Coupling RNNs with LLMs: Does Their Integration Improve Language Modeling Performance?

Anonymous ACL submission

Abstract

Pretrained large language models (LLMs) have demonstrated remarkable success across various language modeling tasks. However, they continue to face inherent limitations in achieving state-of-the-art performance on many domain-specific applications. Previous research has explored diverse methodologies to enhance the performance of LLMs on downstream tasks. In this paper, we propose integrating recurrent neural networks (RNNs) with LLMs and investigate whether this integration improves language modeling performance. Particularly, LLMs are employed to generate rich and meaningful word embeddings, while RNNs excel at capturing the contextual semantics of long-range dependencies. The resulting LLM-RNN model leverages the complementary strengths of sequential and Transformer-based architectures to achieve enhanced performance. We conducted extensive experiments with rigorous hyperparameter tuning on multiple benchmark and real-world datasets. The experimental results highlight the superiority of the integrated LLM-RNN model in commonsense reasoning, code understanding, and biomedical reasoning tasks. Our codes are available at `https://github.com/mostafiz26/CouplingRNNsLLMs`.

1 Introduction

Large Language Models (LLMs) have demonstrated exceptional performance and general capability in various NLP tasks including biomedical text retrieval (Xu et al., 2024), question answering (Robinson and Wingate, 2023), sentiment analysis (Cai et al., 2024; Chang et al., 2024), code understanding (Du et al., 2024), code summarization and generation (Yan et al., 2024; Riddell et al., 2024), text summarization, generation, and translation (Tu et al., 2024; Papi et al., 2024; He et al., 2024). Moreover, incorporating larger training data has led to a substantial increase in model size, equipping LLMs with emergent capabilities (Wei et al., 2022b) and laying the foundation for advancements toward artificial general intelligence (Bubeck et al., 2023). Consequently, LLMs have garnered significant attention from both academia (Wei et al., 2022a; Zhao et al., 2023) and industry (Anil et al., 2023; Achiam et al., 2023).

Given the widespread success of LLMs, 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), 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; Brown et al., 2020; Lester et al., 2021). Additionally, 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 model and typically introduce additional trainable components. Moreover, various innovations, such as incorporating knowledge graph representations of text, feature hybridization, sequential model (i.e., RNNs) integration, and layer-specific adjustments, are being explored to enhance the structural and functional capabilities of LLMs (Md. Mostafizer et al., 2021; Bugueño and de Melo, 2023; Rahman et al., 2024a).

Despite the remarkable success of LLMs in addressing a variety of real-world applications and adapting to specific downstream tasks, they continue to exhibit inherent limitations in accurately capturing and providing grounded knowledge (Pan et al., 2024; Lewis et al., 2020). Challenges such as lexical diversity, the presence of long dependencies, unfamiliar symbols and words in text, and imbalanced datasets pose significant obstacles for LLMs, particularly in sentiment analysis (Chang et al., 2024; Rahman et al., 2024a). Poria et al., 2020 have highlighted existing challenges and proposed new research directions in this area. Additionally, while LLMs can generate complex code, such outputs often lack clarity and maintainability (Imai, 2022; Ziegler et al., 2022), creating challenges for programmers in debugging, maintenance, and extensibility (Liang et al., 2024; Vaithilingam et al., 2022). However, these limitations can be alleviated by incorporating context-aware, domain-specific information. Motivated by these observations, we pose the following research question:

Coupling RNNs with LLMs: Does Their Integration Improve Language Modeling Performance?

To address this question, our study investigates the integration of recurrent neural networks (RNNs) with domain-specific LLMs, including RoBERTa, BioLinkBERT, CodeBERT, CodeT5, and CodeT5^{+}, to evaluate their performance on commonsense reasoning, biomedical reasoning, and code understanding tasks. Specifically, we incorporate RNN variants such as LSTM, BiLSTM, GRU, and BiGRU, conducting extensive hyperparameter tuning to optimize model performance. The proposed hybrid model, LLM-RNN, combines a pretrained LLM, which includes both encoder and decoder components, with RNN architectures. The LLM serves as the primary encoder, tokenizing and transforming input sequences into meaningful embeddings. These embeddings are passed through a dropout layer to mitigate overfitting before being processed by the RNN. The RNN captures sequential dependencies in the text, enhancing the model's ability to understand structural and logical relationships. Finally, a dense layer maps the RNN outputs to target class labels, with a Softmax function applied to produce probability distributions for downstream tasks.

We conducted extensive experiments on multiple public datasets across three tasks: commonsense reasoning, code understanding, and biomedical reasoning. To achieve optimal performance, we fine-tuned the hyperparameters of our model. Our findings demonstrate that coupling RNNs with LLMs enables the model to better capture context, leading to significant improvements in performance. Figure 1 presents the averaged accuracy comparison between the LLM-RNN and stand-alone models across the three tasks using multiple benchmark datasets. Notably, LLM-RNN achieves accuracy improvements of approximately **+1.22%**, **+3.81%**, and **+0.37%** for commonsense reasoning, code understanding, and biomedical reasoning, respectively, compared to stand-alone LLM models. These results highlight the substantial benefits of our approach. In summary, the key contributions are:

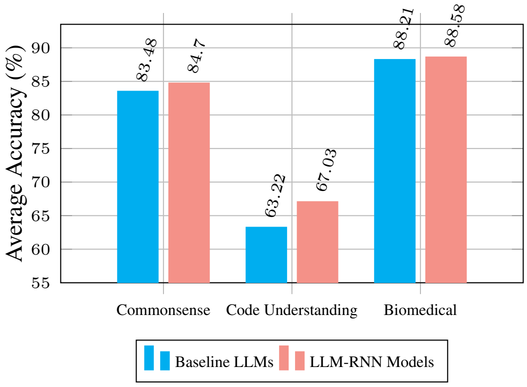


Figure 1: Comparison of accuracy between LLM-RNN models and stand-alone counterparts across three tasks, evaluated on multiple benchmark datasets.

- To the best of our knowledge, this work represents the first attempt to evaluate the performance of coupling RNNs with LLMs across multiple public datasets for diverse downstream tasks.
- We meticulously integrate RNN architectures with LLMs, combining the complementary strengths of transformer-based architectures and the sequential learning capabilities of RNNs. While LLMs generate rich, contextually relevant token embeddings, RNNs further process these embeddings to capture the structural and sequential dependencies inherent in text. This synergistic approach proves critical for effectively addressing complex tasks.
- Extensive experiments across various datasets and hyperparameter settings demonstrate the superiority of coupling RNNs with LLMs. This integration significantly enhances the model's ability to capture semantics, dependencies, and relations more accurately, underscoring its effectiveness in diverse tasks.

2 Methodology

In this section, we describe the coupling of RNNs with LLMs to create LLM-RNN model. This model combines the strengths of the Transformer and RNN architectures to improve efficiency and accuracy in downstream tasks. Figure 2 shows the framework of the LLM-RNN model.

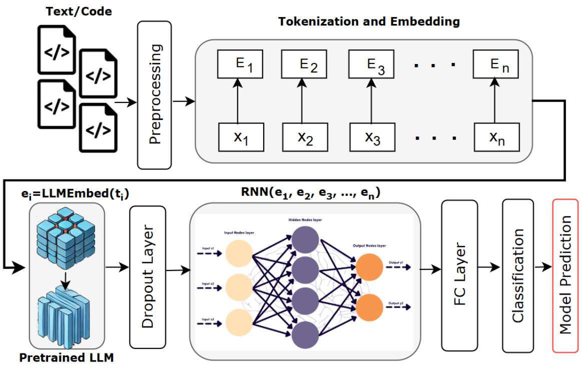


Figure 2: Framework of the proposed LLM-RNN model

2.1 Contextual Embedding with Large Language Model (LLM)

LLM uses *Byte Pair Encoding (BPE)* for tokenization, ensuring efficient representation of the input while minimizing the out-of-vocabulary (OOV) issue. Let $X = \{x_1, x_2, \ldots, x_n\}$ represent the raw input (e.g., code or text) sequence, and the tokenization process can be expressed as $T = \mathrm{BPE}(X) = \{t_1, t_2, \ldots, t_n\}$, where T is the sequence of tokens. Each token t_i is mapped to an **Input ID** $\mathrm{id}_i \in \mathbb{Z}^+$, a **Token type ID** $\mathrm{tt}_i \in \{0, 1\}$, and an **Attention mask** $\mathrm{am}_i \in \{0, 1\}$, enabling focused self-attention. LLM employs a *denoising objective* based on span masking, where the masked sequence $T' = \{t_1, \ldots, [\mathrm{MASK}], \ldots, t_n\}$ is reconstructed to minimize the loss $\mathcal{L}_{\mathrm{recon}} = -\sum_{i \in \mathrm{mask}} \log P(t_i \mid T')$, with $P(t_i \mid T')$ representing the probability of reconstructing the masked token. Using a Transformer architecture, token embeddings $E(T') = \{e_1, e_2, \ldots, e_n\}$ are derived, where $e_i = \mathrm{LLMEmbed}(t_i)$. These embeddings are processed by the Transformer encoder to produce contextual representations $H(T') = \mathrm{TransformerEncoder}(E(T')) = \{h_1, h_2, \ldots, h_n\}$, with h_i being the contextual embedding for t_i. The overall objective combines the span reconstruction loss with auxiliary tasks as $\mathcal{L} = \mathcal{L}_{\mathrm{recon}} + \mathcal{L}_{\mathrm{aux}}$, where $\mathcal{L}_{\mathrm{aux}}$ includes tasks such as code completion. The pipeline can be summarized as $T \xrightarrow{\mathrm{BPE}} T' \xrightarrow{\mathrm{Embedding}} E(T') \xrightarrow{\mathrm{Transformer\ Encoder}} H(T') \xrightarrow{\mathrm{Reconstruction}} T$, enabling robust contextual learning and efficient handling of code and text sequences.

2.2 Further Contextual Information Processing with RNN Sequential Modeling

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 $h_i^{\mathrm{drop}} = \mathrm{Dropout}(h_i^{(L)})$, where $h_i^{\mathrm{drop}} \in \mathbb{R}^d$. To align the dimensionality of the LLM output embeddings with the input requirements of the RNN, a linear transformation is applied to each embedding: $z_i = W_{\mathrm{linear}} h_i^{\mathrm{drop}} + b_{\mathrm{linear}}$, where $z_i \in \mathbb{R}^{d_{\mathrm{RNN}}}$, $W_{\mathrm{linear}} \in \mathbb{R}^{d_{\mathrm{RNN}} \times d}$ is the weight matrix, and $b_{\mathrm{linear}} \in \mathbb{R}^{d_{\mathrm{RNN}}}$ 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: $h_i^{\mathrm{RNN}} = \mathrm{RNN}(z_i, h_{i-1}^{\mathrm{RNN}})$, where $h_i^{\mathrm{RNN}} \in \mathbb{R}^{d_{\mathrm{RNN}}}$ 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 $H_{\mathrm{RNN}} = \{h_1^{\mathrm{RNN}}, h_2^{\mathrm{RNN}}, \ldots, h_n^{\mathrm{RNN}}\}$, which combines the contextual information from LLM with the sequential dependencies modeled by the RNN to enhance predictive performance.

2.3 FC and Classification Layers

A dropout layer is applied to H_{RNN}, $H' = \mathrm{Dropout}(H_{\mathrm{RNN}})$, to mitigate overfitting, followed by an FC layer that maps the RNN outputs to class logits:

$$Z_i = \mathrm{ReLU}(W_{\mathrm{dense}} H' + b_{\mathrm{dense}}) \tag{1}$$

Finally, a softmax function is applied to the dense layer output, producing a probability distribution over classes:

$$P(y_i|X) = \mathrm{Softmax}(W_o Z_i + b_o) \tag{2}$$

3 Experimental Setup

In this section, we outline the experimental setup, detailing the implementation environment, hyperparameter configurations, evaluation metrics, and datasets utilized in our experiments.

3.1 Implementation Details

The experiments are conducted on a system running Ubuntu 22.04.4 LTS (64-bit). The hardware configuration included an AMD Ryzen 9 3950X processor with 16 cores and 32 threads, 64 GB of RAM, and an NVIDIA GeForce RTX 3090 graphics card with 24 GB of dedicated memory. The system also featured a disk capacity of 500 GB, ensuring ample storage for experimental data and model training.

3.2 Hyperparameters

The performance of LLMs is highly dependent on selecting appropriate hyperparameters. In this study, we conducted extensive experiments with various hyperparameter configurations to evaluate model performance on commonsense reasoning, code understanding, and biomedical reasoning tasks. Table 1 details the hyperparameters used for fine-tuning during model training. Each RNN model is paired with an LLM to form a hybrid model. 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 follows:

$$\mathcal{L}(g) = -\sum_{j=1}^{K} u_j \; log(\bar{u_j}) \tag{3}$$

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.

3.3 Metrics

In this study, we undertake tasks such as commonsense reasoning, code defect detection, code classification, and biomedical reasoning as part of our analysis. The performance of the models is evaluated using standard metrics (Rahman et al., 2024a; Younas et al., 2022), including accuracy (A), precision (P), recall (R), and F1-score (F1). The accuracy metric (A) is defined as follows:

$$\boldsymbol{A} = \frac{1}{N} \sum_{l=1}^{|K|} \sum_{x: f(x)=l} H(f(x) = \hat{f}(x)) \tag{4}$$

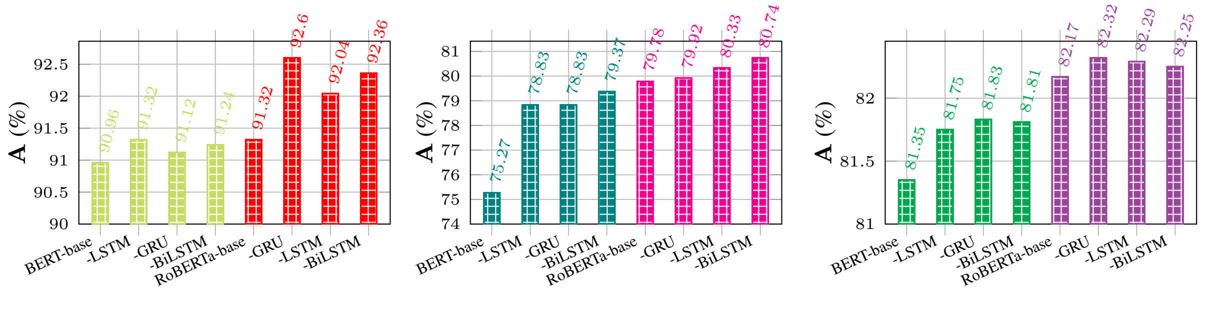
Parameter-Name	**Values**
Pre-trained LLMs	BERT, RoBERTa, CodeBERT, BioLinkBERT, CodeT5, CodeT5$^+$
RNNs	LSTM, BiLSTM, GRU, BiGRU
Optimizer (Δ)	AdamW, NAdam, RMSprop
Loss function ($\mathcal{L}$)	Categorical Cross Entropy (`cross_entropy`)
Epochs ($epoch$)	5
Dropout (d)	0.1, 0.2
Learning rates (l)	$1e^4$,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

Table 1: The list of hyperparameters for the experiments

Here, H 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 accuracy, 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 (Rahman et al., 2024a).

3.4 Datasets

In this study, we utilized five public datasets and three real-world datasets to evaluate our approach across the tasks of commonsense reasoning, code understanding, and biomedical reasoning. 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 et al., 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



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

Figure 3: Best accuracy ($\mathbf{A}$) scores achieved by BERT-base, BERT-LSTM, BERT-GRU, BERT-BiLSTM, RoBERTa-base, RoBERTa-GRU, RoBERTa-LSTM, and RoBERTa-BiLSTM models under various hyperparameter settings: $l = 1e^{-4}, 1e^{-5}, 1e^{-6}$ and $h = 128, 256, 512$. The models are trained for 5 epochs using the `AdamW` optimizer on the IMDb, Twitter US Airline, and Sentiment140 datasets.

Model	IMDb			Twitter US Airline			Sentiment140		
	$\mathbf{F1}_w$	$\mathbf{P}_w$	$\mathbf{R}_w$	$\mathbf{F1}_w$	$\mathbf{P}_w$	$\mathbf{R}_w$	$\mathbf{F1}_w$	$\mathbf{P}_w$	$\mathbf{R}_w$
BERT-Base	90.96	90.96	90.96	75.88	76.62	75.27	81.31	81.56	81.35
-GRU	91.12	91.14	91.12	77.57	77.45	77.73	81.83 ↑	81.84	81.83
-LSTM	91.32 ↑	91.35	91.32	77.72	77.54	78.01	81.75	81.75	81.75
-BiLSTM	91.24	91.25	91.24	78.18 ↑	78.01	78.42	81.81	81.81	81.81
RoBERTa-Base	91.31	91.44	91.32	80.12	80.70	79.78	82.17	82.21	82.17
-GRU	92.60	92.64	92.60	80.93 ↑	81.47	80.60	82.32 ↑	82.32	82.32
-LSTM	92.08	92.08	92.08	80.32	80.47	80.33	82.29	82.29	82.29
-BiLSTM	92.96 ↑	92.96	92.96	80.73	80.94	80.74	82.25	82.25	82.25

Table 2: Quantitative results for commonsense reasoning using BERT-RNN and RoBERTa-RNN models, along with their respective base models. All models are trained for 5 epochs on the IMDb, Twitter US Airline, and Sentiment140 datasets.

datasets—SearchAlg, SearchSortAlg, and SearchSortGT—are collected from AOJ (Rahman et al., 2024b), a respected repository of real-world source code. Finally, the NCBI dataset (O'Leary et al., 2024) is used for the biomedical reasoning task. It comprises approximately 7,298 samples, and the NER entities are converted into three class labels.

4 Results and Analysis

We conducted comprehensive experiments using various LLMs to evaluate three reasoning tasks under different hyperparameter configurations on both benchmark and real-world datasets. Figure 3 presents the accuracy scores of the best-performing BERT-RNN and RoBERTa-RNN[1] models, alongside their corresponding BERT-base and RoBERTa-base counterparts, for the commonsense reasoning task on IMBd, Twitter, and Sentiment140 datasets.

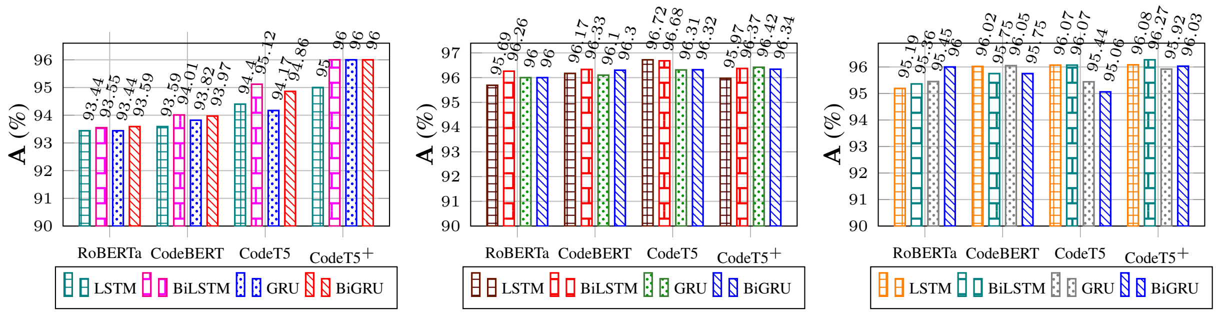
[1]RoBERTa-RNN encompasses four models, each integrating a different RNN variant: LSTM, GRU, BiLSTM, and BiGRU. This similarly applies to the BERT-, CodeBERT-, BioLinkBERT-, CodeT5-, and CodeT5$^+$-RNN models.

Figure 3a demonstrates that the BERT-base model achieved an accuracy of approximately 90.96%, while the BERT-LSTM model attained 91.32%, reflecting a 0.36% improvement. Similarly, the RoBERTa-GRU model achieved an accuracy of 92.60%, marking a 1.28% improvement compared to the RoBERTa-base model. Similar trends are observed in the Twitter and Sentiment140 datasets, as depicted in Figures 3b and 3c, respectively. In these cases, integrating RNNs with either BERT or RoBERTa consistently enhanced model performance. Moreover, Figure 3 highlights that RoBERTa-RNN models outperformed their BERT-RNN counterparts, achieving superior accuracy across three datasets. Additionally, Table 2 presents the weighted $\mathbf{F1}_w$ $\mathbf{P}_w$, and $\mathbf{R}_w$ scores of the BERT-RNN and RoBERTa-RNN models. The results clearly demonstrate that coupling RNNs enhances the performance of both BERT and RoBERTa models (indicated by the ↑) in commonsense reasoning tasks.

For the code understanding task, we employed

Model		Learning Rate (l)	Optimizer (Δ)	Hidden Units (h)	A (%)	F1 (%)	
LLM	**RNN**					**Weighted (ψ)**	**Macro (μ)**
RoBERTa	-	-	-	-	61.05	-	-
CodeBERT	-	-	-	-	62.08	-	-
CodeT5-Small	-	-	-	-	63.40	-	-
CodeT5-Base	-	-	-	-	64.86	64.74	-
CodeT5^{+}	-	-	-	-	64.90	64.74	-
RoBERTa	BiGRU	$1e^{-5}$	NAdam	512	66.40	64.76	64.0
CodeBERT	GRU	$2e^{-5}$	AdamW	512	66.03	65.32	65.0
CodeT5	GRU	$1e^{-4}$	AdamW	512	67.90	67.18	67.0
CodeT5^{+}	BiGRU	$2e^{-5}$	RMSProp	256	67.79	66.82	66.0

Table 3: Comparison of accuracy and F1 scores between top-performing models (RoBERTa-RNN, CodeBERT-RNN, CodeT5-RNN, and CodeT5^{+}-RNN) and state-of-the-art models on the defect detection dataset.



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

Figure 4: Comparison of accuracy ($\boldsymbol{A}$) scores for the top-performing RoBERTa-RNN, CodeBERT-RNN, CodeT5-RNN, and CodeT5+-RNN models on the SearchAlg, SearchSortAlg, and SearchSortGTAlg datasets.

LLM	**RNN**	SearchAlg			SearchSortAlg			SearchSortGTAlg		
		$\mathbf{F1}_{\psi}$	$\mathbf{P}_{\psi}$	$\mathbf{R}_{\psi}$	$\mathbf{F1}_{\psi}$	$\mathbf{P}_{\psi}$	$\mathbf{R}_{\psi}$	$\mathbf{F1}_{\psi}$	$\mathbf{P}_{\psi}$	$\mathbf{R}_{\psi}$
RoBERTa	LSTM	93.48	93.67	93.44	95.62	95.72	95.69	95.19	95.51	95.19
	Bi-LSTM	93.59	93.72	93.55	96.25	96.25	96.26	95.34	95.46	95.36
	GRU	93.47	93.59	93.44	95.96	96.01	96.00	95.42	95.54	95.45
	Bi-GRU	93.63	93.90	93.59	95.99	96.02	96.00	96.00	96.10	96.00
CodeBERT	LSTM	93.63	93.73	93.59	96.19	96.22	96.17	96.04	96.21	96.02
	Bi-LSTM	94.04	94.11	94.01	96.34	96.37	96.33	95.75	95.86	95.75
	GRU	93.85	93.91	93.82	96.09	96.11	96.10	96.04	96.13	96.05
	Bi-GRU	94.01	94.15	93.97	96.28	96.29	96.30	95.71	95.83	95.75
CodeT5	LSTM	94.40	94.41	94.40	96.72	96.76	96.72	95.98	96.11	96.07
	Bi-LSTM	95.12	95.15	95.12	96.68	96.68	96.68	96.01	96.06	96.07
	GRU	94.17	94.25	94.17	96.31	96.43	96.31	95.25	95.26	95.44
	Bi-GRU	94.86	94.86	94.86	96.28	96.29	96.32	94.98	95.15	95.06
CodeT5^{+}	LSTM	94.00	94.02	93.99	95.97	96.05	95.97	96.05	96.17	96.08
	Bi-LSTM	94.26	94.28	94.24	96.36	96.37	96.37	96.26	96.31	96.27
	GRU	94.27	94.30	94.26	96.42	96.44	96.42	95.92	96.06	95.92
	Bi-GRU	94.42	94.43	94.42	96.33	96.39	96.34	96.03	96.15	96.03

Table 4: Experimental results for the weighted $\mathbf{F1}\psi$, $\mathbf{P}\psi$, and $\mathbf{R}_{\psi}$ metrics on real-world datasets using LLM-RNN models

four pretrained LLMs and conducted extensive experiments under various hyperparameter configurations. Table 3 provides a comparative analysis of the accuracy and F1 scores for the top-performing models and their base models on the defect detection dataset. Based on the results, we identified four top-performing models and their corresponding hyperparameter settings from RoBERTa-RNN, CodeBERT-RNN, CodeT5-RNN, and CodeT5$^+$-RNN. Table 3 also includes the results of base models, namely RoBERTa, CodeBERT, CodeT5, and CodeT5+. The base models achieved notable accuracy scores of 61.05%, 62.08%, 64.86%, and 64.90%, respectively. In contrast, the RoBERTa-BiGRU model achieved an accuracy of 66.40%, representing an improvement of approximately 5.35% (↑) over its standalone RoBERTa counterpart. Similarly, the CodeBERT-GRU model attained an accuracy of 66.03%, marking a 3.95% (↑) improvement compared to its base model. The CodeT5-GRU and CodeT5$^+$-BiGRU models achieved accuracies of 67.90% and 67.79%, respectively, reflecting improvements of 3.04% (↑) and 2.89% (↑) over their standalone counterparts. These results highlight the effectiveness of integrating RNN architectures with LLMs, demonstrating significant enhancements in performance for code understanding tasks.

To further assess the effectiveness of the models, we conducted experiments with top-performing LLM-RNN models on three real-world datasets. Weighted scores for $\mathbf{F1}\psi$, $\mathbf{P}\psi$, and $\mathbf{R}\psi$ are computed, with most models achieving notable results, as presented in Table 4. Figure 4 provides a comparative analysis of $\mathbf{A}$ scores across the three datasets. On the SearchAlg dataset, the CodeT5$^+$-RNN model achieved an $\mathbf{A}$ score of 96.00%, outperforming all other models. A similar pattern was observed for the SearchSortAlg and SearchSortGTAlg datasets, where CodeT5- and CodeT5$^+$-RNN models consistently demonstrated superior performance.

Figure 5 illustrates that integrating RNNs with LLMs enhances performance in biomedical reasoning tasks. The RoBERTa-base model achieved an accuracy ($\mathbf{A}$) of 88.15%, which is lower than that of the other models. In comparison, the RoBERTa-LSTM model improved $\mathbf{A}$ by approximately 0.60% (↑) compared to the RoBERTa-base model. Similarly, the BioLinkBERT-GRU model achieved a modest improvement of 0.14% over its base model. Furthermore, Table 5 provides detailed F1 and other evaluation metrics to offer a comprehensive view of model performance. These results clearly indicate that coupling RNNs with LLMs significantly boosts model performance in biomedical reasoning tasks.

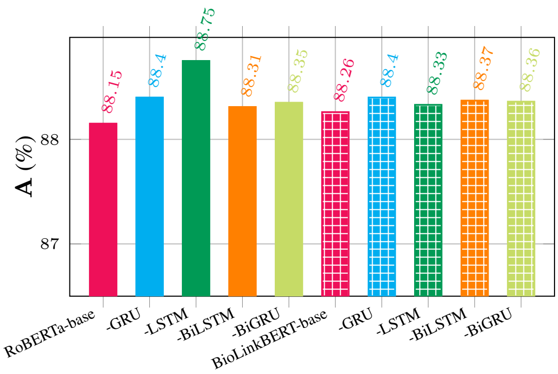


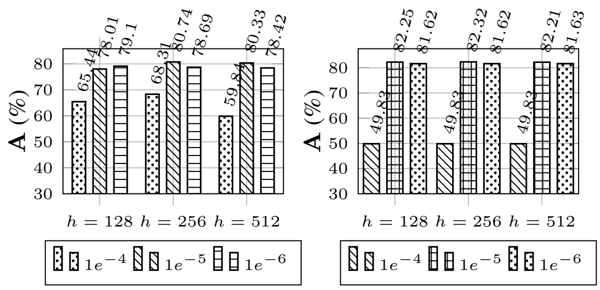
Figure 5: Comparision of the accuracies of RoBERTa-RNN and BioLinkBERT-RNN models on NCBI dataset.

Model	$\mathbf{F1}_w$	$\mathbf{P}_w$	$\mathbf{R}_w$
RoBERTa-Base	86.75	85.40	88.15
-GRU	86.85	85.36	88.40
-LSTM	87.02	85.37	88.75
-BiGRU	86.83	85.38	88.35
-BiLSTM	86.80	85.36	88.31
BioLinkBERT-Base	86.81	85.43	88.26
-GRU	86.86	85.39	88.40
-LSTM	86.84	85.42	88.33
-BiGRU	86.82	85.33	88.37
-BiLSTM	86.84	85.39	88.36

Table 5: Quantitative results for biomedical reasoning using RoBERTa and BioLinkBERT models on the NCBI dataset.

4.1 Hyperparameter Sensitivity

We conducted a sensitivity analysis focusing on key hyperparameters: learning rates, optimizers, and the number of RNN hidden units. The performance of the LLM-RNN models was evaluated by varying these parameters. Figure 6a 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 learning rate of $l = 1e^{-4}$, as shown in Figure 6b. This suggests that a lower learning rate significantly enhances the model's performance. Additionally, Figure 7 presents a comparative analysis of accuracy (**A**) scores achieved with two top-performing optimizers, $\Delta = \{\text{AdamW, NAdam}\}$. The results indicate that the models consistently delivered superior performance, except when the learning rate was set to $l = 1e^{-6}$.



(a) RoBERTa-BiLSTM on Twitter (b) RoBERTa-GRU on Sentiment140

Figure 6: Impact of hyperparameters on model performance.

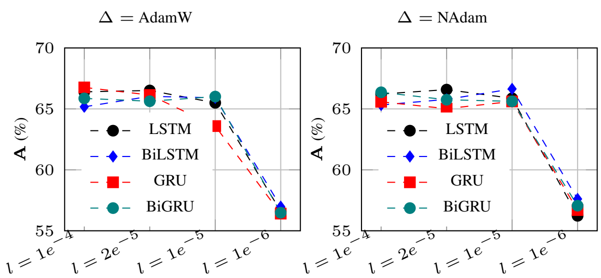
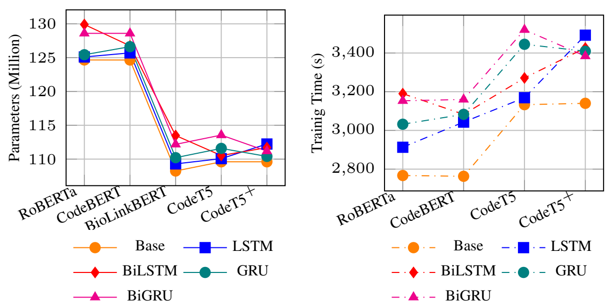


Figure 7: Accuracy (**A**) of CodeT5-RNN models using two top optimizers $\Delta = $ {AdamW, NAdam}, RNN hidden units $h = \{128, 256, 512\}$, and learning rates $l = \{1e^{-4}, 2e^{-5}, 1e^{-5}, 1e^{-6}\}$ with the defect detection benchmark dataset.

4.2 Analysis of Training Parameters and Time Efficiency

Figure 8 presents an analysis of training parameters and training times for the top-performing models and their base counterparts. Notably, LLMs integrated with BiGRU exhibit the highest number of training parameters. Among these, the 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 8a. 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 8b. This discrepancy can be attributed to architectural features, variations in tokenization strategies, 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) Training parameters (b) Training time

Figure 8: Comparison of training parameters and training times between the top-performing models and their respective base models.

5 Conclusion

In this paper, we explore the integration of recurrent neural networks (RNNs) with domain-specific large language models (LLMs) to evaluate their performance across commonsense reasoning, code understanding, and biomedical reasoning tasks. In the proposed LLM-RNN framework, the LLM tokenizes and transforms input sequences into meaningful embeddings. These embeddings are further processed by the RNN to capture sequential dependencies, thereby enhancing the LLM's capacity to understand structural and logical relationships. This approach enables pre-trained LLMs to acquire additional knowledge from input data more effectively. We pose the research question: "*Coupling RNNs with LLMs: Does Their Integration Improve Language Modeling Performance?*". To address this, we conduct extensive experiments across the three aforementioned tasks. Our results demonstrate that coupling RNNs with LLMs improves accuracy by approximately **+1.22%**, **+3.81%**, and **+0.37%** for commonsense reasoning, code understanding, and biomedical reasoning tasks, respectively, compared to stand-alone LLM models. In addition, we perform hyperparameter sensitivity analysis and examine trainable parameters alongside computational time to validate the effectiveness and feasibility of the LLM-RNN integration.

6 Limitations

The empirical investigation of this study demonstrates that coupling RNNs with LLMs significantly enhances performance compared to stand-alone LLMs across various tasks on multiple benchmark datasets. However, the analysis is limited to four specific RNN variants and several pretrained LLMs, selected based on the tasks. The effectiveness of this coupling may vary due to several critical factors. These include (i) differences in data preprocessing strategies, such as tokenization, normalization, and feature extraction, which can influence overall performance, (ii) the selection and tuning of hyperparameters, such as learning rate, batch size, and optimization strategies, play a pivotal role in the observed outcomes, (iii) variability in datasets, as the same tasks may be evaluated using different datasets, can also lead to discrepancies in results, (iv) the specific choice of RNN models (e.g., LSTM, GRU) and LLMs (e.g., BERT, RoBERTa, CodeT5, BioLinkBERT), as well as their internal architectures, significantly affects performance, and (v) differences in how RNNs and LLMs are integrated, including layer connections and attention mechanisms, can influence the synergy between the models. To gain a more comprehensive understanding of this hybrid approach, future studies could explore a broader range of RNN and LLM variants, investigate alternative coupling strategies, and systematically assess the impact of these influencing factors.

References

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. 2023. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*.

Rohan Anil, Andrew M Dai, Orhan Firat, Melvin Johnson, Dmitry Lepikhin, Alexandre Passos, Siamak Shakeri, Emanuel Taropa, Paige Bailey, Zhifeng Chen, et al. 2023. Palm 2 technical report. *arXiv preprint arXiv:2305.10403*.

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

Sébastien Bubeck, Varun Chandrasekaran, Ronen Eldan, Johannes Gehrke, Eric Horvitz, Ece Kamar, Peter Lee, Yin Tat Lee, Yuanzhi Li, Scott Lundberg, et al. 2023. Sparks of artificial general intelligence: Early experiments with gpt-4. *arXiv preprint arXiv:2303.12712*.

Margarita Bugueño and Gerard de Melo. 2023. Connecting the dots: What graph-based text representations work best for text classification using graph neural networks? In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 8943–8960, Singapore. Association for Computational Linguistics.

Hongjie Cai, Heqing Ma, Jianfei Yu, and Rui Xia. 2024. A joint coreference-aware approach to document-level target sentiment analysis. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 12149–12160, Bangkok, Thailand. Association for Computational Linguistics.

Mingshan Chang, Min Yang, Qingshan Jiang, and Ruifeng Xu. 2024. Counterfactual-enhanced information bottleneck for aspect-based sentiment analysis. *Proceedings of the AAAI Conference on Artificial Intelligence*, 38(16):17736–17744.

Yangkai Du, Tengfei Ma, Lingfei Wu, Xuhong Zhang, and Shouling Ji. 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, pages 17942–17950.

Alec Go, Richa Bhayani, and Lei Huang. 2009. Twitter sentiment classification using distant supervision. *CS224N project report, Stanford*, 1(12):2009.

Jianfeng He, Runing Yang, Linlin Yu, Changbin Li, Ruoxi Jia, Feng Chen, Ming Jin, and Chang-Tien Lu. 2024. Can we trust the performance evaluation of uncertainty estimation methods in text summarization? In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 16514–16575, Miami, Florida, USA. Association for Computational Linguistics.

Jeremy Howard and Sebastian Ruder. 2018. Universal language model fine-tuning for text classification. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 328–339, Melbourne, Australia. Association for Computational Linguistics.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2021. Lora: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*.

Saki Imai. 2022. Is github copilot a substitute for human pair-programming? an empirical study. In *Proceedings of the ACM/IEEE 44th International Conference on Software Engineering: Companion Proceedings*, pages 319–321.

Brian Lester, Rami Al-Rfou, and Noah Constant. 2021. The power of scale for parameter-efficient prompt tuning. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 3045–3059, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, et al. 2020. Retrieval-augmented generation for knowledge-intensive nlp tasks. *Advances in Neural Information Processing Systems*, 33:9459–9474.

Xiang Lisa Li and Percy Liang. 2021. Prefix-tuning: Optimizing continuous prompts for generation. In *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)*, pages 4582–4597, Online. Association for Computational Linguistics.

Jenny T Liang, Chenyang Yang, and Brad A Myers. 2024. A large-scale survey on the usability of ai programming assistants: Successes and challenges. In *Proceedings of the 46th IEEE/ACM International Conference on Software Engineering*, pages 1–13.

Andrew Maas, Raymond E Daly, Peter T Pham, Dan Huang, Andrew Y Ng, and Christopher Potts. 2011. Learning word vectors for sentiment analysis. In *Proceedings of the 49th annual meeting of the association for computational linguistics: Human language technologies*, pages 142–150.

Rahman Md. Mostafizer, Yutaka Watanobe, Rage Uday Kiran, and Raihan Kabir. 2021. A stacked bidirectional lstm model for classifying source codes built in mpls. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 75–89. Springer.

Nuala A O'Leary, Eric Cox, J Bradley Holmes, W Ray Anderson, Robert Falk, Vichet Hem, Mirian TN Tsuchiya, Gregory D Schuler, Xuan Zhang, John Torcivia, et al. 2024. Exploring and retrieving sequence and metadata for species across the tree of life with ncbi datasets. *Scientific data*, 11(1):732.

Shirui Pan, Linhao Luo, Yufei Wang, Chen Chen, Jiapu Wang, and Xindong Wu. 2024. Unifying large language models and knowledge graphs: A roadmap. *IEEE Transactions on Knowledge and Data Engineering*.

Sara Papi, Marco Gaido, Matteo Negri, and Luisa Bentivogli. 2024. StreamAtt: Direct streaming speech-to-text translation with attention-based audio history selection. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 3692–3707, Bangkok, Thailand. Association for Computational Linguistics.

Soujanya Poria, Devamanyu Hazarika, Navonil Majumder, and Rada Mihalcea. 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.

Md Mostafizer Rahman, Ariful Islam Shiplu, Yutaka Watanobe, and Md Ashad Alam. 2024a. Roberta-bilstm: A context-aware hybrid model for sentiment analysis. *arXiv preprint arXiv:2406.00367*.

Md. Mostafizer Rahman, Atsushi Shirafuji, and Yutaka Watanobe. 2024b. Big coding data: Analysis, insights, and applications. *IEEE Access*, 12:196010–196026.

Martin Riddell, Ansong Ni, and Arman Cohan. 2024. Quantifying contamination in evaluating code generation capabilities of language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 14116–14137, Bangkok, Thailand. Association for Computational Linguistics.

Joshua Robinson and David Wingate. 2023. Leveraging large language models for multiple choice question answering. In *The Eleventh International Conference on Learning Representations*, Kigali, Rwanda.

Kian Long Tan, Chin Poo Lee, Kalaiarasi Sonai Muthu Anbananthen, and Kian Ming Lim. 2022. Roberta-lstm: a hybrid model for sentiment analysis with transformer and recurrent neural network. *IEEE Access*, 10:21517–21525.

Yijun Tian, Huan Song, Zichen Wang, Haozhu Wang, Ziqing Hu, Fang Wang, Nitesh V Chawla, and Panpan Xu. 2024. Graph neural prompting with large language models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 19080–19088.

Lifu Tu, Semih Yavuz, Jin Qu, Jiacheng Xu, Rui Meng, Caiming Xiong, and Yingbo Zhou. 2024. Unlocking anticipatory text generation: A constrained approach for large language models decoding. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 15532–15548, Miami, Florida, USA. Association for Computational Linguistics.

Priyan Vaithilingam, Tianyi Zhang, and Elena L Glassman. 2022. Expectation vs. experience: Evaluating the usability of code generation tools powered by large language models. In *Chi conference on human factors in computing systems extended abstracts*, pages 1–7.

Jason Wei, Maarten Bosma, Vincent Zhao, Kelvin Guu, Adams Wei Yu, Brian Lester, Nan Du, Andrew M. Dai, and Quoc V Le. 2022a. Finetuned language models are zero-shot learners. In *International Conference on Learning Representations*.

Jason Wei, Yi Tay, Rishi Bommasani, Colin Raffel, Barret Zoph, Sebastian Borgeaud, Dani Yogatama, Maarten Bosma, Denny Zhou, Donald Metzler, Ed H. Chi, Tatsunori Hashimoto, Oriol Vinyals, Percy Liang, Jeff Dean, and William Fedus. 2022b. Emergent abilities of large language models. *Transactions on Machine Learning Research*.

Ran Xu, Wenqi Shi, Yue Yu, Yuchen Zhuang, Yanqiao Zhu, May Dongmei Wang, Joyce C. Ho, Chao Zhang, and Carl Yang. 2024. BMRetriever: Tuning large language models as better biomedical text retrievers. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 22234–22254, Miami, Florida, USA. Association for Computational Linguistics.

Weixiang Yan, Haitian Liu, Yunkun Wang, Yunzhe Li, Qian Chen, Wen Wang, Tingyu Lin, Weishan Zhao, Li Zhu, Hari Sundaram, and Shuiguang Deng. 2024. CodeScope: An execution-based multilingual multitask multidimensional benchmark for evaluating LLMs on code understanding and generation. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 5511–5558, Bangkok, Thailand. Association for Computational Linguistics.

Farah Younas, Muhammad Usman, and Wei Qi Yan. 2022. A deep ensemble learning method for colorectal polyp classification with optimized network parameters. *Applied Intelligence*, pages 1–24.

Wayne Xin Zhao, Kun Zhou, Junyi Li, Tianyi Tang, Xiaolei Wang, Yupeng Hou, Yingqian Min, Beichen Zhang, Junjie Zhang, Zican Dong, et al. 2023. A survey of large language models. *arXiv preprint arXiv:2303.18223*.

Yaqin Zhou, Shangqing Liu, Jingkai Siow, Xiaoning Du, and Yang Liu. 2019. Devign: Effective vulnerability identification by learning comprehensive program semantics via graph neural networks. *Advances in neural information processing systems*, 32.

Albert Ziegler, Eirini Kalliamvakou, X Alice Li, Andrew Rice, Devon Rifkin, Shawn Simister, Ganesh Sittampalam, and Edward Aftandilian. 2022. Productivity assessment of neural code completion. In *Proceedings of the 6th ACM SIGPLAN International Symposium on Machine Programming*, pages 21–29.