A Robust Framework for Improving Next-Word Prediction in Neural Language Models

Nguyen Cong Binh[1], Phung Tuan Kiet[1], Dang Minh Vu[1], Nguyen Bao Ngoc Trieu Anh[1], and Nguyen Viet Hung[1,*]

[1]International Training and Cooperation Institute, East Asia University of Technology, Bacninh, VietNam.
{binhnc, 20232355, 20233641, 25000381, hungnv}@eaut.edu.vn
*Corresponding author: Nguyen Viet Hung. E-mail: hungnv@eaut.edu.vn

Abstract. This paper investigates the effectiveness of standard decoder-only autoregressive Transformers under a simplified optimization framework, in comparison with a JEPA-based approach for next-word prediction. We adopt full-parameter optimization with a standard next-token prediction objective and introduce an adaptive regularization strategy to handle datasets with varying levels of structural noise. Experiments on four benchmark datasets, Synth, Spider, GSM8K, and Turk, show that the proposed framework consistently outperforms the LLM-JEPA baseline in terms of prediction accuracy and convergence speed. Our findings suggest that conventional autoregressive Transformers, when trained under appropriate optimization settings, can remain competitive for structured and semi-structured text learning tasks.

Keywords: Next-Word Prediction · Transformer · Full-Parameter Optimization · Adaptive Dropout

1 Introduction

Natural Language Processing (NLP) is key to artificial intelligence and helps machines understand, interpret and generate human language. Important NLP tasks such as language modeling and sentiment analysis support many applications, including opinion mining, social media analysis, chatbot systems, and misinformation detection [1–3].

Early approaches to NLP relied on symbolic and statistical methods, where words were treated as discrete units. These traditional techniques suffer from fundamental limitations, including sparse representations, combinatorial explosion of word sequences, and limited ability to capture semantic relationships. Such issues are especially pronounced in social media data, which is typically informal, noisy, and highly context-dependent.

Recent studies have demonstrated that combining ontology-based knowledge representation with NLP techniques can significantly improve sentiment understanding in social media contexts. For Vietnamese social networks, ontology-driven sentiment analysis has been shown to enhance interpretability and classification accuracy by incorporating domain-specific semantic relationships [2].

Similar findings have been reported for English social media comments, where ontological structures help capture nuanced sentiment expressions beyond surface-level lexical cues [1]. These works highlight the effectiveness of structured semantic knowledge in addressing linguistic ambiguity in sentiment analysis.

In parallel, opinion mining has been applied to conversational systems, such as chatbots, to enable automated analysis of user feedback and attitudes. Nguyen et al. [3] developed a chatbot-based framework for analyzing opinions in English comments, demonstrating the practicality of sentiment-aware conversational agents in real-world applications. Furthermore, deep learning approaches have been successfully employed for related NLP tasks in Vietnamese, including fake information detection, where neural models effectively capture complex semantic patterns in noisy textual data [4].

While these studies confirm the effectiveness of ontology-based and deep learning approaches, most existing systems depend on predefined feature pipelines or task-specific architectures. As a result, they often struggle to generalize across domains and modeling long-range dependencies in free-form, noisy text.

Similar challenges, including annotation noise, structural variation, and long-range dependencies, also arise in reasoning-focused tasks such as mathematical problem solving, SQL generation, and regular expression synthesis. In these tasks, next-word prediction must generate sequences that are not only meaningful but also syntactically correct and logically consistent.

Motivated by these observations, this work studies a standard autoregressive Transformer-based framework for next-word prediction. While recent methods such as LLM-JEPA [5] introduce joint embedding objectives and parameter-efficient tuning using Low-Rank Adaptation (LoRA) [6], we focus on a simpler approach based only on a standard next-token prediction objective with full-parameter optimization. Our goal is to examine whether low-rank updates limit the model's ability to learn strict syntactic and grammatical rules required for structured output generation.

This paper presents three main findings. First, we show that full-parameter optimization consistently outperforms LoRA-based tuning on structured reasoning tasks. Second, we demonstrate that a standard next-token prediction objective combined with regex-based vocabulary constraints achieves higher accuracy and faster convergence than more complex joint-embedding approaches. Finally, we introduce an *Adaptive Dropout* strategy that adjusts regularization strength based on dataset noise characteristics.

2 Related Work

Next-word prediction is a fundamental task in language modeling, aiming to predict the next token given previous context. Early neural models such as RNNs and LSTMs improved sequence modeling but struggled with long-range dependencies and efficient parallel training [7]. The Transformer architecture [8], based on self-attention, addressed these limitations and has become the dominant model for next-word prediction. Most modern large language models use an autoregressive decoder-only Transformer trained with a next-token prediction objective, forming the basis of widely used models such as GPT-style models [9, 10], PaLM [11], and LLaMA [12].

Structured and reasoning-oriented datasets are commonly used to evaluate next-word prediction beyond plain text. The **Synth** and **Turk** datasets [13] pair natural language descriptions with regular expressions using synthetic and human-annotated data. **GSM8K** [14] evaluates mathematical reasoning and requires accurate prediction over multi-step solutions. **Spider** [15] focuses on text-to-SQL generation, where outputs must be both syntactically and semantically correct. These benchmarks emphasize strict syntax, long-range dependencies, and robustness to noise, and are widely used to assess reasoning and structured generation in language models [16, 17].

Recent work has explored extensions to standard autoregressive training to improve reasoning and generalization. Wei et al. [18] demonstrate benefits of explicitly modeling intermediate steps. Zhuang et al. [19] modify the training process by masking input tokens, encouraging broader context use while maintaining a decoder-only architecture. Other studies explore alternative training objectives and adaptation strategies beyond standard next-token prediction. Huang et al. [5] combine next-token prediction with a joint embedding objective and apply parameter-efficient tuning using LoRA [6]. While these approaches report performance gains, the added objectives and constraints may limit flexibility when learning strict syntactic and grammatical rules.

Although methods such as LLM-JEPA [5] report strong performance on structured reasoning benchmarks, they combine next-token prediction with auxiliary objectives and parameter-efficient tuning. As a result, there has been limited analysis of standard autoregressive Transformer models trained solely with a next-token prediction objective under carefully controlled optimization and regularization settings. Our work addresses this gap by systematically evaluating decoder-only Transformers with full-parameter optimization on Synth, Turk, GSM8K, and Spider, using LLM-JEPA as a strong reference baseline.

3 Proposed Method

3.1 Research Methodology

Our objective is to improve next-word prediction accuracy through an empirical optimization framework. Specifically, we adopt a standard decoder-only Transformer model and focus on optimization and training strategies that better suit structured and noisy data.

We avoid low-rank adaptation (LoRA) methods here. While LoRA saves memory, we empirically find that limiting the trainable parameters can limit the model's ability to learn the rigid syntactic rules required for our structured datasets. By updating every weight in the network, we prioritize prediction accuracy over parameter efficiency.

3.2 System Architecture

Based on the decoder-only Transformer, we designed our system, named *Transformer-NWP*. Rather than introducing architectural modifications, we focus on selecting a compact yet expressive configuration that fits our computational constraints. The final model configuration is determined based on preliminary experiments and is described below.

The Pipeline. To precisely define the data flow, we formalize the Transformer-NWP processing logic in Algorithm 1. This algorithm transforms the discrete input tokens into a probability distribution through a strictly causal sequence.

Algorithm 1 Transformer-NWP Forward Pass

```
 1: Input: Context sequence $X = \{x_1, x_2, \ldots, x_L\}$ (where $L = 100$)
 2: Parameter: $d_{model} = 256, N = 3, h = 4$
 3: Output: Probability distribution $P_{next}$
 4: $E \leftarrow$ Embedding($X$) + PositionalEncoding($X$)
 5: for $i \leftarrow 1$ to $N$ do                    ▷ Process through 3 Decoder Layers
 6:     $H_{attn} \leftarrow$ MaskedMultiHeadAttention($E$)
 7:     $E' \leftarrow$ LayerNorm($E$ + AdaptiveDropout($H_{attn}$))
 8:     $H_{ffn} \leftarrow$ FeedForward($E'$)
 9:     $E \leftarrow$ LayerNorm($E'$ + AdaptiveDropout($H_{ffn}$))
10: end for
11: $Z \leftarrow$ LinearProjection($E$)
12: $P_{next} \leftarrow$ Softmax($Z$)
13: return $P_{next}$
```

Training Focus. Unlike the baseline which tries to align embeddings and generate text at the same time, we focus our entire training signal on a single objective: minimizing the negative log-likelihood of the correct sequence.

$$\mathcal{L}_{total} = -\sum_{t=1}^{T} \log P(x_t | x_{<t}; \theta) \tag{1}$$

This singular focus helps the model converge faster on algorithmic tasks, as evaluated in Section 4.

3.3 Adaptive Dropout

One of the biggest challenges in this work was the large differences between the datasets we analyzed. Some, like *Synth* or *Spider*, are very logical and rigid. Others, like *Turk*, are highly noisy and unstructured. A single regularization setting didn't work for both.

To solve this, we implemented an Adaptive Dropout mechanism. Instead of using a single fixed dropout rate, we set the dropout level based on the dataset type:

$$p_{drop} = \begin{cases} \approx 0.1 & \text{for rigid tasks (Logic/Math)} \\ \approx 0.9 & \text{for noisy tasks (Human Chat)} \end{cases} \tag{2}$$

The dropout values were selected through a linear sweep over candidate rates and finalized based on validation performance. We use a low dropout for logic-based tasks to keep correct syntax, and much higher dropout for noisy human data to reduce memorization of random patterns. This adaptive setting improves the training stability.

4 Evaluation

In this section, we give a detailed evaluation of the Transformer model trained under our *Transformer-NWP* framework. We will describe the experimental setup and data organization, followed by a performance comparison against the *LLM-JEPA* baseline.

4.1 Experimental Settings

Datasets and Adaptive Strategy. To test the framework thoroughly, we selected four different datasets, each representing a different level of difficulty. Instead of applying a uniform regularization strategy, we used an **Adaptive Dropout** strategy based on the noise level of each dataset.

We benchmark our proposed framework against the *LLM-JEPA* architecture, which represents a strong baseline for next-word prediction. As established in [5], LLM-JEPA serves as a formidable baseline, having demonstrated superior performance over traditional approaches, and we follow its original training protocol without adaptive regularization. Comparing against this architecture provides a focused assessment of our framework under competitive settings.

The specific characteristics of the selected datasets, along with the corresponding dropout rates determined in our implementation, are summarized in Table 1.

Data Engineering Pipeline. We did not directly feed raw text into the model. Instead, we implemented a strict preprocessing pipeline to ensure high-quality input:

- **Linearization:** We flattened structured JSON data by replacing newlines and tabs with spaces. This removes formatting bias.
- **Regex Whitelisting:** We used Regular Expressions to strip invisible characters, keeping only alphanumeric characters. This reduces the vocabulary size and results in a more concentrated probability distribution.

Table 1. Dataset Specifications and Adaptive Dropout Strategy

Dataset	**Description & Challenge**	**Dropout Rate**
GSM8K [14]	Math problems requiring step-by-step reasoning. Needs strict logic retention.	0.1 (Low)
Spider [15]	SQL code generation. Tests the model's ability to follow rigid syntax rules.	0.4 (Medium)
Synth [13]	Artificial data with strict logical patterns. Tests pure pattern recognition.	0.65 (High)
Turk [13]	Real-world human instructions. Highly unstructured and noisy.	0.916 (Extreme)

We formulated the problem as a standard next-token prediction task using a **Sliding Window**. For a sequence of tokens, the model uses a window of 100 consecutive tokens (X_i) to predict the next one (Y_i):

$$X_i = (t_i, t_{i+1}, \ldots, t_{i+L-1}), \quad Y_i = (t_{i+1}, t_{i+2}, \ldots, t_{i+L}) \tag{3}$$

where i indexes the sliding window position along the token sequence.

Architecture and Hyperparameters. We implemented a **decoder-only Transformer** architecture. Unlike recurrent architectures that process tokens sequentially, our model uses *Multi-Head Self-Attention* to capture global context within each layer.

For optimization, we implemented **Mixed Precision (FP16)** to reduce memory usage by 50% and used the **AdamW** optimizer for stable convergence.

Table 2. Detailed Hyperparameter Configuration

Hyperparameter	**Value / Setting**
Model Type	Transformer Decoder (GPT)
Embedding Dimension (d_{model})	256
Number of Layers (N)	3
Attention Heads (h)	4
Feed-Forward Dimension (d_{ff})	512
Sequence Length (L)	100
Optimizer	AdamW ($\lambda = 1 \times 10^{-4}$)
Learning Rate	3×10^{-4}
Batch Size	64
Precision	Mixed Float16
Dropout Rate	Adaptive (0.1–0.916)

Following the protocol established in [5], we trained the model for 6 epochs. To make the results more reliable and keep the experiments fair, we ran each experiment 5 times using different random seeds.

4.2 Experimental Results

We evaluated the performance of our proposed *Transformer-NWP* framework by comparing it to the *LLM-JEPA* baseline. We mainly focused on two aspects: final accuracy and the speed of training convergence.

Accuracy and Convergence Analysis. Fig. 1 shows the prediction accuracy. The results show that our method performs better than the baseline:

- On the **Synth** dataset, Transformer-NWP achieved approximately **88%** accuracy, compared to 72% for LLM-JEPA.
- On the noisy **Turk** dataset, our framework showed strong robustness, reaching approximately **57%** accuracy versus 32% for the baseline.

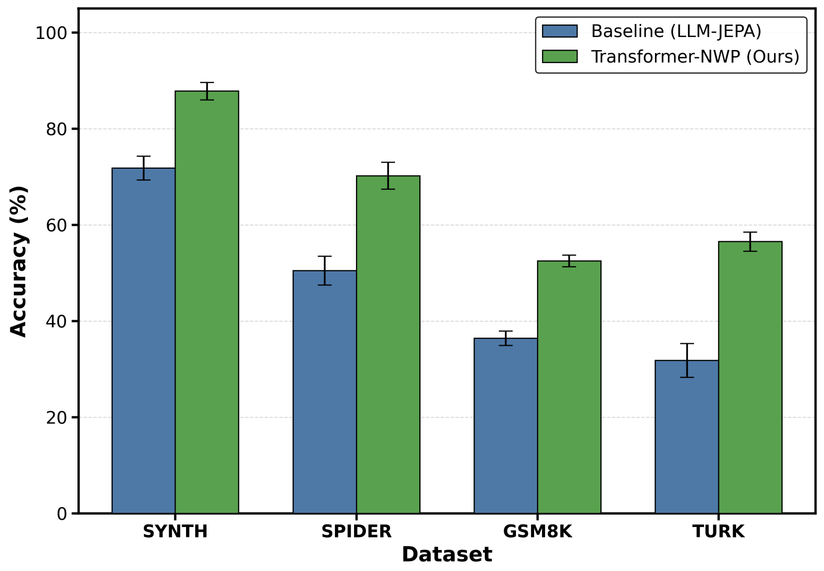

Fig. 1. Comparative analysis of accuracy between the Baseline (LLM-JEPA) and Transformer-NWP.

The learning curves (Fig. 2) confirm the efficiency of our proposed framework, showing rapid convergence under *Transformer-NWP* in the initial epochs. Even with extreme dropout on the *Turk* dataset, our method maintains steady improvement.

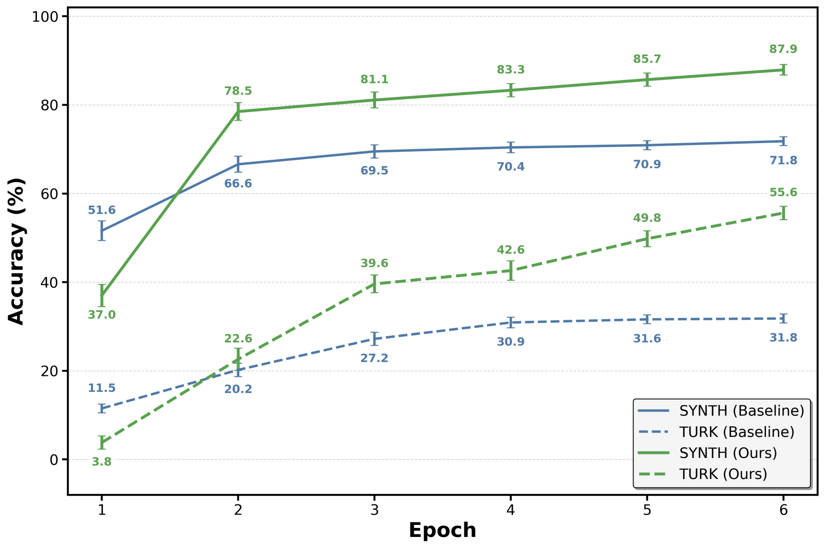

Fig. 2. Comparison of training convergence dynamics between Transformer-NWP and LLM-JEPA on the Synth and Turk datasets.

4.3 Analysis of Convergence Behavior

To explain why Transformer-NWP performs better, we analyze the technical differences:

Unconstrained vs. Low-Rank Updates. The LLM-JEPA baseline uses Low-Rank Adaptation (LoRA), which constrains learning to a lower-dimensional subspace. This means the baseline can only update a small part of the parameters. Our method, however, updates all model parameters, which can enable faster error minimization in practice:

$$\theta_{\text{new}} = \theta_{\text{old}} - \eta \nabla_\theta \mathcal{L} \tag{4}$$

where θ denotes the model parameters, η is the learning rate, and $\mathcal{L}$ is the training loss. By updating the full parameter set θ, we avoid the optimization constraints imposed by non-trainable parameters in the baseline, allowing greater flexibility during learning.

Gradient Fidelity. The LLM-JEPA baseline optimizes two objectives simultaneously: next-word prediction and embedding alignment. This creates an optimization conflict, effectively diluting the learning focus. In contrast, our method focuses entirely on a single objective, predicting the correct sequence, resulting in a clearer training signal. The loss function is:

$$\mathcal{L} = -\sum_t \log P(x_t \mid x_{<t}) \tag{5}$$

This results in a more distinct and effective gradient signal, allowing the model to converge more quickly to higher accuracy.

Simpler Vocabulary. By using our Regex Whitelisting, we reduced the choices the model has to make. Instead of selecting from a large subword vocabulary, it focuses on a small set of characters. This reduction in output space decreases prediction uncertainty and enables faster convergence during training.

5 Conclusions

This work examined next-word prediction using a standard decoder-only autoregressive Transformer trained with a simplified optimization framework. Rather than introducing a new architecture, we focused on how training decisions affect model behavior on structured and noisy tasks.

Our experiments indicate that full-parameter optimization combined with adaptive dropout leads to faster convergence and more stable learning than the LLM-JEPA baseline. These results suggest that auxiliary objectives and low-rank constraints are not always necessary for strong performance.

In summary, conventional autoregressive Transformers remain a competitive choice for structured and semi-structured text generation when trained under carefully designed optimization settings.

References

1. N. V. Hung, N. Tan, N. T. T. Nga, L. H. Trang, T. T. Hang *et al.*, “Using ontology to analyze english comments on social networks,” *Informatics and Automation*, vol. 23, no. 5, pp. 1311–1338, 2024.
2. N. V. Hung, N. A. Quan, N. Van Vu, P. T. Yen, N. H. Binh, and N. T. T. Nga, “Using ontology to analyze sentiment of comments on vietnamese social media,” 2024.
3. H. Nguyen, N. Tan, N. Quan, T. Huong, N. Phat *et al.*, “Building a chatbot system to analyze opinions of english comments,” *Informatics and Automation*, vol. 22, no. 2, pp. 289–315, 2023.
4. N. Hung, T. Loi, N. Huong, T. Hang, T. Huong *et al.*, “Aafndl-an accurate fake information recognition model using deep learning for the vietnamese language,” *Informatics and Automation*, vol. 22, no. 4, pp. 795–825, 2023.
5. H. Huang, Y. LeCun, and R. Balestriero, “Llm-jepa: Large language models meet joint embedding predictive architectures,” 2025. [Online]. Available: https://arxiv.org/abs/2509.14252
6. E. J. Hu, Y. Shen, P. Wallis, Z. Allen-Zhu, Y. Li, S. Wang, L. Wang, and W. Chen, “Lora: Low-rank adaptation of large language models,” 2021. [Online]. Available: https://arxiv.org/abs/2106.09685
7. Y. Goldberg, *Neural network methods for natural language processing.* Springer Cham, 2017, in Synthesis lectures on human language technologies. [Online]. Available: https://doi.org/10.1007/978-3-031-02165-7

8. A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," 2023. [Online]. Available: https://arxiv.org/abs/1706.03762
9. T. B. Brown, B. Mann, N. Ryder, M. Subbiah, J. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell *et al.*, "Language models are few-shot learners," in *Proceedings of the 34th International Conference on Neural Information Processing Systems*, 2020.
10. L. Ouyang, J. Wu, X. Jiang, D. Almeida, C. Wainwright, P. Mishkin, C. Zhang, S. Agarwal, K. Slama, A. Ray *et al.*, "Training language models to follow instructions with human feedback," 2022. [Online]. Available: https://arxiv.org/abs/2203.02155
11. A. Chowdhery, S. Narang, J. Devlin, M. Bosma, G. Mishra, A. Roberts, P. Barham, H. W. Chung, C. Sutton *et al.*, "Palm: Scaling language modeling with pathways," 2022. [Online]. Available: https://arxiv.org/abs/2204.02311
12. H. Touvron, T. Lavril, G. Izacard, X. Martinet, M.-A. Lachaux, T. Lacroix, B. Rozière, N. Goyal, E. Hambro, F. Azhar, A. Rodriguez, A. Joulin, E. Grave, and G. Lample, "Llama: Open and efficient foundation language models," 2023. [Online]. Available: https://arxiv.org/abs/2302.13971
13. N. Locascio, K. Narasimhan, E. DeLeon, N. Kushman, and R. Barzilay, "Neural generation of regular expressions from natural language with minimal domain knowledge," in *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, J. Su, K. Duh, and X. Carreras, Eds. Austin, Texas: Association for Computational Linguistics, Nov. 2016, pp. 1918–1923.
14. K. Cobbe, V. Kosaraju, M. Bavarian, M. Chen, H. Jun, L. Kaiser, M. Plappert, J. Tworek, J. Hilton, R. Nakano, C. Hesse, and J. Schulman, "Training verifiers to solve math word problems," 2021. [Online]. Available: https://arxiv.org/abs/2110.14168
15. T. Yu, R. Zhang, K. Yang, M. Yasunaga, D. Wang, Z. Li, J. Ma, I. Li, Q. Yao, S. Roman, Z. Zhang, and D. Radev, "Spider: A large-scale human-labeled dataset for complex and cross-domain semantic parsing and text-to-SQL task," in *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, E. Riloff, D. Chiang, J. Hockenmaier, and J. Tsujii, Eds. Brussels, Belgium: Association for Computational Linguistics, Oct.-Nov. 2018, pp. 3911–3921. [Online]. Available: https://aclanthology.org/D18-1425/
16. C. Raffel, N. Shazeer, A. Roberts, K. Lee, S. Narang, M. Matena, Y. Zhou, W. Li, and P. J. Liu, "Exploring the limits of transfer learning with a unified text-to-text transformer," 2023. [Online]. Available: https://arxiv.org/abs/1910.10683
17. Y. Tay, M. Dehghani, V. Q. Tran, X. García, J. Wei, X. Wang, H. W. Chung, D. Bahri, T. Schuster, H. S. Zheng, D. Zhou, N. Houlsby, and D. Metzler, "Ul2: Unifying language learning paradigms," in *International Conference on Learning Representations*, 2022.
18. J. Wei, X. Wang, D. Schuurmans, M. Bosma, b. ichter, F. Xia, E. Chi, Q. V. Le, and D. Zhou, "Chain-of-thought prompting elicits reasoning in large language models," in *Advances in Neural Information Processing Systems*, S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh, Eds., vol. 35. Curran Associates, Inc., 2022, pp. 24 824–24 837.
19. X. Zhuang, Z. Jia, J. Li, Z. Zhang, L. Shen, Z. Cao, and S. Liu, "Mask-enhanced autoregressive prediction: Pay less attention to learn more," 2025. [Online]. Available: https://arxiv.org/abs/2502.07490