Diverse Target Representations for Language Models with Word Difference Representations

DongNyeong Heo, Daniela Noemi Rim, Heeyoul Choi

Proceedings of the 39th Pacific Asia Conference on
Language, Information and Computation (PACLIC 39)

Emmanuele Chersoni, Jong-Bok Kim (eds.)

2025

Diverse Target Representations for Language Models with Word Difference Representations

DongNyeong Heo
Handong Global University
Pohang, South Korea
sjglsks@gmail.com

Daniela Noemi Rim
Handong Global University
Pohang, South Korea
danielarim@handong.ac.kr

Heeyoul Choi
Handong Global University
Pohang, South Korea
hchoi@handong.edu

Abstract

Language modeling (LM) serves as the foundational framework underpinning remarkable successes of recent text generation tasks. In this paper, we study the potential benefits of incorporating diverse and contextualized target representations during LM training, in contrast to conventional approaches that rely on fixed target representations for words. We hypothesize that the use of diverse target representations can enhance the generalizability of LM training. To examine this hypothesis, we introduce the word difference representation (WDR) transformation function, which provides diverse target representations for words in LM. In addition, we propose a simple N-gram prediction framework and an ensemble method to facilitate the WDR approach, and manifest the potential of the WDR approach. Through extensive experiments across various model architectures—including large language models and diffusion models—and multiple benchmark datasets, we empirically validate our hypothesis and demonstrate the practical benefits of our proposed methodologies in terms of improved text generation performance.

1 Introduction

With the remarkable advancements in deep learning techniques (Radford et al., 2019; Brown et al., 2020), language modeling (LM) has become a central component in text generation tasks, such as neural machine translation (NMT) and question answering chatbots. In general, an LM model processes given context words through a neural network to output a representation. The inner product between this representation and the weight matrix of the final logit layer produces scores for each word (Bengio et al., 2000). During training, in order to achieve a high score for the target word after the inner product, the neural network is optimized to output a representation similar to the target word's assigned representation in the vector space of the logit layer's weights.

The assigned representation of each word solely represents the target of the word. In other words, the LM task is a sequential prediction of the unique target representation of each word within a sentence. In contrast, tasks such as image generation that has almost infinite variety of target image representations, the variety of the target text representation is limited to the vocabulary size regardless of the context. Based on this limitation, we pose the following question: *"Would it be beneficial to the training process if we* ***provide different target representations*** *for the same word depending on the context?"*. Regarding the practical benefit of this question, we expect that incorporating diverse target representations would lead to greater diversity in gradient computations during backpropagation. Based on prior analysis (Yin et al., 2018), the increased gradient diversity can enhance the generalizability of the model.

In this paper, we explore the question above by proposing a contextual transformation function - word difference representation (WDR). WDR contextualizes subsequent words (N-gram) through an arithmetic subtraction operation, and we incorporate WDRs as additional, diverse target representations alongside the original ones. To verify the effectiveness of the WDR approach in contrast to baseline approaches, we first develop a simple N-gram prediction framework with minimal modifications to the conventional LM model, and then we apply the WDR approach to the simple N-gram framework. Furthermore, we propose an ensemble method that leverages N-gram predictions to enhance next-word prediction, which is the primary objective of the conventional LM model.

We experimented with our proposed methodologies across multiple baseline model architectures, such as conventional Transformer-based models (Vaswani et al., 2017), large language models (GPT-

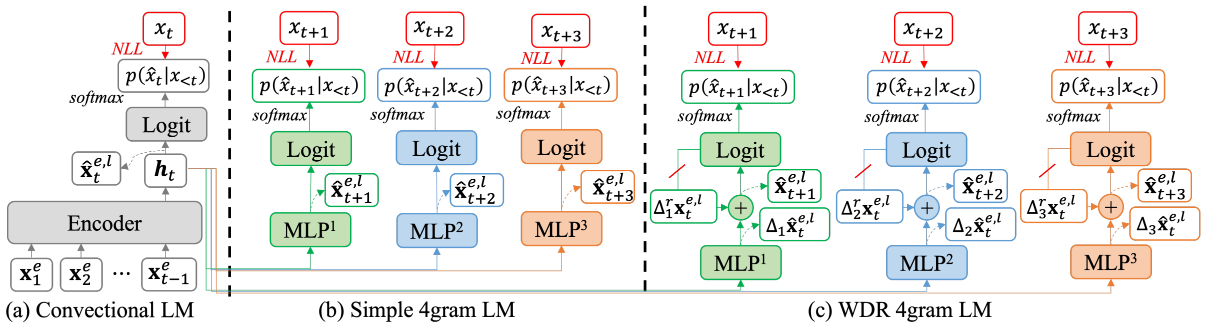

Figure 1: Model illustrations of (a) conventional LM, (b) simple N-gram LM, and (c) WDR N-gram LM when $N = 4$. Note that all of the drawn logit layers above the MLP layers share the parameters. Red diagonal lines in (c) on lines from logit layer to $\Delta_i^r \mathbf{x}_t^{e,l}$ indicate gradient detaching operation.

NEO series) (Black et al., 2021), and the text diffusion model (Ye et al., 2023; Gao et al., 2022), using multiple benchmark datasets for LM and various conditional text generation tasks. In the experiment results, we empirically validate that applying WDR enhances gradient diversity and boosts performance while causing only a slightly increased parameter count and computational cost. These findings provide evidence that incorporating diverse target representations is beneficial for general text generation tasks. Additionally, our results indicate that our ensemble method is also advantageous compared to conventional LM models.

2 Background

2.1 Language Modeling

As background knowledge, in this section we describe the conventional training framework of neural network based LMs.

A sentence consists of words $X = \{x_1, x_2, \cdots, x_T\}$, where $x_t \in \mathcal{V}$, and T and $\mathcal{V}$ indicate the length of the sentence and the vocabulary set, respectively. Conventional LMs compute the likelihood of a word conditioned on its preceding words in the sentence, $p(x_t|x_{<t})$.

First, 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, and d is the dimension of the encoded hidden state and the embedding vector spaces. Recently, most LMs use Transformer (Vaswani et al., 2017) as their encoder architecture. After encoding, the hidden state is linearly transformed to a logit value of each word in $\mathcal{V}$. Finally, the likelihood of the predicted word is calculated as follows:

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

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

Throughout this paper, we consider the logit layer's weight matrix as a word embedding set, $\mathbf{W}^l = [\mathbf{x}_1^{e,l}, \mathbf{x}_2^{e,l}, \cdots, \mathbf{x}_{|\mathcal{V}|}^{e,l}]^\top$, where each d-dimensional embedding vector is aligned with the target word. The superscript notation (e, l) means that it is an embedding vector at the logit layer. From this point of view, the hidden state, $\mathbf{h}_t$, could be understood as the predicted word embedding, $\hat{\mathbf{x}}_t^{e,l}$. The inner product of $\mathbf{W}^l$ and $\hat{\mathbf{x}}_t^{e,l}$ outputs the predicted score of each word based on the similarity between the embedding and predicted vectors.

Then, 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 process is illustrated in Fig.1(a).

2.2 N-gram Prediction

The conventional LM training framework follows the '*next word prediction*' approach that predicts a word given the whole previous words. Despite the successes, this approach might lead models to overfit to local dependencies rather than capturing long-term dependencies between words. This tendency arises from the strong dependencies found in some phrases or word pairs, 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 during the pre-training phase of large language models (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). To utilize the N-gram prediction method, previous works significantly modified the model architecture, the loss function, or the vocabulary set. In this paper, as the development base of our main idea (diverse target representations with WDR), we introduce a 'simple N-gram prediction' method that requires the least modifications from the conventional LM so that is transparent to analyze the main idea's advantage.

3 Proposed Methods

In this section, we introduce all of our proposed methodologies: (1) a simple N-gram prediction framework that becomes the development's base, (2) the main WDR idea and its application to several LM architectures, and (3) ensemble method applicable to N-gram prediction.

3.1 Simple N-gram Prediction

The core idea of our proposed simple N-gram prediction method is adding a multi-layer perception (MLP) layer to predict a future word's embedding (logit layer's) given the same hidden state of the conventional LM. This process is formulated as follows:

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

For instance, assuming N is 4, three MLP layers, MLP^1, MLP^2 and MLP^3, are employed to predict $\hat{\mathbf{x}}^{e,l}_{t+1}$, $\hat{\mathbf{x}}^{e,l}_{t+2}$, and $\hat{\mathbf{x}}^{e,l}_{t+3}$ respectively, as shown in Fig.1(b). Note that the conventional LM model predicts $\hat{\mathbf{x}}^{e,l}_{t}$, so a total 4-gram words are predicted. Then, we compute the likelihoods of the future target words, $p(\hat{x}_{t+1}|x_{<t};\theta)$, $p(\hat{x}_{t+2}|x_{<t};\theta)$, and $p(\hat{x}_{t+3}|x_{<t};\theta)$, following each logit layer and the softmax function. Instead of using separate logit layers for each future word prediction, we share the parameters across all logit layers, including the conventional LM model's original logit layer. Therefore, this approach increases just a small amount of parameters for each additional MLP layer. Finally, the individual word prediction loss and the total loss are formulated as follows:

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

$$\mathcal{L}^{tot}_N = \frac{1}{2}\mathcal{L}_0 + \frac{\alpha}{2N-2}\sum_{n=1}^{N-1}\mathcal{L}_n, \tag{6}$$

where $\alpha < 1$ is a hyperparameter to control the additional loss term's influence. Note that $\mathcal{L}_0$ is the same as the conventional LM's loss, Eq.(3).

The framework of the simple N-gram prediction method is quite similar to the recent speculative decoding approaches proposed by (Gloeckle et al., 2024; Cai et al., 2024) which also add additional heads on top of the conventional LM's encoder to predict future words with shared encoder and logit layer like our method. As independently developed, however, there are several differences. First, we employ an MLP layer with ReLU activation expecting that the limited capacity of the MLP layer appropriately regularizes the main encoder, Enc_θ, to find simultaneously informative hidden states for all N-gram predictions. Second, since the next word typically has stronger dependencies with the preceding words than the other future words do, we multiply the original loss with only $1/2$ while we multiply the future words' losses with $\frac{1}{2N-2}$ and the scalar hyperparameter α in Eq.(6).

3.2 Word Difference Representation (WDR)

In this section, we explain the concept of WDR and how to provide diverse target representations with WDR to simple N-gram prediction LMs and diffusion model-based LMs (Li et al., 2022; Gao et al., 2022; Ye et al., 2023).

3.2.1 Definition of n-level WDR

As we mentioned, WDR is a function that transforms subsequent N words' embedding vectors into a contextualized form. WDR's transformation is based on a form of word embedding compositions: the difference vector, $\mathbf{x}^e_{t+1} - \mathbf{x}^e_t$ as its name implies. Since (Mikolov et al., 2013) demonstrated that arithmetic compositions of learned word embedding can convey semantic meanings, many researchers 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 employed composed word embeddings as model inputs instead of original word embeddings. In contrast, we use the composed word embeddings as the target representation.

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:

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

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

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

Based on the definitions of Eqs. (7) and (8), the n-level WDR can be represented by a linear combination 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. In 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 $t \leq T - n$. See Appendix A.1 for a proof of this equation. From this equation, n-level WDR can be easily reconstructed to the original word embedding. For example, 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:

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

which leads to $\Delta_n\mathbf{x}^e_t + \Delta^r_n\mathbf{x}^e_t = \mathbf{x}^e_{t+n}$.

The subtracting operations of generating n-level WDRs, Eq.(8), and their reconstruction process (note the reconstruction conjugate term, Eq.(10), is the subtraction of the original embedding and n-level WDR) are parallelizable, so that they do not impose computational overhead in long sequence.

3.2.2 WDR on Simple N-gram LM

The application of WDR to simple N-gram LM (we call it 'WDR N-gram LM') follows the idea of adding additional MLP layers and predicting future words. However, in WDR N-gram LM, 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 reconstruction conjugate term, $\Delta^r_n\mathbf{x}^{e,l}_t$, based on the logit layer's embedding matrix and target words. 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 LM. Then, we take the processes of the logit, likelihood, and loss computations as in the simple N-gram LM.

An essential design of this framework is detachment of the produced reconstruction conjugate term, $\Delta^r_n\mathbf{x}^{e,l}_t$, during the backpropagation process. The 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 will be recursively produced from itself.

Unlike the simple N-gram LM, the individual NLL loss, Eq.(5), decreases when the model predicts more accurate n-level WDR, $\Delta_n\hat{\mathbf{x}}^{e,l}_t$, because it will output more accurate future word's embedding after adding the reconstruction conjugate term $\Delta_n\hat{\mathbf{x}}^{e,l}_t + \Delta^r_n\mathbf{x}^{e,l}_t = \hat{\mathbf{x}}^{e,l}_{t+n}$. Since the reconstruction conjugate term 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 LM's training framework can train LM with diverse and contextualized target representations given the same target word. The entire process of WDR 4-gram LM example is illustrated in Fig.1(c).

3.2.3 WDR on Diffusion-based LM

Since their great success in image generative models (Ho et al., 2020; Ramesh et al., 2022), diffusion models have been applied to text generation tasks (Li et al., 2022). However, compared to image diffusion models, text diffusion models' performances are less impressive, and some prior studies have investigated the reasons behind the limited progress of text diffusion models, particularly focusing on the diffusion and denoising processes on the word embedding space (Ye et al., 2023). A common argument from their analyses starts from the discreteness of the embedding vectors which form a finite number of clusters within the high-dimensional embedding space. Since the small noises of early diffusion steps are not significant enough to move an embedding vector from one cluster to another, the denoising process that trains the model becomes trivial. We conjecture that providing diverse target representations can explicitly mitigate this problem because diversifying word embedding vectors will reduce the discreteness.

Given this perspective, we apply our WDR idea to the diffusion-based LM, such as DINOISER (Ye et al., 2023) and Difformer (Gao et al., 2022). As in the application of WDR to simple N-gram LM, we add extra heads to predict WDRs beside the original head that predicts the original embedding given t-times diffused embeddings. Similarly, we compute the reconstruction conjugate term from the shared logit layer with detachment, and then we reconstruct the original embedding. Finally, the diffusion loss for each head's prediction is formulated as follows (the original diffusion loss is when $n = 0$):

$$\mathcal{L}_n^d = \sum_{t=n+1}^{T} \| \hat{\mathbf{x}}_{t,0}^{e,l} - \mathbf{x}_{t,0}^{e,l} \|_2^2, \tag{11}$$

$$\hat{\mathbf{x}}_{t,0}^{e,l} = \begin{cases} f(\mathbf{x}_{t,k}^{e,l};\theta) & \text{if } n = 0, \\ MLP^n(f(\mathbf{x}_{t,k}^{e,l};\theta)) + \Delta_n^r \mathbf{x}_{t,0}^{e,l} & \text{if } n > 0, \end{cases} \tag{12}$$

where $f(\mathbf{x}_{t,k}^{e,l};\theta)$ is the diffusion model's output given k-times diffused embedding vector, $\mathbf{x}_{t,k}^{e,l}$, with parameter set, θ. Analogously to the total loss of simple N-gram LM, we average the original and additional diffusion losses. See the previous works (Ye et al., 2023; Gao et al., 2022) for more details on the diffusion and denoising processes as well as the final loss. We followed their methodologies except for the changes to apply the WDR.

3.3 Ensemble Method to Leverage N-gram Predictions for Next Word Prediction

In this section, we propose a new ensemble method to incorporate the N-gram predictions into the process of the next word prediction. The encoded hidden state $\mathbf{h}_t$ represents the computed hidden state given the inputs up to time-steps $(t-1)$. During testing, in addition to the predicted embedding $\hat{\mathbf{x}}_t^{e,l}$ from the conventional LM, the MLP^n layer of simple N-gram LM can estimate the target word for time t given $\mathbf{h}_{t-n}$. In total, 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:

$$\hat{\mathbf{x}}_{t,ens}^{e,l} = (1-\lambda)\hat{\mathbf{x}}_t^{e,l} + \frac{\lambda}{N-1}\sum_{n=1}^{N-1} MLP^i(\mathbf{h}_{t-n}), \tag{13}$$

where λ is a scalar value between 0 and 1, which 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 N-gram or diffusion LM, we ensemble $MLP^i(\mathbf{h}_{t-n}) + \Delta_n^r \mathbf{x}_{t-n}^{e,l} = \hat{\mathbf{x}}_t^{e,l}$ in the summation part in Eq.(13).

After this ensemble computation, we feed it to the logit layer and compute the next word's likelihood. During testing, this ensemble likelihood result is used to compute perplexity (PPL) in LM tasks or serving as candidate scores for beam search in NMT tasks. It is natural to expect that our ensemble method is beneficial if the additional heads predict independent (and effective) information from the original heads.

Table 1: Word-level PPL results of the preliminary experiment with Transformer decoder-based LMs on the PTB dataset. We tried several λ values in Eq.(13).

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

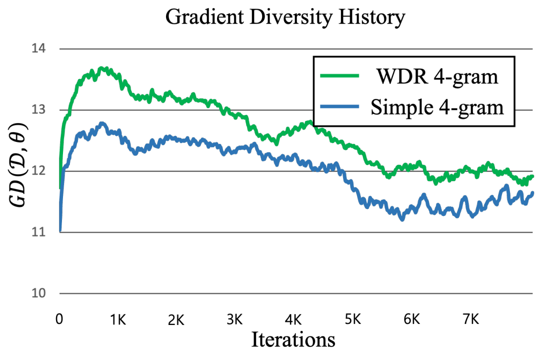

Figure 2: Gradient diversity comparison between simple 4-gram LM and WDR 4-gram LM.

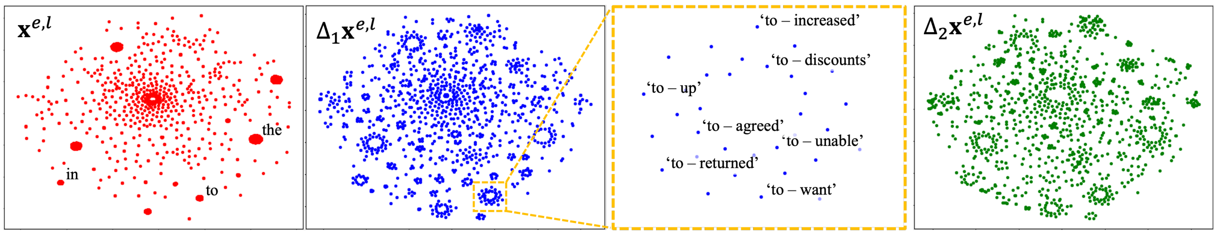

Figure 3: From the left-to-right, they are visualizations of the original embeddings (first), 1-level WDR and the plot zoomed in around the original word 'to' (second and third), and 2-level WDR (last), respectively. In the third plot, ('to-*word*') means the 1-level WDR vector, that is $\mathbf{x}^{e,l}_{to} - \mathbf{x}^{e,l}_{word}$ based on the word '*to*' fragment within the sentence.

4 Experiments and Results

We conducted preliminary analysis and main experiments. During the preliminary analysis, we trained the basic LM model on the Penn TreeBank (PTB) dataset (Marcus et al., 1993) to verify the expectations of our proposed methodologies. During the main experiments, we conducted LM and various conditional text generation tasks with multiple benchmark datasets and multiple baseline models, including an LLM, to demonstrate practical advantages of our proposed methodologies. Due to the page limit, we describe the details of the dataset, model architecture, and training scheme in Appendices A.2 and A.3.

4.1 Preliminary Analysis

As a preliminary analysis, we trained Transformer (TF) decoder-based LM models by applying simple N-gram and WDR N-gram ('TF+Sim' and 'TF+WDR') on the PTB dataset. During testing, we applied our ensemble method with varying the λ value. The total number of parameters of the TF baseline is 12M, and our proposed simple and WDR methods increase only 0.1M parameters for additional MLP layer. The details of the model architecture and training method for this experiment are described in Table 6 in Appendix A.3 as 'Small Dec. TF LM'. Also refer to Section A.2 for the details of PTB data preprocessing.

4.1.1 Perplexity of the Models

Table 1 presents the outcomes of the experiments. We trained the model of each configuration five times with different random seeds, and we report the average PPL scores. As demonstrated in previous N-gram prediction approaches (Sun et al., 2019; Joshi et al., 2020; Xiao et al., 2020; Qi et al., 2020; Gloeckle et al., 2024), both 'TF+Sim' and 'TF+WDR' outperform the conventional LM baseline. More interestingly, we found that 'TF+WDR' is generally better than 'TF+Sim' in various N settings. Our ensemble method consistently improves performance compared to the non-ensemble ones (where λ=0.0). More importantly, 'TF+WDR' models bring greater improvements with the ensemble method than 'TF+Sim'. For example, 'TF+WDR' models' improvements with the ensemble method is 24.07 on average, while 'TF+Sim' models' is 21.83. We interpret that WDR method trained the additional heads to predict more independent and effective information.

4.1.2 Comparison of Gradient Diversity

As mentioned in the introduction section, diverse target representations can bring a higher gradient diversity during training. To verify this expectation, during training, we measured '*gradient diversity (GD)*' (Yin et al., 2018) as follows:

$$\begin{aligned} GD(\mathcal{D},\theta) &= \frac{\sum_{i=1}^{|\mathcal{D}|} ||g_i||_2^2}{||\sum_{i=1}^{|\mathcal{D}|} g_i||_2^2}, \\ &= \frac{\sum_{i=1}^{|\mathcal{D}|} ||g_i||_2^2}{\sum_{i=1}^{|\mathcal{D}|} ||g_i||_2^2 + \sum_{i\neq j}\langle g_i, g_j\rangle}, \quad (14) \\ g_i &= \nabla_\theta \mathcal{L}_N^{tot}(X_i,\theta), \end{aligned}$$

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 the denominator are small, which means the gradients are different from each other.

Fig.2 demonstrates the GD history of 'TF+Sim N=4' and 'TF+WDR N=4' models during training. 'TF+WDR N=4' generally received higher GD than 'TF+Sim N=4'. As the stochastic property of stochastic gradient descent is known to enhance generalizability compared to full-batch gradient descent (Hardt et al., 2016; Yin et al., 2018), higher GD may offer similar advantages due to higher stochasticity. Given this understanding, we believe that WDR-based training could be beneficial to improve generalizability.

4.1.3 Visualization of the Representations

To gain a more profound understanding of WDR's effect on target representations, we collected 1,270 actual target representations of the conventional LM model's training, which are the logit layer's embedding vectors corresponding to target words from the PTB 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 the t-SNE algorithm (Van der Maaten and Hinton, 2008).

Fig.3 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 (e.g., 'in', 'to', and 'the') may be included more times than other words in the collection. We interpret that this is the reason why t-SNE places frequent words 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 than the 1-level WDR as in the last plot. Based on this analysis, we believe that WDR N-gram LM actually receives diverse target representations.

4.2 Language Modeling Experiments

Our main LM experiments consist of training conventional LM models on multiple benchmark datasets and fine-turning the pre-trained LLM. We trained Tensorized Transformer (TT) (Ma et al., 2019) and Reformer (RF) (Kitaev et al., 2020) models on PTB, WikiText-2 (W2), Text8 (T8), and WikiText-103 (W103) datasets (Mikolov et al., 2014; Merity et al., 2016) for the conventional LM experiments. For the fine-tuning experiments, we fine-tuned GPTNEO 1.3B model (Black et al., 2021) on the W2 dataset. We note that our simple and WDR methods increased around 3.43% of total parameters out of the baselines' on average. For more details, refer to Appendices A.2 and A.3.

Table 2: Results of the conventional LM experiments.

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: Results of LLM fine-tuning experiments.

Model	WikiText-2 PPL (w/ Ens.)
GPTNEO 1.3B	13.25
GPTNEO 1.3B + Sim	13.13 (12.93)
GPTNEO 1.3B + WDR	**13.00** (**12.91**)

Table 2 presents the entire results of the conventional LM experiments. The results show that, with the exception of TT-based models on W2, our proposed N-gram LM models consistently either match or surpass the baselines, even without the ensemble method. Remarkably, WDR N-gram LM models generally improve performance on top of the simple N-gram LM models. 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 the ensemble method is relatively significant in the smaller datasets (PTB and W2) rather than the larger datasets (T8 and W103).

Table 3 presents the results of our LLM fine-tuning experiments. Each configured model was trained three times with different random seeds. Notably, our WDR model outperformed the simple baseline. Specifically, the average PPL of WDR model, 13.00, surpasses the 95% confidence interval of the simple model's result, which is 13.02 (without ensemble). Furthermore, we observed that our ensemble method generally leads to improved performance. Based on these LM results, we argue that providing diverse target representations can offer advantages over the conventional reliance on fixed target representations alone.

Table 4: NMT results of conventional Transformer models on several benchmark datasets. 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
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**

Table 5: Results of WDR applications on diffusion models: DINOISER and Difformer.

Baseline Arch.	Task	BLEU Scores	
		Baseline	+WDR
DINOISER	IWSLT14 En2De	25.76	26.26
	IWSLT14 De2En	31.26	31.83
Difformer	QQP	28.62	29.73
	WikiAuto	34.33	37.53

4.3 Conditional Text Generation Experiments

To further investigate the benefits of our proposed methodologies, we conducted multiple conditional text generation tasks, which can be regarded as conditional language modeling tasks. Our experiments are divided into two categories: (1) training conventional Transformer models on NMT datasets such as IWSLT14 En-De (Hwang and Jeong, 2023), WMT14 En-De (Vaswani et al., 2017), and WMT18 En-Tr (Bojar et al., 2018); and (2) training text diffusion models such as DINOISER and Difformer on IWSLT14 En-De, QQP (text paraphrasing), and WikiAuto (text simplification) (Gao et al., 2022). We note that both the simple and WDR methods increased the total number of parameters by approximately 2.45% over the baselines on average. For further details on models and datasets, see Appendices A.2 and A.3.

Table 4 presents the conventional Transformer-based NMT experiment results based on SacreBLEU (Post, 2018) evaluation metric. While simple N-gram method, 'TF+Sim', is sometimes worse than 'TF' baseline, WDR N-gram method, 'TF+WDR', always outperforms or is similar to the baseline. Notably, the integration of the ensemble method from either of 'TF+Sim' or 'TF+WDR' further increases performances. Specifically, we note that 'TF+WDR' with ensemble method improves performances by 0.7$\sim$1.5 BLEU scores compared to 'TF' baseline on both translation directions of 'IWSLT14 En-De', and German-to-English translations of 'WMT14 En-De' testsets.

The N-gram prediction approaches are more effective for De-En translation compared to En-De translation in 'IWSLT14 En-De' and 'WMT14 En-De' experiments. We believe that the difference in word variety between the two languages plays a key 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 De-En translation might have simpler local dependencies to learn compared to En-De 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 De-En translation.

Table 5 presents the experimental results of baseline diffusion models and our WDR models ('+WDR'), evaluated using BLEU scores. Note that we did not apply the simple N-gram method, since generic text diffusion models operate in a non-autoregressive decoding manner, predicting all words simultaneously. By applying our WDR method, we aim to leverage diverse target representations. Notably, our WDR models improved BLEU scores by 0.50, 0.57, 1.11, and 3.20 on the IWSLT14 En2De, IWSLT14 De2En, QQP, and WikiAuto datasets, respectively. Considering the results across all conditional text generation tasks, our WDR approach is also beneficial for a wide range of practical conditional LM tasks.

5 Limitations

During the hyperparameter search in the preliminary analysis (Sec. 4.1), we observed that when N is larger than 4, both simple and WDR N-gram LMs consistently performed worse. We hypothesize that predictions far into the future are too difficult to learn, and thus may have regularized the encoder in a disadvantageous way. Unfortunately, our WDR method could not overcome this limitation under the current experimental setting. In future work, we aim to develop a novel approach that leverages diverse target representations to address the issues associated with large N, or an alternative approach that does not rely on the N-gram prediction framework.

6 Conclusion

In this work, we explored the potential of using contextualized target representations for training language models. We proposed WDR function, which transforms word fragments into contextualized forms and uses them as auxiliary target representations alongside the original targets. Building on our simple N-gram prediction framework, we applied WDR and validated its practical advantages across various models and datasets in both language modeling and conditional language modeling tasks. We found that applying WDR increases gradient diversity, which in turn improves generalization and overall performance.

Acknowledgements

This work was supported by Institute of Information & communications Technology Planning & Evaluation (IITP) grant funded by the Korea government(MSIT) (No.RS-2025-25443930, Development of Human-Level Robot Intelligence Capable of Real-Time Environmental Adaptation), and by the MSIT(Ministry of Science, ICT), Korea, under the Global Research Support Program in the Digital Field program(RS-2024-00431394) supervised by the IITP(Institute for Information & Communications Technology Planning & Evaluation)

References

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

Sid Black, Gao Leo, Phil Wang, Connor Leahy, and Stella Biderman. 2021. Gpt-neo: Large scale autoregressive language modeling with mesh-tensorflow.

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, and 1 others. 2020. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901.

Tianle Cai, Yuhong Li, Zhengyang Geng, Hongwu Peng, Jason D Lee, Deming Chen, and Tri Dao. 2024. Medusa: Simple llm inference acceleration framework with multiple decoding heads. *arXiv preprint arXiv:2401.10774*.

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*.

Zhujin Gao, Junliang Guo, Xu Tan, Yongxin Zhu, Fang Zhang, Jiang Bian, and Linli Xu. 2022. Empowering diffusion models on the embedding space for text generation. *arXiv preprint arXiv:2212.09412*.

Fabian Gloeckle, Badr Youbi Idrissi, Baptiste Rozière, David Lopez-Paz, and Gabriel Synnaeve. 2024. Better & faster large language models via multi-token prediction. *arXiv preprint arXiv:2404.19737*.

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*.

Jonathan Ho, Ajay Jain, and Pieter Abbeel. 2020. Denoising diffusion probabilistic models. *Advances in neural information processing systems*, 33:6840–6851.

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.

Xiang Li, John Thickstun, Ishaan Gulrajani, Percy S Liang, and Tatsunori B Hashimoto. 2022. Diffusion-lm improves controllable text generation. *Advances in neural information processing systems*, 35:4328–4343.

Ilya Loshchilov and Frank Hutter. 2017. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*.

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.

Mitchell P. Marcus, Mary Ann Marcinkiewicz, and Beatrice Santorini. 1993. Building a large annotated corpus of english: The penn treebank. *Comput. Linguist.*, 19(2):313–330.

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, and 1 others. 2019. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9.

Aditya Ramesh, Prafulla Dhariwal, Alex Nichol, Casey Chu, and Mark Chen. 2022. Hierarchical text-conditional image generation with clip latents. *arXiv preprint arXiv:2204.06125*, 1(2):3.

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.

Jiasheng Ye, Zaixiang Zheng, Yu Bao, Lihua Qian, and Mingxuan Wang. 2023. Dinoiser: Diffused conditional sequence learning by manipulating noises. *arXiv preprint arXiv:2302.10025*.

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:

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

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 Dataset Details

In this section, we describe the details of the datasets that we utilized in LM and conditional text generation experiments (Sections 4.2 and 4.3).

A.2.1 Language Modeling Dataset Description

For the training of conventional LM models, we utilized four datsets, such as 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). We followed the open sources for data-related processes (e.g., download, tokenization, vocabulary, and train/valid/test sets splitting). 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]. For the fine-tuning LLM task, we utilized W2 dataset with different data processes based on GPTNEO models' open source from Huggingface[3].

A.2.2 Conditional Text Generation Dataset Description

For the training of the conventional Transformer model for NMT, we utilized three datasets, such as '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 collected 10K, 10K, 32K most frequent subwords to organize vocabularies for datasets, respectively. For test sets, 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. For the IWSLT14 En-De dataset used for the training of DINOISER model, we downloaded the open preprocessed dataset of the Github[4] as the DINOISER baseline used. For the QQP (145K training pairs,

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

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

[3]https://huggingface.co/EleutherAI/gpt-neo-1.3B

[4]https://github.com/shawnkx/Fully-NAT#dataset

Table 6: Model and optimizer configurations of Transformer architectures used in the preliminary experiment of LM and NMT tasks. We used the same notation for model configurations as in (Vaswani et al., 2017), except for 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 Dec. TF LM	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

text paraphrasing) and WikiAuto (678K training pairs, text simplification) datsaets that we used for the training Difformer models, we mainly followed the instructions of their official Github[5].

A.3 Model and Training Details

We explain the details of the model and training scheme that were used in our experiments (Sections 4.2 and 4.3) in this section.

A.3.1 Language Modeling Experiments

For the models and training of the conventional LM experiments, including 'TT', 'RF' baselines, and applications of our simple and WDR N-gram LM models, we followed the configurations reported in the previous works' papers (Ma et al., 2019; Kitaev et al., 2020) with several changes as described in Table 7. The total parameters of (TT, RF) baselines according to datasets are (6.7M, 15.3M) for PTB and W2, (82.4M, 236.6M) for T8 and W103, respectively. Both of our proposed simple and WDR methods increased the number of parameters by 0.1M for TT and 0.5M for RF per an additional MLP layer regardless of the type of dataset. The optimal hyperparameter settings of our proposed methods were found after the hyperparameter search. They are described in Table 8. Refer to the previous works for all of the details. The experiments of small datasets, PTB and W2, took around 3 hours on average based on a single GTX1080Ti GPU, while the experiments of large datasets, T8 and W103, took around 24 hours in average based on a single RTX3090 GPU. Unexpectedly, we found that the PPL of 'RF (baseline)' on W103 in Table 2 is unsatisfying compared to other results of PTB, W2, and T8 datasets. We trained 'RF' on W103 based on the same provided source code with the default configuration except for a few changes described in Table 7. Note that 'RF+Sim' and 'RF+WDR' models were trained under the same setting for fair comparisons.

For the fine-tuning of GPTNEO 1.3B pre-trained model, we downloaded the pre-trained models and their tokenizers with Huggingface API. We fully fine-tuned the pre-trained models on W2 dataset for 3 epochs with 8 batch size, AdamW optimizer (Loshchilov and Hutter, 2017), 1^{-5} initial learning rate, and linearly decreasing learning rate schedule (with 1K warmup). For our simple and WDR N-gram applications, we mainly followed the same training scheme of MEDUSA's fine-tuning (Cai et al., 2024). We shortly fine-tuned only for the additional heads before the main full fine-tuning because randomly initialized parameters of the additional heads on top of the pre-trained model can cause unstable training. Our simple and WDR applications increase the number of parameters by 8.39M per an additional MLP layer which are 0.65% out of the 1.3B total parameters. Similar to the conventional LM experiments, we conducted the hyperparameter search and we report the resulting optimal settings in Table 9. The fine-tuning experiments with baseline GPTNEO 1.3B pre-trained model took around 22.25 hours on average based on a single RTX3090 GPU. Our simple and WDR applications took 28.03 and 29.35 hours on average, respectively. We note that WDR method increased the total training times with 4.71% on top of the simple N-gram LM models.

A.3.2 Conditional Text Generation Experiments

We implemented the encoder-decoder Transformers (Vaswani et al., 2017) in different scales, small and base. We trained the small Transformer for the 'IWSLT14 En-De' and 'WMT18 En-Tr' datasets, and the base Transformer for the 'WMT14 En-De' dataset. Model and training configurations of these

[5]https://github.com/zhjgao/difformer

Table 7: 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 8: Configurations of our proposed N-gram approaches: N, α, and λ, used in the conventional LM and NMT experiments.

LM Task						NMT Task				
Model	Config.	Dataset				Model	Config.	Dataset		
		PTB	W2	T8	W103			IWSLT14	WMT14	WMT18
TT+Sim	$N/\alpha/\lambda$	2/1.0/0.2	4/1.0/0.2	3/1.0/0.2	2/1.0/0.1	TF+Sim	$N/\alpha/\lambda$	3/1.0/0.3	2/1.0/0.1	2/1.0/0.2
TT+WDR	$N/\alpha/\lambda$	2/1.0/0.4	4/1.0/0.3	3/1.0/0.1	2/1.0/0.1					
RF+Sim	$N/\alpha/\lambda$	4/1.0/0.2	2/1.0/0.2	3/1.0/0.1	4/1.0/0.1	TF+WDR	$N/\alpha/\lambda$	3/1.0/0.5	2/1.0/0.1	2/1.0/0.3
RF+WDR	$N/\alpha/\lambda$	4/1.0/0.1	2/1.0/0.3	3/1.0/0.1	4/1.0/0.1					

Table 9: Configurations of our proposed N-gram approaches: N, α, and λ, used in the LLM fine-tuning experiments.

Model	Config.	Value
GPTNEO 1.3B + Sim	$N/\alpha/\lambda$	2/0.10/0.08
GPTNEO 1.3B + WDR	$N/\alpha/\lambda$	4/0.10/0.04

Table 10: Configurations of our proposed N-gram approaches: N, α, and λ, used in the text diffusion model experiments.

Dataset	Config.	Value
IWSLT14 En2De	$N/\alpha/\lambda$	2/0.1/0.1
IWSLT14 De2En	$N/\alpha/\lambda$	3/0.1/0.1
QQP	$N/\alpha/\lambda$	3/0.5/0.2
WikiAuto	$N/\alpha/\lambda$	4/1.0/0.1

models are described in 'Small Enc-Dec TF NMT' and 'Base Enc-Dec TF NMT' columns of Table 6. The total number of parameters of small and base Transformer baselines are 32M and 77M, respectively. We applied our simple and WDR N-gram LM methods onto the decoder part. Each additional MLP layer in our simple and WDR methods required the number of parameters by around 0.5M. After hyperparameter search, we determined the optimal hyperparameters, N, α, and λ, and those are described in the 'NMT Task' column of Table 8. During training, we saved the best checkpoint based on the validation results. We early stopped the training whenever the model did not beat its previous best performance for the 'Patience' times on the validation (Heo and Choi, 2023). The experiments of small datasets, such as IWSLT14 En-De and WMT18 En-Tr, took 3 days in average based on 2 GTX1080Ti GPUs, while the experiments of large dataset, that is WMT14 En-De, took 3 days on average based on 4 RTX3090 GPUs. During testing, we applied beam search with 5 widths to output the final translation results.

For the experiments of text diffusion model, we heavily followed the model, diffusion process, and training configurations of the previous works (Ye et al., 2023; Gao et al., 2022). We refer to previous works for that information. Consequently, the total number of parameters of DINOISER(IWSLT14) / Difformer(QQP) / Difformer(WikiAuto) are 37M/109M/112M, respectively. As explained in Section 3.2.3, we added additional heads for WDR predictions on top of the denoising model's encoder (Transformer encoder). Notably, we did not apply the simple N-gram method, since diffusion-based text generative models usually follow non-autoregressive decoding so there is no need for future word prediction. Each added head increases by around 0.5M. We also conducted a hyperparameter search for WDR-related configurations, then we determined the optimal hyperparameters as demonstrated in Table 10. These experiments took 30 hours in average based on a single A4000 GPU.