraging WDR as the target, we expect the model can learn more diverse targets than previous works. Generating WDR is simple repetition of vector subtractions which is computationally cheap and easy to parallelize, so it does not impose a high computational cost. Moreover, generating WDR is reversible, so that original embedding vectors can be reconstructed from WDR. This property facilitates the development of WDR-based N-gram CLM integrating the same framework of the simple N-gram CLM without a significant modification. Detailed explanations elucidating these advantages are provided in the subsequent sections.

3.2.1 Definition of n-level WDR

As we briefly mentioned above, we use the difference of contiguous embedding vectors as the base of WDR. Given an embedding vector sequence $\{\mathbf{x}^e_1, \mathbf{x}^e_2, \cdots, \mathbf{x}^e_T\}$, the 1-level WDR at the time-step t is defined as follows:

$$\Delta_1 \mathbf{x}^e_t = \begin{cases} \mathbf{x}^e_{t+1} - \mathbf{x}^e_t & \text{if } 1 \le t < T, \\ \mathbf{x}^e_T & \text{if } t = T. \end{cases} \tag{7}$$

In an inductive manner, the n-level WDR at the time-step t when $n > 1$ is defined as follows:

$$\Delta_n \mathbf{x}^e_t = \begin{cases} \Delta_{n-1}\mathbf{x}^e_{t+1} - \Delta_{n-1}\mathbf{x}^e_t & \text{if } 1 \le t < T, \\ \Delta_{n-1}\mathbf{x}^e_T = \mathbf{x}^e_T & \text{if } t = T. \end{cases} \tag{8}$$

As an alternative of the above n-level WDR definition, we explored the opposite direction to subtract the contiguous vectors, that is $\Delta_{n-1}\mathbf{x}^e_t - \Delta_{n-1}\mathbf{x}^e_{t+1}$. In our internal empirical studies, we discovered the alternative design achieved similar performances. Therefore, we follow the design of Eq. 8 throughout this paper.

Based on the definitions of Eqs. 7 and 8, the n-level WDR can be represented by the composition of original word embeddings. For example, the 2 and 3-level WDRs at time-step t can be represented as follows: $\Delta_2 \mathbf{x}^e_t = \mathbf{x}^e_{t+2} - 2\mathbf{x}^e_{t+1} + \mathbf{x}^e_t$ and $\Delta_3 \mathbf{x}^e_t = \mathbf{x}^e_{t+3} - 3\mathbf{x}^e_{t+2} + 3\mathbf{x}^e_{t+1} - \mathbf{x}^e_t$, respectively. With this manner, we can derive the formulation of n-level WDR as follows:

$$\Delta_n \mathbf{x}^e_t = \sum_{i=0}^{n} \binom{n}{i} (-1)^i \mathbf{x}^e_{t+(n-i)}, \tag{9}$$

where $\binom{n}{i} = \frac{n!}{(n-i)!i!}$ is the binomial coefficient. This equation holds for every positive integer of n and for every time-step t when $t \le T - n$. See Appendix A.1 for a proof of this equation.

As we mentioned earlier, n-level WDR is reversible to the original word embedding. For the 1-level WDR, $\mathbf{x}^e_{t+1}$ can be reconstructed by adding $\mathbf{x}^e_t$ to $\Delta_1 \mathbf{x}^e_t$. Likewise, $\mathbf{x}^e_{t+n}$ can be reconstructed by adding $-\sum_{i=1}^{n} \binom{n}{i}(-1)^i \mathbf{x}^e_{t+(n-i)}$ to $\Delta_n \mathbf{x}^e_t$ (note that the first term of the right-hand side of Eq.(9) is $\mathbf{x}^e_{t+n}$). For simplicity, we use a new notation for the conjugate term that reconstructs the original embedding by addition to the n-level WDR as follows:

$$\Delta^r_n \mathbf{x}^e_t = -\sum_{i=1}^{n} \binom{n}{i} (-1)^i \mathbf{x}^e_{t+(n-i)}, \tag{10}$$

This leads to $\Delta_n \mathbf{x}^e_t + \Delta^r_n \mathbf{x}^e_t = \mathbf{x}^e_{t+n}$. The conjugate term for reconstruction, $\Delta^r_n \mathbf{x}^e_t$, can be obtained by Eq.(10) or iterative operations of Eq.(8).

3.2.2 Training of WDR N-gram CLM

We develop the WDR-based N-gram CLM from the framework of simple N-gram CLM. To achieve the mentioned goal that providing the WDR as the target of the model, we apply the definitions and derivations in Sec.3.2.1 to the logit layer's embeddings. Following the idea of the simple N-gram CLM described in Sec.3.1, we employ MLP layers for predictions of N-gram. However, in WDR N-gram CLM, the MLP^n layer outputs $\Delta_n \hat{\mathbf{x}}^{e,l}_t$ instead of $\hat{\mathbf{x}}^{e,l}_{t+n}$. Then we produce its corresponding conjugate term, $\Delta^r_n \mathbf{x}^{e,l}_t$, based on the logit layer's embedding matrix. Adding those two, $\Delta_n \hat{\mathbf{x}}^{e,l}_t + \Delta^r_n \mathbf{x}^{e,l}_t$, yields $\hat{\mathbf{x}}^{e,l}_{t+n}$ as in the simple N-gram CLM. Then, we take the same processes of the logit, likelihood, and loss computations as in the simple N-gram CLM.

An essential design of this framework is detachment of the produced conjugate term, $\Delta^r_n \mathbf{x}^{e,l}_t$, from the backpropagation process. Absence of this detachment might lead the model to adjust the logit layer's weight matrix in a distorted manner, because the input of the logit layer is recursively produced from itself.

In WDR N-gram CLM, the minimum value of NLL loss of x_{t+n} prediction, Eq.(5), is achieved when $\hat{\mathbf{x}}^{e,l}_{t+n} = \mathbf{x}^{e,l}_{t+n}$, which is $\Delta_n \hat{\mathbf{x}}^{e,l}_t + \Delta^r_n \mathbf{x}^{e,l}_t = \Delta_n \mathbf{x}^{e,l}_t + \Delta^r_n \mathbf{x}^{e,l}_t$ based on the equation led by Eq.(10). Because the conjugate term, $\Delta^r_n \mathbf{x}^{e,l}_t$, is detached, the model would learn to predict $\Delta_n \mathbf{x}^{e,l}_t$, which is true n-level WDR. In other words, WDR N-gram CLM learns to predict composed word embeddings, offering diverse and contextualized target representations, even for the same target word. The entire process of WDR trigram CLM example is illustrated in Fig.1(c).

3.2.3 How Diverse Are WDR-based Target Representations?

In order to gain a more profound understanding of WDR as target representations, we explored how WDR would diversify target representations compared to the conventional CLM or the simple N-gram CLM. As we mentioned in Sec.2 and Sec.3.1, the conventional CLM and the simple N-gram CLM utilize the logit layer's embeddings as target representations to predict. To see the practical examples of these target representations, we collected 1,270 representations from the logit layer's embedding matrix of the pre-trained conventional CLM model ('TF' in the preliminary experiment,

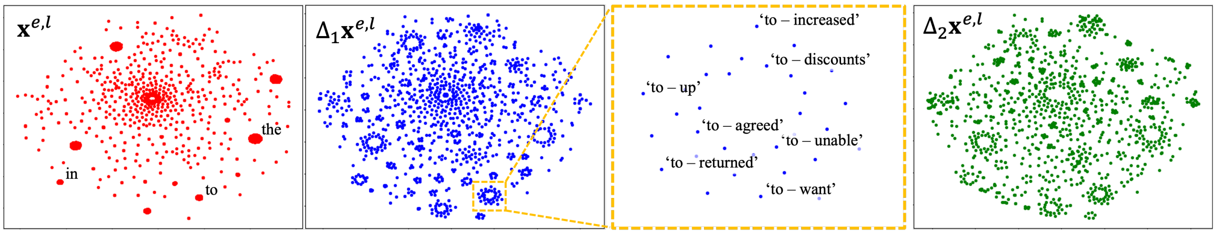


Figure 2: From the left-to-right, they are visualizations of the original embeddings (first), 1-level WDR and the plot zoomed in around the original word 'to' (second and third), and 2-level WDR (last), respectively. In the third plot, ('to-*word*') means the 1-level WDR vector, that is $\mathbf{x}^{e,l}_{to} - \mathbf{x}^{e,l}_{word}$ based on the word '*to*' fragment within the sentence.

Sec.4.1.3). The 1,270 representations correspond to all the tokens of randomly selected 10 sentences from the Penn TreeBank (PTB) (Mikolov et al., 2014) testset. Also, we computed 1 and 2-level WDRs with the collected embeddings, and added them to the collection, resulting in 3,810 representations in total. Finally, we reduced the dimension of the total collection to 2-dimension with t-SNE algorithm (Van der Maaten and Hinton, 2008).

Fig.2 shows the collected representations in a 2-dimensional space. The first plot illustrates the original embeddings, $\mathbf{x}^{e,l}$. Note that the representations of frequent words, such as 'to' may be included more times than other words in the collection. We interpret that this is the reason why t-SNE places frequent words (e.g., 'in', 'to', and 'the') distant from other less frequent words to resemble the non-uniform distribution of the collection. On the other hand, the 1-level WDR representations, $\Delta_1\mathbf{x}^{e,l}$, look more diverse compared to the original embeddings as in the second plot. For example, by composing adjacent words such as 'want', 'unable', 'returned', into the frequent word 'to', it diversifies the embedding representations according to its previous word as in the third plot which is zoomed in. The 2-level WDR looks more diverse even compared to 1-level WDR as in the last plot. Based on this analysis, we expect WDR N-gram CLM to give more diverse target representations than other methods, such as conventional CLM and the simple N-gram CLM.

3.3 Ensemble Method to Refine the Next Word Prediction Leveraging N-gram Predictions

We propose a new ensemble method to incorporate the N-gram predictions into the process of the next word prediction. The encoder model, such as Transformer, outputs $\{\mathbf{h}_2, \mathbf{h}_3, \cdots, \mathbf{h}_t\}$ given the embedded input sentence $\{\mathbf{x}^e_1, \mathbf{x}^e_2, \cdots, \mathbf{x}^e_{t-1}\}$. The encoded hidden state $\mathbf{h}_i$ represents the computed hidden state given the inputs up to time-steps $(i-1)$. At testing, in addition to the predicted embedding $\hat{\mathbf{x}}^{e,l}_t$ from the conventional CLM, MLP^n layer of N-gram CLM can estimate the target word for time t given $\mathbf{h}_{t-n}$. Therefore, we can get N predicted embeddings for the current time-step. We ensemble these predicted embeddings just before the logit layer using the following formulation:

$$\hat{\mathbf{x}}^{e,l}_{t,ens} = (1-\lambda)\hat{\mathbf{x}}^{e,l}_t + \frac{\lambda}{N-1}\sum_{i=1}^{N-1} MLP^i(\mathbf{h}_{t-i}), \tag{11}$$

where λ is a scalar value between 0 and 1. It controls the influences of future word predictions (but derived from past time-steps) on the current word prediction. Similar to the rationale behind the dominance of the original NLL loss in its total loss formulation, Eq.(6), we do not equally average the original predicted embedding with others. In the case of WDR-based N-gram CLM, we ensemble $MLP^i(\mathbf{h}_{t-i}) + \Delta^r_i\mathbf{x}^{e,l}_{t-i} = \hat{\mathbf{x}}^{e,l}_t$ in the summation part in Eq.(11).

After this ensemble computation, we input it to the logit layer and compute the next word's likelihood. At testing, this ensemble likelihood result is used to compute perplexity (PPL) in CLM tasks or serving as candidate scores for beam search in NMT tasks.

4 Experiments and Results

To assess the performances of our proposed methods, we conducted CLM and NMT experiments on multiple benchmark datasets.

4.1 Causal Language Modeling (CLM)

For the CLM task, we executed two experiments: preliminary and primary. The preliminary experiment was dedicated to monitor the dynamics of two hyperparameters: N and λ toward the performance. In contrast, we only report the results of the best hyperparameters in the primary experiment's demonstration.

4.1.1 Data Description

PTB (-, 0.9M tokens, 10K vocabulary), WikiText-2 (W2, 2M tokens, 33K vocabulary), Text8 (T8, 15M tokens, 254K vocabulary), and WikiText-103 (W103, 103M tokens, 268K vocabulary) (Mikolov et al., 2014; Merity et al., 2016). To ensure standardization and transparency in our data-related processes (e.g., download, tokenization, vocabulary, and train/valid/testsets splitting), we relied on open sources. Specifically, the W2 and T8 datasets were sourced from the GitHub repository[1], while the PTB and W103 datasets were sourced from the Tensorized Transformer (Ma et al., 2019)'s GitHub repository[2]. In the primary experiment, we used the whole datasets, whereas the preliminary experiment was conducted solely on the PTB dataset.

4.1.2 Models and Training

For the baseline model of the preliminary experiment, we implemented Transformer (TF) encoder-based CLM. The total number of parameters of the TF baseline is 12M, and our proposed simple and WDR methods increase only 0.1M parameters per an additional MLP layer (note that the logit layer's parameters are all shared). The details of model architecture and training method for the preliminary experiment are described in Table 4 (in Appendix A.2) in the column of 'Small Enc. TF CLM'.

For the baseline models of the primary experiment, we trained the two baseline models that are advanced ones based on TF: tensorized transformer (TT) (Ma et al., 2019) and Reformer (RF)[3] (Kitaev et al., 2020). We mostly followed their reported configurations, except some minor changes such as the number of tokens in a mini-batch and learning rates. The details of these changes for each dataset are described in Table 5 (in Appendix A.2). As a result, the total numbers of parameters of (TT, RF) models according to datasets are (6.7M, 15.3M) for PTB and W2, (82.4M, 236.6M) for T8 and W103, respectively. Our proposed simple and WDR methods increase the number of parameters by 0.1M and 0.5M, respectively, per an additional MLP layer regardless of the type of dataset.

On top of the baseline models, we applied our proposed method, and we call them 'TF+Sim', 'TF+WDR', 'TT+Sim', 'TT+WDR', 'RF+Sim', and 'RF+WDR'. We varied N from 2 to 4 and λ from 0.0 to 0.6 for every experiment of our proposed methods. In the demonstration of the primary experiment results, we report the result of the best hyperparameter setting of each model. These settings are reported in the 'CLM Task' column of Table 6 (in Appendix A.2).

Table 1: Word-level PPL results of the preliminary experiment with Transformer encoder-based CLMs on the PTB dataset. A different value of λ indicates the application of the proposed ensemble method with the λ value.

Model	Test PPL			
	λ=0.0	0.2	0.4	0.6
TF	161.0	-	-	-
TF+Sim N=2	150.8	134.6	135.3	156.3
N=3	153.3	134.4	133.0	151.9
N=4	158.1	133.6	129.1	147.1
TF+WDR N=2	149.0	136.5	129.8	**128.1**
N=3	153.1	136.1	128.2	128.8
N=4	150.5	131.6	124.1	127.5

4.1.3 Preliminary Experimental Results

Table 1 presents the outcomes of the preliminary experiments. We trained the model of each configuration five times with different seeds, and we report the average PPL scores. Both 'TF+Sim' and 'TF+WDR' surpass the performances of the conventional CLM baseline. This observation aligns with findings from previous studies on other tasks (Sun et al., 2019; Joshi et al., 2020; Xiao et al., 2020; Qi et al., 2020). The ensemble method consistently improves performance compared to the non-ensemble ones (where λ=0.0). It usually achieves the best scores at λ=0.4 for both the 'TF+Sim' and 'TF+WDR' models. Also, we observed that the 'TF+WDR' model maintains strong performance even at λ=0.6, while the 'TF+Sim' model does not. This implies that 'TF+WDR' generally generates more accurate predictions for future words. Moreover, 'TF+WDR' tends to outperform their 'TF+Sim' counterparts in each setting. These findings collectively suggest that the WDR training approach offers benefits over N-gram prediction methodologies.

4.1.4 Gradient Diversity Analysis

As an additional exploration of the advantages of WDR, we checked the connection between the diverse target representations and its benefit during

[1] https://github.com/chakki-works/chazutsu

[2] https://github.com/szhangtju/The-compression-of-Transformer

[3] https://github.com/lucidrains/reformer-pytorch

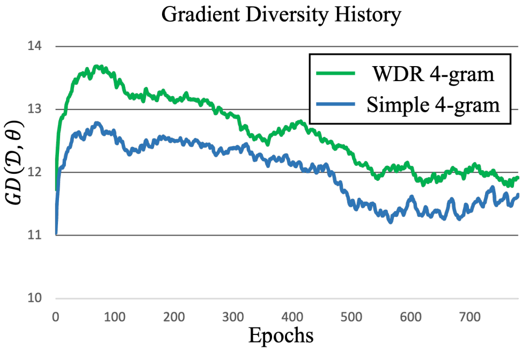


Figure 3: Gradient diversity comparison between simple 4-gram CLM and WDR 4-gram CLM.

training. Given the evidence in Sec.3.2.3 that WDR gives more diverse target representations compared to other CLMs, it is plausible to guess the backpropagated gradients are also diverse. To quantify this property, we measured '*gradient diversity (GD)*' (Yin et al., 2018) which is formulated as follows:

$$\begin{aligned} GD(\mathcal{D}, \theta) &= \frac{\sum_{i=1}^{|\mathcal{D}|} ||g_i||_2^2}{|| \sum_{i=1}^{|\mathcal{D}|} g_i||_2^2}, \\ &= \frac{\sum_{i=1}^{|\mathcal{D}|} ||g_i||_2^2}{\sum_{i=1}^{|\mathcal{D}|} ||g_i||_2^2 + \sum_{i \neq j} \langle g_i, g_j \rangle}, \quad (12) \\ g_i &= \nabla_\theta \mathcal{L}_N^{tot}(X_i, \theta), \end{aligned}$$

where $\mathcal{D} = \{X_1, X_2, \cdots, X_{|\mathcal{D}|}\}$ is a mini-batch, $|| \cdot ||_2^2$ is the squared L^2 norm operation, $\langle \cdot, \cdot \rangle$ is the inner product operation, and ∇_θ is gradient operator with respect to θ. This metric is large when the inner product terms in denominator are small, which means the gradients are different from each other.

We measured GD of the 'TF+Sim N=4' and 'TF+WDR N=4' models in Table 1 during training. The GDs over epochs are presented in Fig.3. 'TF+WDR N=4' usually has higher GD than 'TF+Sim N=4'. As the stochastic property of stochastic gradient descent is known for noisy gradient which enhances generalizability compared to full-batch gradient descent (Hardt et al., 2016; Yin et al., 2018), higher GD may offer similar advantages due to the stochastic property. Given this understanding, we believe WDR-based training could be beneficial for improving generalization.

4.1.5 Primary Experimental Results

Table 2 presents the entire results of the primary experiments (6 models on 4 datasets). Results show that, with the exception of TT-based models on W2, our proposed N-gram CLMs consistently either match or surpass the baseline CLMs, even without the ensemble method. Remarkably, WDR N-gram CLMs generally improve performance on top of the simple N-gram CLMs. Upon applying our proposed ensemble method, they generally exhibit improvements over their non-ensemble counterparts, except the models trained on W103. Notably, the effect of ensemble method is relatively significant in smaller datasets (PTB and W2) in contrast to larger datasets (T8 and W103). Based on these results, we argue that our proposed methods have actual advantages on various models and datasets for the CLM task.

Table 2: Word-level PPL results of the primary experiment. Regarding the unsatisfying PPL of 'RF (baseline)' on W103, as in the experiments on PTB, W2, and T8 datasets, we trained 'RF' on W103 based on the same provided source code with the default configuration except a few changes described in Table 5. Note that 'RF+Sim' and 'RF+WDR' models were trained under the same setting for fair comparisons.

Model	Test Word-level PPL			
	PTB	W2	T8	W103
TT (baseline)	55.0	56.1	121.4	20.1
TT+Sim	51.6	62.0	106.5	17.1
Ensemble	45.5	56.0	**89.5**	17.9
TT+WDR	47.5	57.7	91.7	**16.8**
Ensemble	**44.4**	**53.8**	90.2	16.9
RF (baseline)	28.0	31.6	64.3	50.3
RF+Sim	27.8	31.6	**62.1**	43.1
Ensemble	26.4	31.0	62.2	43.4
RF+WDR	26.0	31.5	62.2	**41.8**
Ensemble	**25.9**	**30.8**	**62.1**	41.9

Table 3: Experiment results of NMTs on several benchmark datasets. We used translations of TED and TEDx talks for IWSLT14 En-De. Also, we used Newstest18 and Newstest14 for WMT18 En-Tr and WMT14 En-De, respectively. The left and right numbers of '/' mean En-to-*(De or Tr)* and *(De or Tr)*-to-En translation results, respectively.

Model	BLEU Scores		
	IWSLT	WMT14	WMT18
TF	27.6/32.5	26.5/30.4	**11.9**/18.2
BOW NMT	27.5/32.3	26.3/30.4	**11.9**/18.3
TF+Sim	28.0/33.0	26.2/30.9	11.6/18.2
Ensemble	**28.3**/33.4	26.3/31.0	11.6/18.3
TF+WDR	27.9/33.5	**26.7**/31.1	11.8/18.5
Ensemble	**28.3/34.0**	**26.7/31.2**	**11.9/18.8**

4.2 Neural Machine Translation

4.2.1 Data Description

Since NMT includes language modeling as a part of the decoder, we view the NMT could be an appropriate additional experimental task to demonstrate the effectiveness of our proposed approach in addition to the main CLM tasks. We conducted NMT experiments on several datasets: 'IWSLT14 English-German'(En-De, 160K training pairs) (Hwang and Jeong, 2023), 'WMT14 English-German'(En-De, 3.9M training pairs) (Vaswani et al., 2017), and 'WMT18 English-Turkish' (En-Tr, 207K training pairs) (Bojar et al., 2018). We used the same preprocessing, tokenization and subword byte-pair encoding methods with (Ott et al., 2019). We used 10K, 10K, 32K most frequents subwords to organize vocabularies for datasets, respectively.

4.2.2 Models and Training

As a baseline, we used our implementation of Transformer (TF) (Vaswani et al., 2017) in the encoder-decoder architecture. We used the small Transformer for the 'IWSLT14 En-De' and 'WMT18 En-Tr' datasets, and the base Transformer for the 'WMT14 En-De' dataset. The total number of parameters of small and base TF baselines are 32M and 77M, respectively. We applied our simple and WDR N-gram CLM methods onto the decoder parts of the baselines, 'TF+Sim' and 'TF+WDR'. Each additional MLP layer in our simple and WDR methods increases the number of parameters by around 0.5M. Information about the models and how TF models are optimized can be found in the columns labeled 'Small Enc-Dec TF NMT' and 'Base Enc-Dec TF NMT' in Table 4. Also, the hyperparameters (N and λ) for 'TF+Sim' and 'TF+WDR' are described in the 'NMT Task' column of Table 6 (in Appendix A.2).

As a more closely related baseline, bag-of-words (BOW) NMT was proposed to predict the whole words in the context of the original NMT task (Ma et al., 2018). However, their approach was not applied to the TF architecture, and they evaluated the model only on the English-Chinese translation dataset of NIST. To ensure a fair comparison, we re-implemented BOW NMT based on our TF architecture and compared with our proposed method. Following their prescribed approach, we integrated the computed loss of whole words prediction into the original loss.

4.2.3 BLEU Results

Table 3 presents the experiment results of the models on each testset with SacreBLEU (Post, 2018) as the evaluation metric. Our proposed 'TF+Sim' and 'TF+WDR' models exhibit usually enhanced performances compared to the 'TF' and 'BOW NMT' baselines. 'TF+WDR' always outperforms its counterpart of 'TF+Sim'. Notably, the integration of the ensemble method from both of 'TF+Sim' and 'TF+WDR' further increases performances. Specifically, we note that 'TF+WDR' with ensemble method improved performances by 0.7 1.5 BLEU scores compared to 'TF' baseline on the both translation directions of 'IWSLT14 En-De', and German-to-English translations of 'WMT14 En-De' testsets.

To explain why N-gram prediction approaches are more effective for German-to-English translation compared to English-to-German translation in 'IWSLT14 En-De' and 'WMT14 En-De' experiments, we hypothesize that the difference in word diversity between the two languages plays a role. We analyzed the 'WMT14 En-De' training dataset (subword-level tokenized) and found that English has around 33.6K unique unigrams and 6.7M unique bigrams, while German has around 34.9K unique unigrams and 9.3M unique bigrams. This suggests that German-to-English translation might have simpler local dependencies to learn compared to English-to-German translation due to the lower number of unique bigrams. Considering simple local dependencies might lead to the over-fitting problem, we believe that this is a potential reason why N-gram prediction approaches, which can help mitigate over-fitting to local dependencies, are more effective for German-to-English translation.

5 Conclusion

In this work, we have constructed an advanced N-gram prediction framework tailored specifically to causal language modeling. In addition to the construction of this framework, our work includes the introduction of new strategies for providing diverse target representations and an ensemble method over the predicted N words. Extensive experiments on language modeling and neural machine translation have confirmed the practical benefits of the proposed method.

6 Limitations

Given the demonstrated performance improvements of the WDR-based N-gram CLM, we tried to apply the WDR method to other tasks beyond CLM, such as the MLM task. In addition to the standard loss function of MLM, which involves predicting the masked word (Devlin et al., 2018), we added new loss terms to predict n-level WDR target representations of the masked position. For this experiment, we utilized the CrammedBERT model (Geiping and Goldstein, 2023), a streamlined variant of BERT that facilitates faster pre-training while maintaining competitive performance on the GLUE benchmark. We integrated the WDR approach into this model and conducted a comparative analysis with the original CrammedBERT configuration. Further experimental details are provided in Appendix A.3.

Table 7 (in Appendix A.3) presents the results of our experiments comparing CrammedBERT and the applications of WDR models on the GLUE test set. While the application of 2-level WDR resulted in a 1.0 point increase in the average GLUE score, the performance benefits of the WDR method is less consistent across individual sub-tasks compared to the benefits observed in the CLM tasks. We attribute this result to the fundamental difference between the CLM and MLM tasks. Specifically, in MLM, when the WDR method combines the masked word embedding with the embeddings of the next words, such information is already provided as input. This partial visibility of the target representation might lead to an unexpected optimization behavior, such as the model disproportionately focusing on the right-side (future) context which is incorporated in the target, rather than considering the entire context.

Since there are prior works for N-gram prediction within the MLM framework (Sun et al., 2019; Joshi et al., 2020; Xiao et al., 2020; Qi et al., 2020), we believe we can apply the WDR method to the prior works by combining the only masked words when WDR is calculated to solve the aforementioned issue. We expect that the high gradient diversity characteristic of the WDR method may offer additional benefits to the prior MLM framework.

Acknowledgements

References

Yoshua Bengio, Réjean Ducharme, and Pascal Vincent. 2000. A neural probabilistic language model. *Advances in neural information processing systems*, 13.

Ondřej Bojar, Christian Federmann, Mark Fishel, Yvette Graham, Barry Haddow, Matthias Huck, Philipp Koehn, and Christof Monz. 2018. Findings of the 2018 conference on machine translation (WMT18). In *Proceedings of the Third Conference on Machine Translation: Shared Task Papers*, pages 272–303, Belgium, Brussels. Association for Computational Linguistics.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Abraham Frandsen and Rong Ge. 2019. Understanding composition of word embeddings via tensor decomposition. *arXiv preprint arXiv:1902.00613*.

Jonas Geiping and Tom Goldstein. 2023. Cramming: Training a language model on a single gpu in one day. In *International Conference on Machine Learning*, pages 11117–11143. PMLR.

Moritz Hardt, Ben Recht, and Yoram Singer. 2016. Train faster, generalize better: Stability of stochastic gradient descent. In *International conference on machine learning*, pages 1225–1234. PMLR.

Matthias Hartung, Fabian Kaupmann, Soufian Jebbara, and Philipp Cimiano. 2017. Learning compositionality functions on word embeddings for modelling attribute meaning in adjective-noun phrases. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, pages 54–64.

DongNyeong Heo and Heeyoul Choi. 2023. Shared latent space by both languages in non-autoregressive neural machine translation. *arXiv preprint arXiv:2305.03511*.

Soon-Jae Hwang and Chang-Sung Jeong. 2023. Integrating pre-trained language model into neural machine translation. *arXiv preprint arXiv:2310.19680*.

Mandar Joshi, Danqi Chen, Yinhan Liu, Daniel S Weld, Luke Zettlemoyer, and Omer Levy. 2020. Spanbert: Improving pre-training by representing and predicting spans. *Transactions of the association for computational linguistics*, 8:64–77.

Nikita Kitaev, Łukasz Kaiser, and Anselm Levskaya. 2020. Reformer: The efficient transformer. *arXiv preprint arXiv:2001.04451*.

Bofang Li, Aleksandr Drozd, Tao Liu, and Xiaoyong Du. 2018. Subword-level composition functions for learning word embeddings. In *Proceedings of the second workshop on subword/character level models*, pages 38–48.

Shuming Ma, Xu Sun, Yizhong Wang, and Junyang Lin. 2018. Bag-of-words as target for neural machine translation. *arXiv preprint arXiv:1805.04871*.

Xindian Ma, Peng Zhang, Shuai Zhang, Nan Duan, Yuexian Hou, Ming Zhou, and Dawei Song. 2019. A tensorized transformer for language modeling. *Advances in neural information processing systems*, 32.

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. 2016. Pointer sentinel mixture models. *arXiv preprint arXiv:1609.07843*.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

Tomas Mikolov, Armand Joulin, Sumit Chopra, Michael Mathieu, and Marc'Aurelio Ranzato. 2014. Learning longer memory in recurrent neural networks. *arXiv preprint arXiv:1412.7753*.

Myle Ott, Sergey Edunov, Alexei Baevski, Angela Fan, Sam Gross, Nathan Ng, David Grangier, and Michael Auli. 2019. fairseq: A fast, extensible toolkit for sequence modeling. *arXiv preprint arXiv:1904.01038*.

Adam Poliak, Pushpendre Rastogi, M Patrick Martin, and Benjamin Van Durme. 2017. Efficient, compositional, order-sensitive n-gram embeddings. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, pages 503–508.

Matt Post. 2018. A call for clarity in reporting BLEU scores. In *Proceedings of the Third Conference on Machine Translation: Research Papers*, pages 186–191, Belgium, Brussels. Association for Computational Linguistics.

Weizhen Qi, Yu Yan, Yeyun Gong, Dayiheng Liu, Nan Duan, Jiusheng Chen, Ruofei Zhang, and Ming Zhou. 2020. Prophetnet: Predicting future n-gram for sequence-to-sequence pre-training. *arXiv preprint arXiv:2001.04063*.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. 2019. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9.

Thijs Scheepers, Evangelos Kanoulas, and Efstratios Gavves. 2018. Improving word embedding compositionality using lexicographic definitions. In *Proceedings of the 2018 World Wide Web Conference*, pages 1083–1093.

Chenze Shao, Yang Feng, and Xilin Chen. 2018. Greedy search with probabilistic n-gram matching for neural machine translation. *arXiv preprint arXiv:1809.03132*.

Chenze Shao, Jinchao Zhang, Yang Feng, Fandong Meng, and Jie Zhou. 2020. Minimizing the bag-of-ngrams difference for non-autoregressive neural machine translation. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pages 198–205.

Yu Sun, Shuohuan Wang, Yukun Li, Shikun Feng, Xuyi Chen, Han Zhang, Xin Tian, Danxiang Zhu, Hao Tian, and Hua Wu. 2019. Ernie: Enhanced representation through knowledge integration. *arXiv preprint arXiv:1904.09223*.

Laurens Van der Maaten and Geoffrey Hinton. 2008. Visualizing data using t-sne. *Journal of machine learning research*, 9(11).

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in neural information processing systems*, 30.

Dongling Xiao, Yu-Kun Li, Han Zhang, Yu Sun, Hao Tian, Hua Wu, and Haifeng Wang. 2020. Ernie-gram: Pre-training with explicitly n-gram masked language modeling for natural language understanding. *arXiv preprint arXiv:2010.12148*.

Ruifeng Xu, Tao Chen, Yunqing Xia, Qin Lu, Bin Liu, and Xuan Wang. 2015. Word embedding composition for data imbalances in sentiment and emotion classification. *Cognitive Computation*, 7:226–240.

Dong Yin, Ashwin Pananjady, Max Lam, Dimitris Papailiopoulos, Kannan Ramchandran, and Peter Bartlett. 2018. Gradient diversity: a key ingredient for scalable distributed learning. In *International Conference on Artificial Intelligence and Statistics*, pages 1998–2007. PMLR.

A Appendix

A.1 Proof of Eq.(9)

We provide a proof of Eq.(9) with the induction method. To avoid confusion, we temporarily change the notation of $\Delta_n \mathbf{x}_t^e$ in conjecture Eq.(9) to $\hat{\Delta}_n \mathbf{x}_t^e$ until it is proved. Based on the definitions of the 1 and n-level WDR, Eq.(7) and Eq.(8), we can verify the initial condition, that is $n = 1$, holds as follows:

$$\begin{aligned}
\Delta_1 \mathbf{x}_t^e &= \mathbf{x}_{t+1}^e - \mathbf{x}_t^e \\
&= \binom{1}{0}(-1)^0 \mathbf{x}_{t+1}^e + \binom{1}{1}(-1)^1 \mathbf{x}_t^e \\
&= \sum_{i=0}^{1} \binom{1}{i}(-1)^i \mathbf{x}_{t+(1-i)}^e \\
&= \hat{\Delta}_1 \mathbf{x}_t^e.
\end{aligned}$$

Therefore, the conjecture holds for the initial condition. Then, by following the induction method, we assume the conjecture at n-level is true, that is $\hat{\Delta}_n \mathbf{x}_t^e = \Delta_n \mathbf{x}_t^e$. Then, the $(n+1)$-level WDR from the definition Eq.(8) is derived to $\Delta_{n+1} \mathbf{x}_t^e = \Delta_n \mathbf{x}_{t+1}^e - \Delta_n \mathbf{x}_t^e = \hat{\Delta}_n \mathbf{x}_{t+1}^e - \hat{\Delta}_n \mathbf{x}_t^e$. Each term is derived as follows:

$$\begin{aligned}
\hat{\Delta}_n \mathbf{x}_{t+1}^e &= \binom{n}{0}(-1)^0 \mathbf{x}_{t+n+1}^e + \binom{n}{1}(-1)^1 \mathbf{x}_{t+n}^e + \\
&\quad \cdots + \binom{n}{n-1}(-1)^{n-1} \mathbf{x}_{t+2}^e + \binom{n}{n}(-1)^n \mathbf{x}_{t+1}^e, \\
-\hat{\Delta}_n \mathbf{x}_t^e &= \binom{n}{0}(-1)^1 \mathbf{x}_{t+n}^e + \binom{n}{1}(-1)^2 \mathbf{x}_{t+n-1}^e + \\
&\quad \cdots + \binom{n}{n-1}(-1)^n \mathbf{x}_{t+1}^e + \binom{n}{n}(-1)^{n+1} \mathbf{x}_t^e, \\
\hat{\Delta}_n \mathbf{x}_{t+1}^e - \hat{\Delta}_n \mathbf{x}_t^e &= \binom{n}{0}(-1)^0 \mathbf{x}_{t+n+1}^e + \left(\binom{n}{0} + \binom{n}{1}\right)(-1)^1 \mathbf{x}_{t+n}^e + \\
&\quad \cdots + \left(\binom{n}{n-1} + \binom{n}{n}\right)(-1)^n \mathbf{x}_{t+1}^e + \binom{n}{n}(-1)^{n+1} \mathbf{x}_t^e \\
&= \binom{n+1}{0}(-1)^0 \mathbf{x}_{t+n+1}^e + \binom{n+1}{1}(-1)^1 \mathbf{x}_{t+n}^e + \\
&\quad \cdots + \binom{n+1}{n}(-1)^n \mathbf{x}_{t+1}^e + \binom{n+1}{n+1}(-1)^{n+1} \mathbf{x}_t^e \\
&= \sum_{i=0}^{n+1} \binom{n+1}{i}(-1)^i \mathbf{x}_{t+(n+1-i)}^e \\
&= \hat{\Delta}_{n+1} \mathbf{x}_t^e.
\end{aligned}$$

Note that the binomial coefficient, $\binom{n}{i}$, is the n-th row and i-th value of Pascal's triangle, and it satisfies $\binom{n}{i-1} + \binom{n}{i} = \binom{n+1}{i}$. Based on this outcome, the conjecture holds for $(n+1)$-level if the n-level is true. Therefore, the conjecture is proved.

A.2 Experiment Details

We trained the models described in Sec. 4.1.2 and Sec. 4.2.2 following the configurations described in Table 4 for Transformer-based models, 'TF', and the configurations reported in the previous works' papers (Ma et al., 2019; Kitaev et al., 2020) with several changes as described in Table 5 for the primary CLM baselines, 'TT' and 'RF'. For Transformer-based models' experiments, we saved the best checkpoint based on the validation results. We early stopped the training whenever the model does not beat its

Table 4: Model and optimizer configurations of Transformer architectures used in the preliminary experiment of CLM and NMT tasks. We used the same notation for model configurations as in (Vaswani et al., 2017), except the number of layers (# of Layers) and multi-head attention's heads (# of Heads). 'ISRS' means the inverse square root learning rate scheduler (Ott et al., 2019) and '# of Tokens' indicates the total number of tokens in a mini-batch at each iteration.

Config.	Small Enc. TF CLM	Small Enc-Dec TF NMT	Base Enc-Dec TF NMT
d_{model}	256	512	512
d_{ff}	2100	1024	2048
$d_k = d_v$	64	64	64
P_{drop}	0.3	0.3	0.1
ϵ_{ls}	0.1	0.1	0.1
# of Layers	6	6	6
# of Head	4	4	8
Optimizer	Adam	Adam	Adam
Learning Rate	0.00025	0.0005	0.001
Scheduler	None	ISRS	ISRS
# of Tokens	4K	4K	25K
Patience	50	50	50

Table 5: Changed configurations from the original Tensorized Transformer and Reformer (Ma et al., 2019; Kitaev et al., 2020). We note that '# of Tokens' indicates the total number of tokens in a mini-batch at each iteration.

Dataset	Tensorized Transformer			Reformer	
	# of Tokens	# of Layers	Learning Rate	# of Tokens	Learning Rate
PTB	3,840	3	0.0025	16,384	0.0001
WikiText-2	3,840	3		8,192	
Text8	4,800	6		512	
WikiText-103	4,800	6		512	

Table 6: Configurations of our proposed N-gram approaches: N and λ, used in the primary experiments of the CLM task and experiments of the NMT task.

CLM Task						NMT Task				
Model	Config.	Dataset				Model	Config.	Dataset		
		PTB	W2	T8	W103			IWSLT14	WMT14	WMT18
TT+Sim	N/λ	2/0.2	4/0.2	3/0.2	2/0.1	TF+Sim	N	3	2	2
TT+WDR	N/λ	2/0.4	4/0.3	3/0.1	2/0.1		λ	0.3	0.1	0.2
RF+Sim	N/λ	4/0.2	2/0.2	3/0.1	4/0.1	TF+WDR	N	3	2	2
RF+WDR	N/λ	4/0.1	2/0.3	3/0.1	4/0.1		λ	0.5	0.1	0.3

Table 7: Experiment results of MLMs on the GLUE task. We used the same metrics with (Geiping and Goldstein, 2023) for each sub-task in GLUE.

Model	MNLI	SST-2	STSB	RTE	QNLI	QQP	MRPC	CoLA	GLUE Avg.
CrammedBERT	78.5/79.0	**90.0**	82.3	**57.4**	85.7	85.7	85.2	28.5	74.6
+1-level WDR	78.3/**79.2**	88.2	80.0	54.2	85.9	85.7	84.4	30.3	74.0
+2-level WDR	78.6/79.1	88.4	**82.4**	55.2	85.5	85.8	**86.9**	**38.6**	**75.6**
+3-level WDR	**78.8**/79.1	89.0	81.8	56.3	**86.4**	**85.9**	85.6	32.8	75.1

previous best performance for the 'Patience' times on the validation (Heo and Choi, 2023). For the primary CLM baselines, we followed the pre-defined total training iterations. Table 6 describes the specific configurations, such as N and λ, we used for our proposed N-gram CLMs, simple-based and WDR-based.

About the information of our computational environment, we used a single NVIDIA RTX3090 GPU for the large CLM datasets, such as T8 and W103, and a GTX1080Ti GPU for the small CLM datasets, such as PTB and W2. On average, they took 1 day and 3 hours, respectively, for training. We used 4x NVIDIA RTX3090 GPUs for the large NMT datasets, such as WMT14 English-German, and 2x GTX1080Ti GPUs for the small NMT datasets, such as IWSLT14 English-German and WMT18 English-Turkish. On average, they took 3 days for training.

A.3 Masked Language Modeling Experiment

We adhered to the environmental settings established by CrammedBERT (Geiping and Goldstein, 2023) for all aspects of our study, including dataset preprocessing, model configurations, pre-training, fine-tuning procedures, and evaluations. Comprehensive details of these settings can be found in the associated GitHub repository[4]. Building on the CrammedBERT architecture, we apply the WDR method that is analogous to the method conducted in our WDR-based N-gram CLM experiment. Specifically, we utilized N additional MLP layers designed to predict n-level WDRs alongside the original word embedding at the masked position. These n-level WDRs are calculated by composing the next words of the masked word. The final loss is computed as the average of the original loss and the additional losses derived from the WDR method, with the original and additional losses being averaged unequally, as described in Section 3.1.

Table 7 presents the experimental results for CrammedBERT and our proposed models, evaluated on the GLUE test set following fine-tuning. We varied the number of grams, N, from 1 to 3. The results indicate that the application of 2-level WDR yields an increase of 1.0 point in the average GLUE score. However, the performance improvements across individual sub-tasks are not consistently superior; in some cases, they were similar to or worse than the baseline.

[4] https://github.com/JonasGeiping/cramming