Coupling RNN with LLM: Does Their Integration Improve Highly Order-Sensitive Language Understanding?

Md Mostafizer Rahman[1, 2], Ariful Islam Shiplu[3], Yutaka Watanobe[4], Syed Rameez Naqvi[1], Lu Peng[1]

[1]Tulane University, New Orleans, USA
[2]Lucy Family Institute for Data & Society, University of Notre Dame, IN, USA
[3]Dhaka University of Engineering & Technology, Gazipur, Bangladesh
[4]University of Aizu, Aizuwakamatsu, Japan
mrahman9@tulane.edu, shipluarifulislam@gmail.com, yutaka@u-aizu.ac.jp, snaqvi@tulane.edu, lpeng3@tulane.edu

Abstract

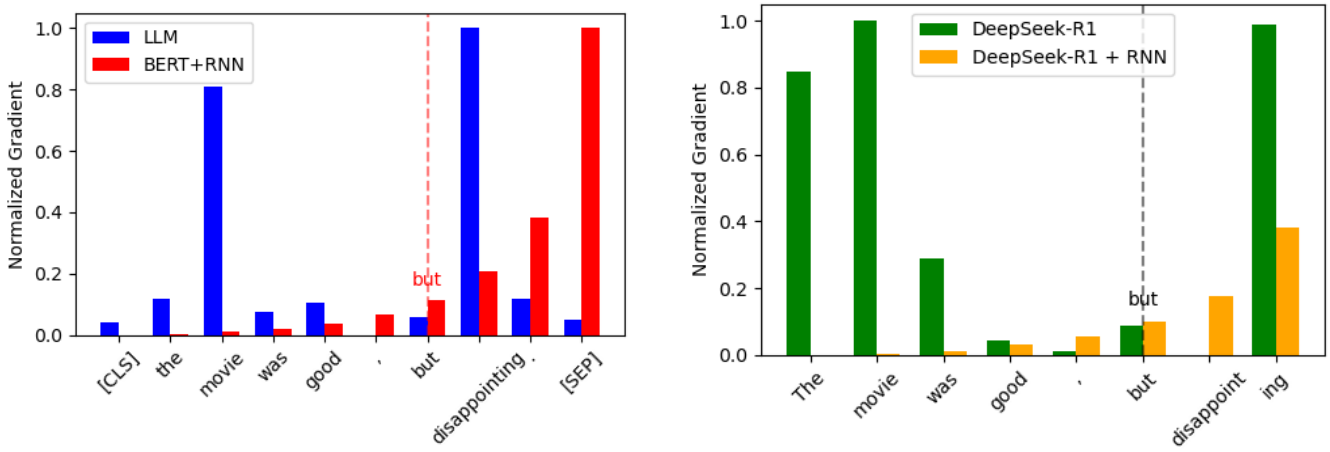
Pretrained large language models (LLMs) have demonstrated remarkable success across various language modeling tasks. However, in domain-specific applications, particularly those involving highly order-sensitive data, general LLMs exhibit limitations in achieving state-of-the-art performance. One notable issue is that the contextual embeddings by LLMs still lack a strong positional inductive bias, especially for long and highly ordered sequences, leading to the *lost in the middle* problem. In this work, we utilized the potential of sequential models (RNNs) with LLMs to address the issue and investigate whether RNN integration improves LLM performance. The LLM generates rich contextual embeddings using the attention mechanism of the Transformer. The RNN further processes the LLM embeddings to capture the contextual semantics of long and order-sensitive dependencies. The LLM-RNN model leverages the potential of both Transformer and recurrent structures to enhance performance in domain-specific tasks. We perform a wide range of experiments leveraging multiple types of LLMs (encoder-only, encoder-decoder, and decoder-only) and RNNs (GRU, LSTM, BiGRU, and BiLSTM) across diverse public and real-world datasets to investigate the potential (either positive or negative) of LLM-RNN models. The experimental results highlight the superiority of the LLM-RNN model, showing improvements in commonsense reasoning, code understanding, and biomedical reasoning tasks.

Introduction

Large Language Models (LLMs) achieved groundbreaking performance in diverse NLP tasks, including commonsense analysis (Chang et al. 2024), code understanding (Du et al. 2024), code summarization and generation (Yan et al. 2024), biomedical text retrieval (Xu et al. 2024), question answering (Robinson and Wingate 2023), and text summarization, generation, and translation (Tu et al. 2024; Papi et al. 2024). For LLMs, training data is a pivotal factor leading to enhanced model performance (Wei et al. 2022), and at the same time, incorporating larger training data has substantially increased model size. However, these models are laying the foundation toward artificial general intelligence (Bubeck et al. 2023). Thus, LLMs have attained continuous attention from both academia (Zhao et al. 2023) and industry (Achiam et al. 2023).

Considering the wide range of use and success of LLMs in recent years, numerous methodologies and techniques have been developed to adapt these general-purpose models to domain-specific downstream tasks. Beyond the conventional model fine-tuning approach, where all parameters are adjusted during training (Howard and Ruder 2018), which is expensive, prompt-based adaptation methods have been introduced to modulate the behavior of frozen LLMs using carefully designed prompts (Li and Liang 2021; Tian et al. 2024; Lester, Al-Rfou, and Constant 2021). Low-rank adaptation techniques allow the pretrained model weights to remain fixed while introducing trainable rank-decomposition matrices, significantly reducing the number of trainable parameters (Hu et al. 2021). Rather than modifying the core parameters of LLMs, these approaches freeze the pretrained weights and typically introduce additional trainable parameters.

In recent years, LLMs have achieved great success in tackling real-world applications (e.g., translation, generation, and summarization), and the adaptability of LLMs to various downstream tasks has increased significantly. However, there is still room for performance improvement as they continue to show limitations in capturing and providing fundamental knowledge (Pan et al. 2024; Lewis et al. 2020). Lexical and semantic diversity, order sensitivity, the presence of long dependencies, unfamiliar symbols and words in text, and imbalanced datasets pose continuous challenges for LLMs, especially in sentiment analysis and code understanding (Chang et al. 2024; Rahman et al. 2025). In the study Poria et al. 2020, the authors explored significant challenges and highlighted some interesting research directions. Another notable issue is that contextual embeddings produced by LLMs lack a strong positional inductive bias, especially in long, highly ordered sequences, leading to phenomena such as the $lost\ in\ the\ middle$ problem (Wu et al. 2025). These limitations point to the need for models that can integrate richer contextual and sequential cues, particularly for highly order-sensitive data in domain-specific applications. As illustrated in Figure 1, a token-level comparison of saliency values for the sentence *"The movie was good, but disappointing"* is analyzed using BERT and DeepSeek



(a) Saliency of tokens with BERT and BERT+RNN

(b) Saliency of tokens with DS and DS+RNN

Figure 1: A token-level comparison of embedding shifts and saliency values for the sentence *"The movie was good, but disappointing."* BERT and DeepSeek serve as the base LLMs, each coupled with a BiGRU RNN.

(DS) as the LLMs, and BiGRU as the recurrent neural networks (RNN). The integration of RNN with BERT and DS significantly enhances context and order sensitivity, as evidenced by the model's ability to down-weight the token *good* and emphasize the contrastive cue *but*, thereby correctly prioritizing *disappointing* as the dominant token. The observed saliency adjustments indicate that the LLM-RNN captures nuanced, order-sensitive dependencies that standard transformer-based models may underrepresent. Motivated by these observations, we pose the following research question:

Coupling RNN with LLM: Does Their Integration Improve Highly Order-Sensitive Language Understanding?

To address the research question, we comprehensively examine the integration of RNNs with domain-specific pretrained LLMs, encompassing encoder-decoder models (CodeT5 (Wang et al. 2021), CodeT5$^+$ (Wang et al. 2023)), encoder-only models (RoBERTa (Liu et al. 2019), BioLinkBERT (Yasunaga, Leskovec, and Liang 2022), CodeBERT (Feng et al. 2020)), and decoder-only models (GPT2 (Radford et al. 2019), DeepSeek-Coder (Guo et al. 2024), DeepSeek-R1 (Guo et al. 2025)), in order to evaluate LLM-RNNs' performance on *commonsense reasoning*, *biomedical reasoning*, and *code understanding* tasks. We leverage RNN variants (e.g., GRU, LSTM, BiGRU, and BiLSTM) and also perform comprehensive hyperparameter tuning for model optimization. The proposed LLM-RNN model combines the strengths of both Transformer and Recurrent architectures. The LLM serves as the primary encoder, tokenizing and transforming input sequences into meaningful embeddings. The LLM-generated embeddings are passed through a dropout layer to mitigate overfitting before being processed with the RNN. The RNN captures sequential dependencies in the text, enhancing the model's ability to understand structural, order, and logical relationships. Finally, an FC layer maps the RNN outputs to target class labels, enabling the classification layer to perform the downstream predictive tasks.

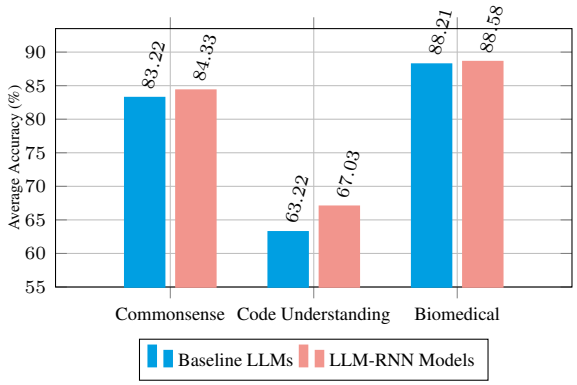


Figure 2: Comparison of average accuracy between LLM-RNN models and stand-alone counterparts across three tasks, evaluated on multiple benchmark datasets.

We conducted extensive experiments on multiple public datasets involving order sensitivity, with a particular focus on coding datasets across three diverse tasks. To achieve optimal performance, we fine-tuned the model hyperparameters. Our findings demonstrate that coupling RNN with LLM enables the model to better capture context, leading to significant improvements in performance. Figure 2 presents the averaged accuracy comparison between the LLM-RNN and stand-alone LLM across the three tasks using multiple benchmark datasets. Notably, LLM-RNN achieves accuracy (Avg.) improvement of approximately **+1.11%**, **+3.81%**, and **+0.37%** for commonsense reasoning, code understanding, and biomedical reasoning, respectively, compared to stand-alone LLM models. The obtained results highlight the effectiveness of the proposed approach, particularly in handling highly order-sensitive data such as source code. In summary, the key contributions are:

- To the best of our knowledge, this work presents the first comprehensive evaluation of coupling RNN with LLM, including encoder-only, encoder-decoder, and decoder-only architectures, across multiple publicly available datasets and diverse downstream tasks. We systematically investigate the performance of RNN-coupled LLMs within each architectural category across datasets.
- We integrate RNN architectures with LLMs and fine-tune the hyperparameters to harness the complementary strengths of both Transformer-based models and the sequential learning capabilities of RNNs. Since LLMs generate rich, contextually relevant token embeddings, RNNs further refine the contextual representations by capturing the structural, temporal, and order-sensitive dependencies inherent in the input. Furthermore, we adjust the layers (e.g., dropout, activation, and FC) of the LLM-RNN framework for optimal results.

Methodology

In this section, we describe the coupling of RNN with LLM to create LLM-RNN model. LLM-RNN model combines the strengths of the Transformer and RNN architectures to improve efficiency and accuracy in downstream tasks. Figure 3 shows the framework of the LLM-RNN model.

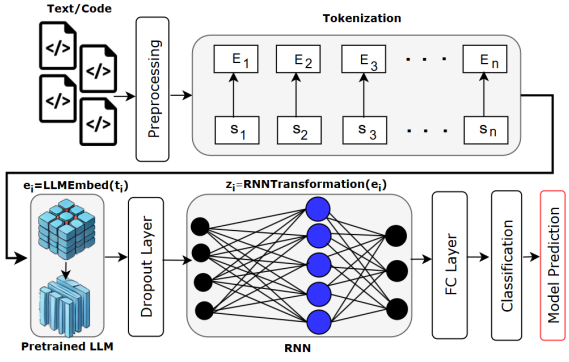


Figure 3: Architectural framework of the RNN-coupled LLM

Contextual Embedding with LLM

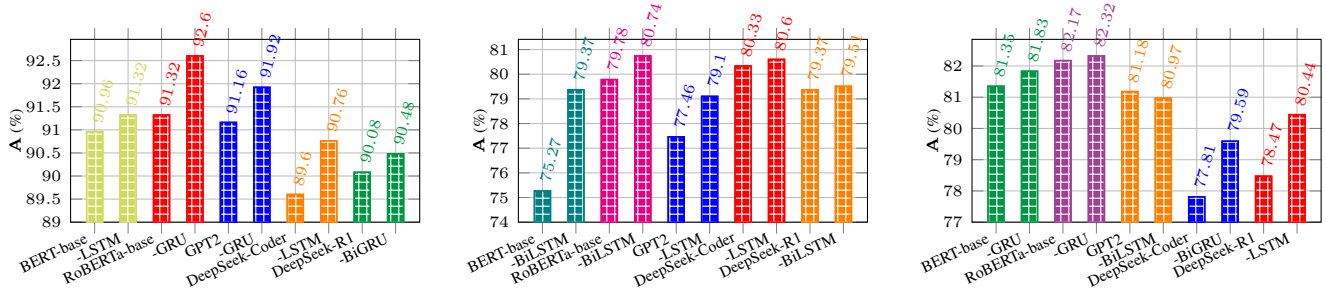
The tokenization strategies, such as Byte Pair Encoding (BPE), WordPiece, byte-level BPE, and SentencePiece in LLM, are not fixed but vary from model to model; tokenization is primarily used to represent input with the aim of reducing out-of-vocabulary problems (Sathe, Aggarwal, and Sitaram 2025; Sennrich, Haddow, and Birch 2016). Let $S = \{s_1, s_2, \ldots, s_n\}$ be the input (e.g., text or code), and the tokenization of the input S can be written as $G = \text{Tokenizer}(S) = \{g_1, g_2, \ldots, g_n\}$, where G is the sequence of tokens. Now, each token g_i is mapped to three (03) key categories, namely input ID ($\text{id}_i \in \mathbb{Z}$), token ID ($\text{gg}_i \in \{0, 1\}$), and attention mask ($\text{m}_i \in \{0, 1\}$), which will be fed into the pre-trained LLM encoder. Since we include encoder-only, decoder-only, and encoder-decoder LLMs as part of our model and investigation, they adopted different objectives; for example, encoder-only and encoder-decoder models typically use span-masking (denoising) losses, and decoder-only models use causal language modeling (next-token prediction). For our tasks (e.g., classification), all types of LLMs act as relevant encoders, and their final hidden representations are mapped to task-specific labels. Using a Transformer architecture, token embeddings $E(G') = \{e_1, e_2, \ldots, e_n\}$ are derived, where $e_i = \text{LLMEmbed}(g_i)$. These embeddings are processed by the Transformer encoder to produce contextual representations $H(G') = \text{TransformerEncoder}(E(G')) = \{h_1, h_2, \ldots, h_n\}$, with h_i being the contextual embedding for g_i. In our approach, the contextual embeddings are reprocessed through the RNN, which captures sequential dependencies for downstream tasks such as reasoning and code defect detection.

Contextual Embedding Reprocessing with RNN Sequential Model

The LLM-RNN highlights the effectiveness of RNN models in capturing rich contextual details, establishing them as a popular choice for sequential data analysis tasks due to their enhanced performance and resilience. The output embeddings from the final layer L of the LLM model are represented as a sequence $H^{(L)} = \{h_1^{(L)}, h_2^{(L)}, \ldots, h_n^{(L)}\}$, where $h_i^{(L)} \in \mathbb{R}^d$ denotes the i-th embedding in the sequence, and d is the dimensionality of the embeddings. To reduce overfitting, a dropout operation is applied to these embeddings, resulting in $h_i^{\text{drop}} = \text{Dropout}(h_i^{(L)})$, where $h_i^{\text{drop}} \in \mathbb{R}^d$. To align the dimensionality of the LLM output embeddings with the input requirements of the RNN, a linear transformation is applied to each embedding: $z_i = W_{\text{linear}} h_i^{\text{drop}} + b_{\text{linear}}$, where $z_i \in \mathbb{R}^{d_{\text{RNN}}}$, $W_{\text{linear}} \in \mathbb{R}^{d_{\text{RNN}} \times d}$ is the weight matrix, and $b_{\text{linear}} \in \mathbb{R}^{d_{\text{RNN}}}$ is the bias vector. The sequence of transformed embeddings $\{z_1, z_2, \ldots, z_n\}$ is then processed by the RNN, which computes the hidden states sequentially: $h_i^{\text{RNN}} = \text{RNN}(z_i, h_{i-1}^{\text{RNN}})$, where $h_i^{\text{RNN}} \in \mathbb{R}^{d_{\text{RNN}}}$ is the i-th hidden state, and h_{i-1}^{RNN} is the hidden state from the previous time step. The final output sequence of the RNN is given by $H_{\text{RNN}} = \{h_1^{\text{RNN}}, h_2^{\text{RNN}}, \ldots, h_n^{\text{RNN}}\}$, which combines the contextual embeddings from LLM with the sequential dependencies modeled by the RNN to enhance the semantic understanding and order-sensitivity. The sequential recurrence of the LLM-RNN approach allows the model to explicitly learn and track token-order dependencies. The proposed LLM-RNN framework can be written as $G \xrightarrow{\text{Tokenizer}} G' \xrightarrow{\text{Embedding}} E(G') \xrightarrow{\text{Transformer Encoder}} H(G') \xrightarrow{\text{RNN + Classifier}} \hat{y}$.

FC and Classification Layers

A dropout layer is applied to the RNN output H_{RNN} to mitigate overfitting, resulting in $H' = \text{Dropout}(H_{\text{RNN}})$, followed by an FC layer that maps the RNN hidden states to class logits: $\text{logits}_i = W_o H' + b_o$. Unlike traditional approaches that apply a softmax activation to produce class probabilities, we used raw logits directly.



(a) IMDb (b) Twitter US Airline (c) Sentiment140

Figure 4: The best **A** scores achieved by the BERT-base, RoBERTa-base, GPT2, DeepSeek-Coder, and DeepSeek-R1, along with their corresponding RNN-coupled models.

Parameter-Name	**Values**
Pre-trained LLMs	BERT, RoBERTa, CodeBERT, BioLinkBERT, CodeT5, CodeT5^{+}, GPT2, DeepSeek-Coder and -R1
RNNs	LSTM, BiLSTM, GRU, BiGRU
Optimizer (Δ)	AdamW, NAdam, RMSprop
Loss function ($\mathcal{L}$)	Categorical Cross Entropy (`cross_entropy`)
Epochs (*epoch*)	5
Dropout (d)	0.1, 0.2
Learning rates (l)	$1e^4$, $1e^5$, $2e^5$, $1e^6$
Hidden units (h) of RNNs	128, 256, 512

Table 1: The list of hyperparameters for the experiments

Experimental Setup

Hyperparameters

The performance of LLMs is highly dependent on selecting appropriate hyperparameters. In this work, we conducted extensive experiments with various hyperparameter configurations to evaluate model performance on multiple tasks. Table 1 details the hyperparameters used for fine-tuning during model training. For BiLSTM and BiGRU architectures, the number of RNN hidden units (h) is doubled ($2 \times h$) due to their bidirectional processing capabilities, which incorporate both forward ($\overrightarrow{h}$) and backward ($\overleftarrow{h}$) information. During training, categorical cross-entropy is employed to calculate the loss, defined as: $\mathbf{L}(g) = -\sum_{j=1}^{K} u_j \; log(\bar{u_j})$. Where g and K represent the model parameter and the number of classes, respectively, while u_j and $\bar{u_j}$ denote the true and predicted labels for the j^{th} sample.

Metrics

The performance of the models is evaluated using standard metrics (Rahman et al. 2025; Younas, Usman, and Yan 2022), including accuracy (A), precision (P), recall (R), and F1-score (F1). The accuracy metric (A) is defined as: $\boldsymbol{A} = \frac{1}{N}\sum_{l=1}^{|K|}\sum_{x:f(x)=l} B(f(x) = \hat{f}(x))$. Where, B is a function that returns 1 if the predicted class is correct and 0, otherwise. K represents the total number of classes, and $f(x) \in K = \{1, 2, 3, \cdots\}$. In addition to **A**, weighted-precision ($\boldsymbol{P}_\psi$), recall ($\boldsymbol{R}_\psi$), and F1-score ($\boldsymbol{F1}_\psi$) are computed to provide an unbiased and comprehensive performance evaluation.

Datasets

We evaluated the LLM-RNN approach using five public and three real-world datasets across multiple tasks. For commonsense reasoning, we employed the IMDb, Twitter US Airline, and Sentiment140 datasets. The IMDb dataset (Maas et al. 2011) comprises 50,000 reviews evenly split between positive and negative sentiments, providing a balanced dataset with 50% of samples in each class. The Twitter US Airline dataset (Tan et al. 2022) contains 14,640 tweets categorized into three sentiment classes: positive, neutral, and negative. The Sentiment140 dataset (Go, Bhayani, and Huang 2009) is a substantial collection of approximately 1.6 million tweets curated by Stanford University in 2009 for sentiment analysis. This dataset is equally balanced, with 50% of tweets representing positive sentiment and 50% representing negative sentiment. For code understanding, we used the defect detection, SearchAlg, SearchSortAlg, and SearchSortGT datasets. The defect detection benchmark dataset, sourced from CodeXGLUE (Zhou et al. 2019), is utilized to assess the model's ability to identify code defects. The other three datasets—SearchAlg, SearchSortAlg, and SearchSortGT—are collected from AOJ (Rahman, Shirafuji, and Watanobe 2024), a reputed repository of real-world source code. Finally, the NCBI dataset (O'Leary et al. 2024) is leveraged for the biomedical reasoning task.

Implementation Details

The experiments are conducted on a system running RHEL 8.8. The hardware configuration included an Intel Ice Lake (Xeon Platinum 8358) (2 sockets *32 cores/socket), 256 GB of RAM, and an NVIDIA A100 80GB PCIe graphics card.

Model (Base and RNNs)	IMDb (1)			Twitter Airline (2)			Sentiment140 (3)		
	$\mathbf{F1}_\psi$	$\mathbf{P}_\psi$	$\mathbf{R}_\psi$	$\mathbf{F1}_\psi$	$\mathbf{P}_\psi$	$\mathbf{R}_\psi$	$\mathbf{F1}_\psi$	$\mathbf{P}_\psi$	$\mathbf{R}_\psi$
BERT	90.96	90.96	90.96	75.88	76.62	75.27	81.31	81.56	81.35
+ LSTM (1), BiLSTM(2), GRU(3)	91.32 ↑	91.35	91.32	78.18 ↑	78.01	78.42	81.83 ↑	81.84	81.83
RoBERTa	91.31	91.44	91.32	80.12	80.70	79.78	82.17	82.21	82.17
+ BiLSTM(1), GRU(2,3)	92.96 ↑	92.96	92.96	80.93 ↑	81.47	80.60	82.32 ↑	82.32	82.32
GPT2	91.16	91.19	91.16	76.92	77.80	77.46	81.18	81.19	81.18
+ GRU(1), LSTM(2), BiLSTM(3)	91.92 ↑	91.94	91.92	78.86 ↑	78.74	79.10	80.97	80.97	80.97
DeepSeek-R1	90.09	90.21	90.08	79.36	79.42	79.37	78.47	78.47	78.47
+ BiGRU(1), BiLSTM(2), LSTM(3)	90.49 ↑	90.52	90.48	79.03	78.96	79.51	80.43 ↑	80.53	80.44
DeepSeek-Coder	89.59	89.92	89.60	79.89	79.80	80.33	77.81	77.81	77.81
+ LSTM(1,2), BiGRU(3)	90.76 ↑	90.77	90.76	80.14 ↑	80.18	80.60	79.57 ↑	79.68	79.59

Table 2: The best results for commonsense reasoning obtained using encoder-only and decoder-only LLM architectures, including their respective base models and RNN-coupled variants. **Note:** Notation such as BERT + LSTM (1) indicates that the model achieved the best performance on the IMDb dataset; this notation applies similarly to the other models and datasets.

Model		Learning Rate (l)	Optimizer (Δ)	Hidden Units (h)	A (%)	F1 (%)	
LLM	RNN					Weighted (ψ)	Macro (μ)
RoBERTa	-	-	-	-	61.05	-	-
	BiGRU	$1e^{-5}$	NAdam	512	66.40 (↑)	64.76	64.0
CodeBERT	-	-	-	-	62.08	-	-
	GRU	$2e^{-5}$	AdamW	512	66.03 (↑)	65.32	65.0
DeepSeek-Coder	-	$1e^{-5}$	AdamW	256	65.37	64.08	63.39
	BiLSTM	$1e^{-5}$	AdamW	256	59.11	58.86	58.42
CodeT5	-	-	-	-	64.86	64.74	-
	GRU	$1e^{-4}$	AdamW	512	67.90 (↑)	67.18	67.0
CodeT5$^+$	-	-	-	-	64.90	64.74	-
	BiGRU	$2e^{-5}$	RMSProp	256	67.79 (↑)	66.82	66.0

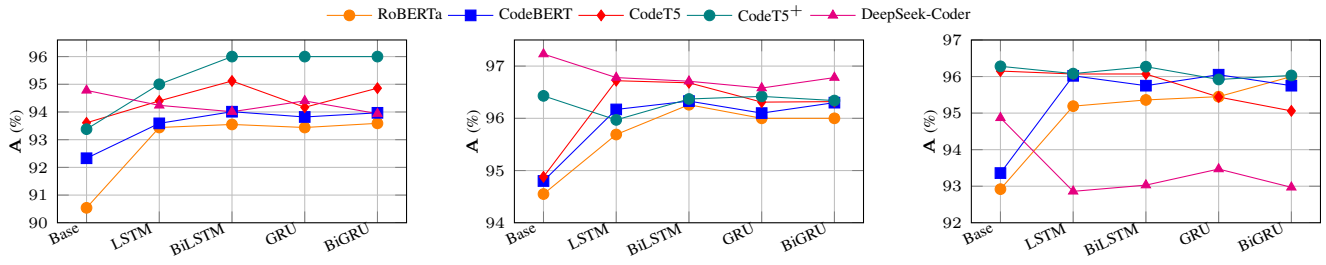
Table 3: Comparison of the best $\mathbf{A}$ and F1 scores achieved by the top-performing encoder-only, decoder-only, and encoder-decoder LLMs integrated with RNNs, along with their respective base models, on the defect detection dataset.

Results and Analysis

We conducted extensive experiments using various LLM-RNN models. Figure 4 presents the $\mathbf{A}$ scores of the best-performing BERT-, GPT2-, DeepSeek-Coder-, DeepSeek-R1-, and RoBERTa-RNN[1] models, alongside their corresponding base counterparts, for the commonsense reasoning task on IMBd, Twitter, and Sentiment140 datasets. Figure 4a demonstrates that the BERT-base model achieved an $\mathbf{A}$ score of approximately 90.96%, while the BERT-LSTM model attained 91.32%, reflecting a 0.36% improvement. The RoBERTa-GRU model attained an $\mathbf{A}$ score of 92.60%, marking a 1.28% improvement compared to the RoBERTa-base model. Among the decoder-only models, GPT2-GRU, DeepSeek-R1-BiGRU, and DeepSeek-Coder-LSTM exhibited improvements of 0.76%, 1.16%, and 0.40%, respectively, compared to their corresponding base models. Similar trends are observed in the Twitter and Sentiment140 datasets, as depicted in Figures 4b and 4c, respectively. In most cases, integrating RNNs with both encoder-only and decoder-only models enhanced the performance of LLMs. Figure 4 highlights that the RoBERTa-RNN models achieved greater improvements than the other models across all datasets. Additionally, Table 2 presents the best weighted scores ($\mathbf{F1}_\psi$, $\mathbf{P}_\psi$, and $\mathbf{R}_\psi$). The results obtained clearly demonstrate that integrating RNNs improves the performance (as marked by ↑) of both encoder-only and decoder-only LLMs on commonsense reasoning tasks.

For the code understanding task, we employed encoder-only, decoder-only, and encoder-decoder LLMs and conducted extensive experiments under various hyperparameter settings. Table 3 presents a comparative analysis of the best $\mathbf{A}$ and F1 scores achieved by the top-performing models alongside their corresponding base models on the defect detection benchmark dataset. The $\mathbf{A}$ scores for the base models—RoBERTa, CodeBERT, DeepSeek-Coder, CodeT5, and CodeT5$^+$—are 61.05%, 62.08%, 65.37%, 64.86%, and 64.90%, respectively. In comparison, their RNN-augmented counterparts (except for DeepSeek-Coder) demonstrated notable performance gains. Specifically, the RoBERTa-BiGRU model achieved an $\mathbf{A}$ score of 66.40%, indicating a 5.35% improvement over the base RoBERTa. Similarly, the CodeBERT-GRU model reached 66.03%, showing a 3.95% improvement. The CodeT5-GRU and CodeT5$^+$-BiGRU models attained $\mathbf{A}$ scores of 67.90% and 67.79%,

[1] RoBERTa-RNN encompasses four models, each integrating a different RNN variant: LSTM, GRU, BiLSTM, and BiGRU. This similarly applies to the BERT-, CodeBERT-, GPT2-, DeepSeek-R1-, DeepSeek-Coder-, BioLinkBERT-, CodeT5-, and CodeT5$^+$-RNN models.



(a) SearchAlg (b) SearchSortAlg (c) SearchSortGTAlg

Figure 5: Comparison of $\boldsymbol{A}$ scores for the top-performing RoBERTa-RNN, CodeBERT-RNN, CodeT5-RNN, CodeT5+-RNN, and DeepSeek-Coder LLMs with their respective base models on the SearchAlg, SearchSortAlg, and SearchSortGTAlg datasets.

corresponding to gains of 3.04% and 2.89%, respectively, over their base models. Among all models, CodeT5-GRU recorded the highest $\mathbf{A}$ score. These results highlight the effectiveness of integrating RNN architectures with LLMs, demonstrating significant enhancements in performance for code understanding tasks.

To further evaluate the impact of integrating RNN with LLM, we conducted experiments on three real-world coding datasets. Figure 5 presents a comparative analysis of the $\mathbf{A}$ scores across RoBERTa, CodeBERT, CodeT5, CodeT5$^+$, and DeepSeek-Coder, both in their stand-alone and RNN-coupled forms. Among the models, CodeT5$^+$-BiLSTM achieved the highest $\mathbf{A}$ scores of approximately 96.00% and 96.27% on the SearchAlg and SearchSortGTAlg datasets, respectively. Additionally, DeepSeek-Coder-BiGRU attained an $\mathbf{A}$ score of 96.78% on the SearchSortAlg dataset. These results indicate that coupling RNN with LLM generally leads to significant performance improvements. However, this trend does not consistently hold for DeepSeek-Coder, where RNN integration offered limited or no gain. We also computed weighted evaluation metrics—$\mathbf{F1}_\psi$, $\mathbf{P}_\psi$, and $\mathbf{R}_\psi$—to provide a more nuanced assessment. The results, summarized in Table 4, reveal consistent performance improvement patterns across all datasets, further validating the benefit of RNN integration.

Figure 6 demonstrates that incorporating RNN with LLM enhances performance on biomedical reasoning tasks. The RoBERTa-LSTM model achieved an absolute improvement of 0.60% in $\mathbf{A}$ score over the RoBERTa-base model, which attained 88.15%. BioLinkBERT-GRU yielded a modest gain of 0.14%, while GPT2-LSTM and DeepSeek-R1-LSTM achieved increases of 0.10% and 0.53%, respectively, compared to their corresponding base models. Table 5 presents detailed weighted evaluation metrics to offer a comprehensive assessment of model performance. All RNN-augmented models outperformed their base counterparts in terms of $\mathbf{F1}_\psi$ score. These findings highlight that coupling RNN with LLM can lead to consistent and significant gains in biomedical reasoning accuracy and robustness.

Hyperparameter Sensitivity

We conducted a sensitivity analysis focusing on key hyperparameters: learning rates, optimizers, and the number of

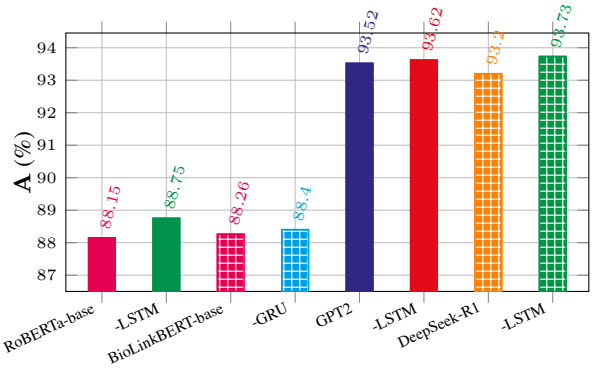


Figure 6: Comparison of the best $\mathbf{A}$ score achieved by the base models—RoBERTa, BioLinkBERT, GPT-2, and DeepSeek-R1—and their RNN-coupled variants on the NCBI dataset.

RNN hidden units. Figure 7a illustrates that the RoBERTa-BiLSTM model achieved optimal performance on the Twitter dataset with a learning rate of $l = 1e^{-5}$ and hidden units $h = 256$, outperforming other parameter configurations. For the Sentiment140 dataset, the RoBERTa-GRU model failed to achieve optimal results with a $l = 1e^{-4}$ and $h = 256$, as shown in Figure 7b. These findings suggest that a lower l significantly enhances the model's performance, and highlight the importance of selecting appropriate parameters for optimal results. Additionally, Figure 8 compares the $\mathbf{A}$ scores obtained using the two top-performing optimizers, Δ = AdamW, NAdam. The results indicate that the models consistently achieved superior performance across most configurations, with the sole exception occurring at a learning rate of $l = 1e^{-6}$ for both optimizers. The model performance with these optimizers is sensitive to the l values, and excessively lowering l decreases the performance gains.

Analysis of Training Parameters and Time

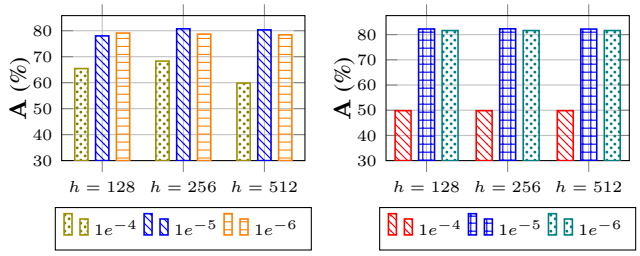
Figure 9 shows the training parameters and training times for the top-performing LLM-RNN models and their base counterparts. Notably, BiGRU coupled LLMs exhibit the highest number of training parameters. Among these, the

Model	SearchAlg (4)			SearchSortAlg (5)			SearchSortGTAlg (6)		
	$\mathbf{F1}_\psi$	$\mathbf{P}_\psi$	$\mathbf{R}_\psi$	$\mathbf{F1}_\psi$	$\mathbf{P}_\psi$	$\mathbf{R}_\psi$	$\mathbf{F1}_\psi$	$\mathbf{P}_\psi$	$\mathbf{R}_\psi$
RoBERTa	90.53	90.55	90.54	94.40	94.51	94.55	92.79	92.89	92.92
+ BiGRU (4, 6) BiLSTM (5)	93.63↑	93.90	93.59	96.25↑	96.25	96.26	96.00↑	96.10	96.00
CodeBERT	92.36	92.40	92.33	94.73	94.93	94.80	93.22	93.37	93.36
+ BiLSTM(4, 5), LSTM(6)	94.04↑	94.11	94.01	96.34↑	96.37	96.33	96.04↑	96.21	96.02
CodeT5	93.63	93.67	93.61	94.88	95.14	94.88	96.01	96.03	96.15
+ BiLSTM(4, 6), LSTM(5)	95.12↑	95.15	95.12	96.72↑	96.76	96.72	96.01↑	96.06	96.07
CodeT5$^+$	93.38	93.39	93.38	96.42	96.44	96.43	96.26	96.32	96.28
+ BiGRU(4), GRU(5), BiLSTM(6)	94.42↑	94.43	94.42	96.42↑	96.44	96.42	96.26	96.31	96.27
DeepSeek-Coder	94.79	94.81	94.78	97.23	97.23	97.23	94.78	94.84	94.87
+ GRU(4), LSTM(5, 6)	94.42	94.50	94.40	96.77	96.78	96.78	92.97	92.90	92.86

Table 4: Performance comparison between the base LLMs and their best-performing RNN-coupled counterparts in terms of weighted $\mathbf{F1}\psi$, $\mathbf{P}\psi$, and $\mathbf{R}_\psi$ scores. The evaluation includes encoder-only, decoder-only, and encoder-decoder LLMs, each integrated with different RNN variants.

Model	$\mathbf{F1}_\psi$	$\mathbf{P}_\psi$	$\mathbf{R}_\psi$
RoBERTa	86.75	85.40	88.15
+ LSTM	87.02 (↑)	85.37	88.75
BioLinkBERT	86.81	85.43	88.26
+ GRU	86.86 (↑)	85.39	88.40
GPT2	93.52	93.52	93.52
+ LSTM	93.63 (↑)	93.64	93.62
DeepSeek-R1	93.20	93.20	93.20
+ LSTM	93.75 (↑)	93.98	93.73

Table 5: The best results for biomedical reasoning using RoBERTa, BioLinkBERT, GPT2, and DeepSeek-R1 models on the NCBI dataset.



(a) RoBERTa-BiLSTM on Twitter dataset

(b) RoBERTa-GRU on Sentiment140 dataset

Figure 7: Impact of hyperparameters on model performance.

RoBERTa and CodeBERT models each comprise approximately 125 million parameters, while the BioLinkBERT, CodeT5 and CodeT5$^+$ models contain around 112 million parameters, as depicted in Figure 9a. In addition, the trainable parameter counts for decoder-only models are also calculated: DeepSeek-Coder, DeepSeek, and GPT-2 contain approximately 1.28 billion, 1.55 billion, and 124.44 million parameters, respectively. Interestingly, despite having fewer parameters, the CodeT5 and CodeT5$^+$ models, when combined with RNN variants, required significantly more training time compared to the RoBERTa and CodeBERT models, as shown in Figure 9b. This discrepancy can be attributed to architectural features, variations in tokenization strategies,

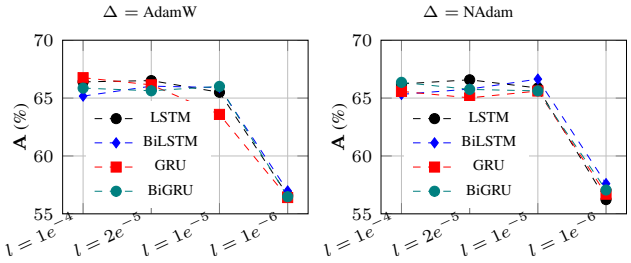
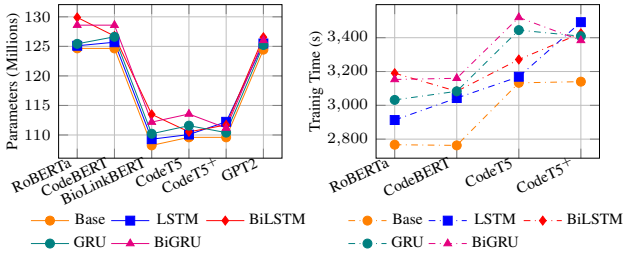


Figure 8: Accuracy ($\mathbf{A}$) scores of CodeT5-RNN models with several key parameters on defect detection dataset.

potentially less efficient computational optimization when coupling RNNs with CodeT5 and CodeT5$^+$, and specific hyperparameter settings, all of which may collectively contribute to the extended training time.



(a) Trainable parameters

(b) Training time

Figure 9: Overview of trainable parameters and training times for the RNN-augmented LLMs and their counterparts.

Ablation Study

We conducted an additional ablation study examining how LLM-RNN performance changes when the RNN is attached to early, intermediate, or final layer embeddings of the LLM. Table 6 shows the results for GPT2 on commonsense datasets. We observed that attaching the RNN to the final

Model	Dataset	$F1_\psi$ (early layer)	$F1_\psi$ (inter. layer)	$F1_\psi$ (final layer)
GPT2-LSTM	Twitter	76.31	78.34	78.86
GPT2-GRU	IMDB	85.08	87.56	91.92
GPT2-BiLSTM	Sentiment140	78.98	80.76	80.97

Table 6: Performance analysis of the GPT2–RNN model on commonsense reasoning datasets, where the RNN is attached to different GPT2 embedding layers.

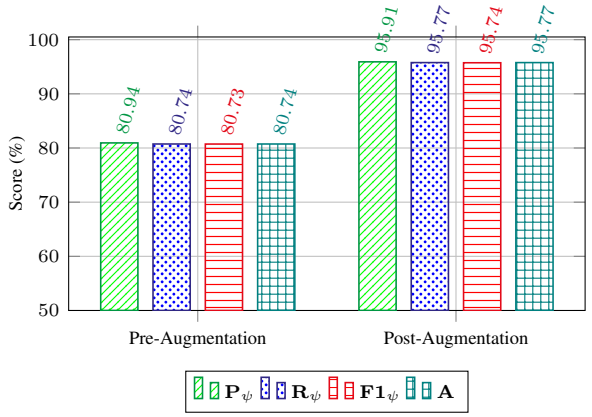


Figure 10: Performance comparison of the RoBERTa-BiLSTM model on the Twitter dataset pre- and post- data augmentation.

layer embeddings consistently yields the best performance. For example, GPT2 (final layer) + GRU achieves approximately 6.84% and 4.36% higher F1 scores compared to attaching the RNN to the early and intermediate layers, respectively.

Furthermore, the LLM-RNN models consistently achieved higher $\mathbf{F1}\psi$, $\mathbf{P}\psi$, and $\mathbf{R}_\psi$ scores compared to their base LLM counterparts. However, on the Twitter US Airline dataset, performance gains were limited compared to the IMDb and Sentiment140 datasets (Figure 4 and Table 2,). This is primarily attributed to severe class imbalance—62.69% of tweets are labeled as negative, 16.14% as positive, and 21.17% as neutral. To address class imbalance in the Twitter dataset, we performed data augmentation by synthetically generating samples for the neutral and positive classes to match the number of negative samples. The augmented dataset was then used to train the RoBERTa-BiLSTM model, as illustrated in Figure 10. Following augmentation, the model demonstrated substantial performance gains, achieving $\mathbf{F1}_\psi$ and $\mathbf{A}$ scores of 95.74% and 95.77%, respectively—an improvement of approximately 15% (↑) over its performance on the original imbalanced dataset (see in Table 2).

Discussion

The contextual embeddings generated by the LLM are further refined through an RNN layer, enabling the model to better capture deeper semantic structures and effectively handle highly order-sensitive data. Our experiments investigate how coupling RNNs with LLMs enhances semantic understanding and improves performance on tasks that require sensitivity to input order. The results show that the LLM-RNN models yield notable performance gains: an average accuracy improvement of approximately +1.11% for commonsense reasoning, +0.37% for biomedical reasoning, and a significant +3.81% for code understanding tasks. These findings underscore the utility of RNNs in augmenting LLMs, particularly for domains such as source code, where the sequential structure and order of the input are critical. We conducted an additional ablation study to examine how LLM-RNN performance varies when the RNN is attached to early, intermediate, or final-layer embeddings of the LLM. The results, presented in Table 6, show that attaching the RNN to the LLM's final-layer embeddings provides the strongest performance. Overall, the LLM-RNN integration proves especially effective in modeling highly order-sensitive data and contributes to addressing the well-known lost-in-the-middle problem associated with language models.

Conclusion

Our study encompasses three distinct application domains—commonsense reasoning, code understanding, and biomedical reasoning—spanning both benchmark and real-world datasets. We systematically explore encoder-only, decoder-only, and encoder-decoder language model architectures, each coupled with four RNN variants, thereby offering one of the most extensive evaluations to date of LLM-RNN hybrid models. To the best of our knowledge, no prior work has simultaneously considered this breadth of architectural configurations, LLM and RNN types, and task diversity. Our results consistently show that the sequential modeling capacity of RNNs complements the contextual representation of LLMs, particularly for order-sensitive inputs. Among all evaluated tasks, code understanding emerges as the most sensitive to sequential dependencies. In this context, LLM-RNN models achieved an average accuracy improvement of approximately 3.81% over their stand-alone LLM counterparts. It underscores the effectiveness of RNNs in refining LLM-generated embeddings by capturing syntactic and structural nuances inherent to programming languages. Similarly, for commonsense and biomedical reasoning tasks, the integration of RNNs led to measurable performance gains. Our findings affirmatively answer the research question posed in this study: *Yes, coupling RNN with LLM enhances model performance across a range of downstream tasks, particularly in scenarios requiring strong order-sensitive language understanding.*

References

Achiam, J.; Adler, S.; Agarwal, S.; Ahmad, L.; Akkaya, I.; Aleman, F. L.; Almeida, D.; Altenschmidt, J.; Altman, S.; Anadkat, S.; et al. 2023. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*.

Bubeck, S.; Chandrasekaran, V.; Eldan, R.; Gehrke, J.; Horvitz, E.; Kamar, E.; Lee, P.; Lee, Y. T.; Li, Y.; Lundberg, S.; et al. 2023. Sparks of artificial general intelligence: Early experiments with gpt-4. *arXiv preprint arXiv:2303.12712*.

Chang, M.; Yang, M.; Jiang, Q.; and Xu, R. 2024. Counterfactual-Enhanced Information Bottleneck for Aspect-Based Sentiment Analysis. *Proceedings of the AAAI Conference on Artificial Intelligence*, 38(16): 17736–17744.

Du, Y.; Ma, T.; Wu, L.; Zhang, X.; and Ji, S. 2024. AdaCCD: Adaptive Semantic Contrasts Discovery Based Cross Lingual Adaptation for Code Clone Detection. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, 17942–17950.

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

Go, A.; Bhayani, R.; and Huang, L. 2009. Twitter sentiment classification using distant supervision. *CS224N project report, Stanford*, 1(12): 2009.

Guo, D.; Yang, D.; Zhang, H.; Song, J.; Zhang, R.; Xu, R.; Zhu, Q.; Ma, S.; Wang, P.; Bi, X.; et al. 2025. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*.

Guo, D.; Zhu, Q.; Yang, D.; Xie, Z.; Dong, K.; Zhang, W.; Chen, G.; Bi, X.; Wu, Y.; Li, Y.; et al. 2024. DeepSeek-Coder: When the Large Language Model Meets Programming–The Rise of Code Intelligence. *arXiv preprint arXiv:2401.14196*.

Howard, J.; and Ruder, S. 2018. Universal Language Model Fine-tuning for Text Classification. In Gurevych, I.; and Miyao, Y., eds., *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 328–339. Melbourne, Australia: Association for Computational Linguistics.

Hu, E. J.; Shen, Y.; Wallis, P.; Allen-Zhu, Z.; Li, Y.; Wang, S.; Wang, L.; and Chen, W. 2021. Lora: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*.

Lester, B.; Al-Rfou, R.; and Constant, N. 2021. The Power of Scale for Parameter-Efficient Prompt Tuning. In Moens, M.-F.; Huang, X.; Specia, L.; and Yih, S. W.-t., eds., *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, 3045–3059. Online and Punta Cana, Dominican Republic: Association for Computational Linguistics.

Lewis, P.; Perez, E.; Piktus, A.; Petroni, F.; Karpukhin, V.; Goyal, N.; Küttler, H.; Lewis, M.; Yih, W.-t.; Rocktäschel, T.; et al. 2020. Retrieval-augmented generation for knowledge-intensive nlp tasks. *Advances in Neural Information Processing Systems*, 33: 9459–9474.

Li, X. L.; and Liang, P. 2021. Prefix-Tuning: Optimizing Continuous Prompts for Generation. In Zong, C.; Xia, F.; Li, W.; and Navigli, R., eds., *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, 4582–4597. Online: Association for Computational Linguistics.

Liu, Y.; Ott, M.; Goyal, N.; Du, J.; Joshi, M.; Chen, D.; Levy, O.; Lewis, M.; Zettlemoyer, L.; and Stoyanov, V. 2019. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*.

Maas, A.; Daly, R. E.; Pham, P. T.; Huang, D.; Ng, A. Y.; and Potts, C. 2011. Learning word vectors for sentiment analysis. In *Proceedings of the 49th annual meeting of the association for computational linguistics: Human language technologies*, 142–150.

O'Leary, N. A.; Cox, E.; Holmes, J. B.; Anderson, W. R.; Falk, R.; Hem, V.; Tsuchiya, M. T.; Schuler, G. D.; Zhang, X.; Torcivia, J.; et al. 2024. Exploring and retrieving sequence and metadata for species across the tree of life with NCBI Datasets. *Scientific data*, 11(1): 732.

Pan, S.; Luo, L.; Wang, Y.; Chen, C.; Wang, J.; and Wu, X. 2024. Unifying large language models and knowledge graphs: A roadmap. *IEEE Transactions on Knowledge and Data Engineering*.

Papi, S.; Gaido, M.; Negri, M.; and Bentivogli, L. 2024. StreamAtt: Direct Streaming Speech-to-Text Translation with Attention-based Audio History Selection. In Ku, L.-W.; Martins, A.; and Srikumar, V., eds., *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 3692–3707. Bangkok, Thailand: Association for Computational Linguistics.

Poria, S.; Hazarika, D.; Majumder, N.; and Mihalcea, R. 2020. Beneath the tip of the iceberg: Current challenges and new directions in sentiment analysis research. *IEEE transactions on affective computing*, 14(1): 108–132.

Radford, A.; Wu, J.; Child, R.; Luan, D.; Amodei, D.; Sutskever, I.; et al. 2019. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8): 9.

Rahman, M. M.; Shiplu, A. I.; Watanobe, Y.; and Alam, M. A. 2025. RoBERTa-BiLSTM: A Context-Aware Hybrid Model for Sentiment Analysis. *IEEE Transactions on Emerging Topics in Computational Intelligence*, 1–18.

Rahman, M. M.; Shirafuji, A.; and Watanobe, Y. 2024. Big Coding Data: Analysis, Insights, and Applications. *IEEE Access*, 12: 196010–196026.

Robinson, J.; and Wingate, D. 2023. Leveraging Large Language Models for Multiple Choice Question Answering. In *The Eleventh International Conference on Learning Representations*. Kigali, Rwanda.

Sathe, A.; Aggarwal, D.; and Sitaram, S. 2025. Improving Consistency in LLM Inference using Probabilistic Tokenization. In Chiruzzo, L.; Ritter, A.; and Wang, L., eds., *Findings of the Association for Computational Linguistics: NAACL 2025*, 4766–4778. Albuquerque, New Mexico: Association for Computational Linguistics.

Sennrich, R.; Haddow, B.; and Birch, A. 2016. Neural Machine Translation of Rare Words with Subword Units. In Erk, K.; and Smith, N. A., eds., *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 1715–1725. Berlin, Germany: Association for Computational Linguistics.

Tan, K. L.; Lee, C. P.; Anbananthen, K. S. M.; and Lim, K. M. 2022. RoBERTa-LSTM: a hybrid model for sentiment analysis with transformer and recurrent neural network. *IEEE Access*, 10: 21517–21525.

Tian, Y.; Song, H.; Wang, Z.; Wang, H.; Hu, Z.; Wang, F.; Chawla, N. V.; and Xu, P. 2024. Graph neural prompting with large language models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, 19080–19088.

Tu, L.; Yavuz, S.; Qu, J.; Xu, J.; Meng, R.; Xiong, C.; and Zhou, Y. 2024. Unlocking Anticipatory Text Generation: A Constrained Approach for Large Language Models Decoding. In Al-Onaizan, Y.; Bansal, M.; and Chen, Y.-N., eds., *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, 15532–15548. Miami, Florida, USA: Association for Computational Linguistics.

Wang, Y.; Le, H.; Gotmare, A. D.; Bui, N. D.; Li, J.; and Hoi, S. C. 2023. Codet5+: Open code large language models for code understanding and generation. *arXiv preprint arXiv:2305.07922*.

Wang, Y.; Wang, W.; Joty, S.; and Hoi, S. C. 2021. Codet5: Identifier-aware unified pre-trained encoder-decoder models for code understanding and generation. *arXiv preprint arXiv:2109.00859*.

Wei, J.; Tay, Y.; Bommasani, R.; Raffel, C.; Zoph, B.; Borgeaud, S.; Yogatama, D.; Bosma, M.; Zhou, D.; Metzler, D.; Chi, E. H.; Hashimoto, T.; Vinyals, O.; Liang, P.; Dean, J.; and Fedus, W. 2022. Emergent Abilities of Large Language Models. *Transactions on Machine Learning Research*.

Wu, X.; Wang, Y.; Jegelka, S.; and Jadbabaie, A. 2025. On the emergence of position bias in transformers. *arXiv preprint arXiv:2502.01951*.

Xu, R.; Shi, W.; Yu, Y.; Zhuang, Y.; Zhu, Y.; Wang, M. D.; Ho, J. C.; Zhang, C.; and Yang, C. 2024. BMRetriever: Tuning Large Language Models as Better Biomedical Text Retrievers. In Al-Onaizan, Y.; Bansal, M.; and Chen, Y.-N., eds., *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, 22234–22254. Miami, Florida, USA: Association for Computational Linguistics.

Yan, W.; Liu, H.; Wang, Y.; Li, Y.; Chen, Q.; Wang, W.; Lin, T.; Zhao, W.; Zhu, L.; Sundaram, H.; and Deng, S. 2024. CodeScope: An Execution-based Multilingual Multitask Multidimensional Benchmark for Evaluating LLMs on Code Understanding and Generation. In Ku, L.-W.; Martins, A.; and Srikumar, V., eds., *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 5511–5558. Bangkok, Thailand: Association for Computational Linguistics.

Yasunaga, M.; Leskovec, J.; and Liang, P. 2022. Linkbert: Pretraining language models with document links. *arXiv preprint arXiv:2203.15827*.

Younas, F.; Usman, M.; and Yan, W. Q. 2022. A deep ensemble learning method for colorectal polyp classification with optimized network parameters. *Applied Intelligence*, 1–24.

Zhao, W. X.; Zhou, K.; Li, J.; Tang, T.; Wang, X.; Hou, Y.; Min, Y.; Zhang, B.; Zhang, J.; Dong, Z.; et al. 2023. A survey of large language models. *arXiv preprint arXiv:2303.18223*.

Zhou, Y.; Liu, S.; Siow, J.; Du, X.; and Liu, Y. 2019. Devign: Effective vulnerability identification by learning comprehensive program semantics via graph neural networks. *Advances in neural information processing systems*, 32.