Sahand University of Technology

DOR: 20.1001.1.23223146.1403.11.2.5.3

Journal of Nonlinear Systems in Electrical Engineering

Vol. 11, No. 2

Autumn and Winter 2024

ISSN: 2322 – 3146

http:// jnsee.sut.ac.ir

Performance Enhancement of Small Language Models via Distillation and Retrieval-Augmented Generation

Zhila Amini-Sheshdeh*, Helia Mirhosseini

Department of Electrical Engineering, Engineering Faculty, Alzahra University, Tehran, Iran

* Corresponding author email: zh.amini@alzahra.ac.ir

Abstract

In recent years, large language models (LLMs) such as CodeGen have demonstrated remarkable capabilities in automatic code generation. However, their practical deployment is limited by high computational cost and demanding hardware requirements. This study proposes a three-stage pipeline to enhance the performance of the small CodeGen-350M model using supervised fine-tuning, knowledge distillation from CodeGen-6B, and a retrieval-augmented generation (RAG) module to compensate for limited internal knowledge. The results show significant improvements in code quality and consistency, enabling the student model to approach the behavior and accuracy of the larger teacher model. Incorporating RAG notably enhances semantic correctness and completeness in code outputs.

Keywords

Knowledge Distillation; Code Generation; Small Language Models; RAG; FAISS

1. Short Introduction

Large Language Models (LLMs) have revolutionized code generation, yet their massive hardware requirements hinder deployment in resource-limited settings. Although knowledge distillation and Retrieval-Augmented Generation (RAG) are proven methods for model optimization, their synergy for lightweight coding assistants is not fully exploited. This work introduces a three-stage pipeline—comprising SFT, Distillation, and RAG—to significantly boost the performance of the CodeGen-350M model. By transferring teacher-level logic and integrating external code indexes, we enable a student model to achieve competitive accuracy with a minimal parameter footprint.

2. Proposed Work

This study implements a three-stage optimization pipeline for the CodeGen-350M model. The design begins with Supervised Fine-Tuning (SFT) on the Code Alpaca dataset to establish foundational syntax stability. Subsequently, knowledge distillation from a 6B-parameter teacher model is utilized to align the student's output distribution, successfully reducing KL Divergence from 63.31 to 59.62. The final stage integrates a Retrieval-Augmented Generation (RAG) module using FAISS and Jina embeddings to provide real-time external knowledge during inference. Experimental results demonstrate a significant performance improvement, with the average problem-solving score rising from 0.35 to 0.81. Discussion of the results confirms that this framework effectively mitigates parameter constraints, allowing a lightweight model to achieve accuracy levels competitive with much larger architectures.

3. Conclusion

This study proves that a three-stage pipeline—SFT, Knowledge Distillation, and RAG—effectively elevates the performance of small language models for code generation. While SFT establishes fundamental syntax, distillation aligns logic, and RAG compensates for parameter limitations. Results show a significant jump in problem-solving scores from 0.35 to 0.81. Ultimately, this framework enables a 350M-parameter model to achieve accuracy competitive with much larger architectures.

بهبود عملکرد مدل های زبانی کوچک با انتقال دانش و RAG

ژیلا امینی ششده[1،*]، هلیا میرحسینی[2]

[1] استادیار، گروه مهندسی برق، دانشکده فنی و مهندسی، دانشگاه الزهرا (س)، تهران، ایران

[2] گروه مهندسی برق، دانشکده فنی و مهندسی، دانشگاه الزهرا (س)، تهران، ایران

* **آدرس پست الکترونیکی نویسنده مسئول:** zh.amini@alzahra.ac.ir

دانشگاه صنعتی سهند

DOR:
20.1001.1.23223146.1403.11.2.5.3

نشریه سامانه‌های غیرخطی در مهندسی برق

دوره ۱۱ - شماره ۲

پاییز و زمستان ۱۴۰۳

صفحات ۷۳ الی ۹۱

ISSN: 2322-3146

http:// jnsee.sut.ac.ir

تاریخ ارسال: ۱۴۰۴/۹/۲ **تاریخ بازنگری: ۱۴۰۴/۱۰/۱۴** **تاریخ پذیرش: ۱۴۰۴/۱۱/۷**

چکیده

در سال‌های اخیر، مدل‌های زبانی بزرگ (**LLMs**) مانند **CodeGen** توانسته‌اند عملکرد چشمگیری در تولید خودکار کدهای برنامه‌نویسی ارائه دهند ولیکن حجم بالا و نیاز به منابع سخت‌افزاری پرهزینه، کاربرد عملی این مدل‌ها را محدود می‌سازد. هدف این پژوهش، کوچک‌سازی مدل در کنار حفظ کیفیت خروجی و کاهش وابستگی به منابع محاسباتی سنگین است. در این راستا، از مدل **CodeGen-6B** به‌عنوان معلم و مدل **CodeGen-350M** به‌عنوان دانش‌آموز استفاده شده است.

روش پیشنهادی شامل یک خط لوله سه‌مرحله‌ای است که ابتدا مدل دانش‌آموز با استفاده از داده‌های متنی – کدی تحت آموزش نظارت‌شده قرار می‌گیرد تا توانایی پایه‌ای در تولید کد کسب کند. سپس، با به کارگیری انتقال دانش (**Distillation**)، رفتار و توزیع احتمالات خروجی مدل معلم به مدل دانش‌آموز منتقل می‌شود. در نهایت، برای جبران محدودیت ظرفیت پارامتری مدل کوچک، یک سیستم بازیابی همراه با تولید (**RAG**) مبتنی بر ایندکس **FAISS** و امبدینگ‌های کدی، طراحی و به مدل افزوده می‌شود.

نتایج نشان می‌دهد که ترکیب انتقال دانش و بازیابی همراه با تولید منجر به بهبود کیفیت خروجی مدل کوچک شده و امتیاز حل مسائل کدنویسی در نمونه‌های ارزیابی‌شده از ۰/۵ در حالت مدل پایه به ۱/۵ در نسخه مجهز به **RAG** افزایش یافته است. همچنین، تحلیل صحت نحوی کدها با استفاده از ساختار **AST** و کاهش فاصله توزیعی خروجی‌ها بر اساس معیار **Divergence KL** نشان می‌دهد که مدل دانش‌آموز توانسته رفتاری نزدیک به مدل معلم ارائه دهد. این نتایج بیانگر آن است که مدل ۳۵۰ میلیون پارامتری پیشنهادی، بدون افزایش اندازه مدل، قادر به دستیابی کیفیتی قابل رقابت با مدل‌های بزرگ‌تر در تولید کد می‌باشد.

واژه‌های کلیدی

انتقال دانش

مدل‌های زبانی کوچک

تولید کد

بازیابی همراه با تولید

مدل CodeGen

۱- مقدمه

گسترش مدل‌های زبانی بزرگ در سال‌های اخیر، تحول چشمگیری در حوزه پردازش زبان طبیعی، به‌ویژه در زمینه تولید خودکار کد، ایجاد کرده است. مدل‌هایی مانند GPT، Codex و CodeGen که بر پایه معماری ترنسفورمر توسعه یافته‌اند، با آموزش بر روی حجم عظیمی از داده‌های متنی و کدنویسی، قابلیت‌هایی فراتر از تکمیل ساده متن از جمله تولید کدهای پیچیده، حل مسائل برنامه‌نویسی، بازنویسی و بهینه‌سازی کد، و حتی تعامل چندمرحله‌ای در گفتگوهای فنی از خود نشان داده‌اند. این توانمندی‌ها موجب شده است که مدل‌های زبانی بزرگ به ابزارهایی پرکاربرد در توسعه نرم‌افزار، آموزش برنامه‌نویسی و تحلیل رفتار کد تبدیل شوند [۱, ۲, ۵].

با وجود این موفقیت‌ها، استفاده عملی از مدل‌های زبانی بزرگ با چالش‌های قابل توجهی همراه است. مصرف بالای حافظه، نیاز به سخت‌افزارهای قدرتمند، هزینه‌های پردازشی زیاد و دشواری استقرار در محیط‌های واقعی، از جمله محدودیت‌های اصلی این مدل‌ها محسوب می‌شوند [۱, ۳]. به‌ویژه برای کاربران شخصی، سازمان‌های کوچک و سیستم‌های تعبیه‌شده، به‌کارگیری مدل‌هایی با میلیاردها پارامتر عملاً امکان‌پذیر نیست. از این رو، توجه پژوهش‌ها به سمت روش‌هایی معطوف شده است که بتوانند کیفیت مدل‌های کوچک‌تر را بدون افزایش قابل توجه حجم آن‌ها ارتقا دهند.

در میان این رویکردها، انتقال دانش به‌عنوان یکی از راهکارهای مؤثر برای کوچک‌سازی مدل‌های بزرگ همراه با حفظ عملکرد آن‌ها مطرح شده است [۹]. در این روش، مدل بزرگ به عنوان معلم عمل کرده و مدل کوچک‌تر تلاش می‌کند توزیع احتمالات خروجی معلم را فرا گیرد. مطالعات متعدد نشان داده‌اند که به‌کارگیری معیارهایی نظیر واگرایی کولبک - لایبلر (KL Divergence) و استفاده از دمای نرم‌سازی‌شده می‌تواند نقش مهمی در انتقال دقیق‌تر دانش ایفا کند. همچنین، پژوهش‌هایی مانند Self-Instruct و Stanford Alpaca در [۲] و [۳] نشان داده‌اند که استفاده از مراحل آموزش مبتنی بر دستور (Instruction Tuning) پیش از انتقال دانش، موجب بهبود عملکرد مدل دانش‌آموز می‌شود.

در ادامه این مسیر، برخی مطالعات تلاش کرده‌اند انتقال دانش را با روش‌های یادگیری تقویتی، به‌ویژه الگوریتم PPO ترکیب کنند [۱]. با این حال، پیچیدگی محاسباتی بالا، ناپایداری فرآیند آموزش و نیاز شدید به منابع سخت‌افزاری موجب شده است که این روش‌ها در بسیاری از موارد صرفاً در محیط‌های پژوهشی و آزمایشگاهی قابل اجرا باشند.

از سوی دیگر، پژوهش‌های اخیر توجه ویژه‌ای به معماری‌های بازیابی همراه با تولید معطوف کرده‌اند [۴, ۱۰]. در این رویکرد، مدل زبانی به‌جای اتکا صرف به حافظه پارامتری خود، از یک پایگاه دانش بیرونی برای تولید پاسخ بهره می‌گیرد. استفاده از بازیابی همراه با تولید این امکان را فراهم می‌کند که مدل‌های کوچک‌تر به دانشی فراتر از ظرفیت داخلی خود دسترسی داشته باشند و در وظایف پیچیده‌تری مانند تولید کد، پرسش و پاسخ دانش محور و بازیابی اطلاعات، عملکرد بهتری از خود نشان دهند.

۲- کارهای مرتبط

پژوهش‌های مرتبط با موضوع حاضر را می‌توان به سه دسته اصلی مدل‌های زبانی بزرگ برای تولید کد، روش‌های انتقال دانش برای کوچک‌سازی مدل‌ها و معماری‌های بازیابی همراه با تولید تقسیم کرد که هر کدام را بررسی می کنیم.

- **مدل‌های زبانی بزرگ در تولید کد:** مدل‌هایی نظیر GPT، Codex و CodeGen نشان داده‌اند که معماری‌های مبتنی بر ترنسفورمر قادر به یادگیری هم‌زمان زبان طبیعی و زبان‌های برنامه‌نویسی هستند. برای مثال Nijkamp و همکاران در معرفی CodeGen نشان دادند که آموزش مدل بر روی داده‌های گسترده کدنویسی منجر

به بهبود قابل توجه توانایی تولید کد و حل مسائل چندمرحله‌ای می‌شود [۶]. با این حال، این مدل‌ها به دلیل اندازه بسیار بزرگ، محدودیت‌های عملی جدی در استقرار دارند.

- **انتقال دانش و آموزش مبتنی بر دستور:** انتقال دانش به‌عنوان روشی مؤثر برای انتقال توانمندی‌های مدل‌های بزرگ به مدل‌های کوچک‌تر، در مطالعات متعددی بررسی شده است. Devlin و همکاران [۹] و Ouyang و همکاران [۱] نشان داده‌اند که استفاده از توزیع احتمالات نرم مدل معلم می‌تواند موجب بهبود همگرایی مدل دانش‌آموز شود. همچنین، رویکردهایی مانند Self-Instruct و Stanford Alpaca در [۲] و [۳] نشان داده‌اند که آموزش مبتنی بر دستور پیش از انتقال دانش، نقش مهمی در افزایش کیفیت خروجی مدل‌های کوچک ایفا می‌کند. با این وجود، بسیاری از این روش‌ها یا نیازمند داده‌های گسترده انسانی هستند یا به مراحل بهینه‌سازی پیچیده وابسته‌اند.
- **معماری‌های بازیابی همراه با تولید:** رویکرد RAG که نخستین بار توسط Lewis و همکاران معرفی شد [۱۰]، با ترکیب بازیابی اطلاعات و تولید متن، امکان استفاده از دانش بیرونی را برای مدل‌های زبانی فراهم می‌کند. این معماری به‌ویژه برای مدل‌های کوچک اهمیت دارد، زیرا می‌تواند محدودیت ظرفیت پارامتری آن‌ها را جبران کند. مطالعات اخیر نشان داده‌اند که RAG در وظایفی نظیر پرسش و پاسخ دانش‌محور، بازیابی اطلاعات و تولید کد، موجب بهبود دقت و کامل‌بودن پاسخ‌ها می‌شود [۴, ۱۰].

۲-۱- معیارهای ارزیابی مدل‌های تولید کد

ارزیابی کیفیت خروجی در مدل‌های تولید کد، به دلیل ماهیت مولد و وجود چندین راه‌حل صحیح برای یک مسئله، همواره یکی از چالش‌های اصلی پژوهش در این حوزه بوده است. برخلاف مسائل طبقه‌بندی، در تولید کد نمی‌توان صرفاً به معیارهای ساده‌ای مانند دقت[1] اکتفا کرد و پژوهش‌ها به سمت معیارهای ساختاری، نحوی و معنایی سوق یافته‌اند.

در برخی مطالعات اولیه، معیارهایی مبتنی بر تطابق متنی مانند BLEU برای ارزیابی کد تولید شده مورد استفاده قرار گرفته‌اند [۸]. با این حال، پژوهش‌ها نشان داده‌اند که این معیارها همبستگی ضعیفی با صحت عملکرد واقعی کد دارند؛ زیرا دو قطعه کد با ساختار متنی متفاوت می‌توانند از نظر عملکرد کاملاً معادل باشند. به همین دلیل، استفاده از معیارهای مبتنی بر شباهت متنی به‌تنهایی برای ارزیابی مدل‌های تولید کد ناکافی تلقی می‌شود.

در ادامه، پژوهش‌ها به سمت معیارهای مبتنی بر صحت نحوی حرکت کرده‌اند. بررسی قابل‌کامپایل بودن یا معتبر بودن کد از نظر قواعد زبان برنامه‌نویسی، یکی از رویکردهای رایج در این زمینه است. برای مثال، در مطالعات مرتبط با CodeGen و مدل‌های مشابه، از تحلیل ساختار درخت نحو انتزاعی (AST) به‌عنوان معیاری برای سنجش صحت نحوی خروجی‌ها استفاده شده است [۶]. این روش امکان تشخیص کدهایی را فراهم می‌کند که از نظر ظاهری صحیح به نظر می‌رسند اما در عمل قابل اجرا نیستند.

علاوه بر صحت نحوی، ارزیابی معنایی و منطقی کد نیز در پژوهش‌های اخیر مورد توجه قرار گرفته است. برخی مطالعات از اجرای کد بر روی مجموعه‌ای از تست‌ها برای بررسی درستی منطق پیاده‌سازی استفاده کرده‌اند، در حالی که در مواردی دیگر، ارزیابی کیفی مبتنی بر مقایسه با پاسخ مرجع یا تحلیل رفتار تابع انجام شده است [۱, ۶]. این رویکردها به‌ویژه در مسائل برنامه‌نویسی که دارای پاسخ‌های متنوع هستند، اهمیت بیشتری پیدا می‌کنند.

در زمینه ارزیابی فرآیند انتقال دانش، معیارهایی نظیر واگرایی کولبک–لایبلر به‌طور گسترده برای سنجش میزان نزدیکی توزیع احتمالات خروجی مدل دانش‌آموز به مدل معلم به‌کار رفته‌اند [۹]. کاهش مقدار این معیار نشان‌دهنده موفقیت فرآیند انتقال دانش و همگرایی رفتاری میان دو مدل است.

[1] Accuracy

با توجه به این پیشینه، پژوهش حاضر از ترکیبی از معیارهای رایج و پذیرفته‌شده در ادبیات استفاده می‌کند؛ به‌گونه‌ای که صحت نحوی خروجی‌ها از طریق تحلیل AST، نزدیکی رفتاری مدل‌ها با استفاده از KL Divergence و کیفیت منطقی پاسخ‌ها از طریق ارزیابی کیفی نمونه‌های کدنویسی مورد بررسی قرار می‌گیرد. این رویکرد ترکیبی، امکان ارزیابی جامع عملکرد مدل پیشنهادی را بدون وابستگی به معیارهای متنی ناکارآمد فراهم می‌سازد.

برخلاف بسیاری از کارهای پیشین که بر استفاده از مدل‌های بزرگ یا روش‌های یادگیری تقویتی پرهزینه تمرکز دارند، پژوهش حاضر با ترکیب انتقال دانش و معماری بازیابی همراه با تولید در قالب یک خط لوله سبک و قابل اجرا، تلاش می‌کند کیفیت تولید کد را در یک مدل ۳۵۰ میلیون پارامتری ارتقا دهد. این رویکرد، بدون افزایش اندازه مدل و بدون اتکا به الگوریتم‌های یادگیری تقویتی سنگین، مسیری عملی برای بهبود مدل‌های زبانی کوچک ارائه می‌دهد.

۳- رویکرد روش انجام شده در این پژوهش

در این بخش، چارچوب کلی پژوهش، مراحل پیاده‌سازی مدل پیشنهادی و روش‌های ارزیابی در قالب یک ساختار منسجم ارائه می‌گردد.

۳-۱- معماری مدل ها و جزئیات پیاده‌سازی

در این پژوهش، به‌منظور بهبود عملکرد مدل‌های زبانی کوچک در تولید کد، از خانواده مدل‌های CodeGen استفاده شده است. CodeGen یک مدل زبانی مولد مبتنی بر معماری ترنسفورمر از نوع فقط - رمزگشا[1] است که برای تولید خودرگرسیو[2] دنباله‌های متنی و کدهای برنامه‌نویسی طراحی شده است. در این معماری، مدل با پیش‌بینی نشانه[3] بعدی بر اساس توکن‌های قبلی، قادر به تولید پیوسته و ساختار یافته کد می‌باشد.

در چارچوب پیشنهادی، مدل CodeGen-6B به‌عنوان مدل معلم و مدل CodeGen-350M به‌عنوان مدل دانش‌آموز مورد استفاده قرار گرفته‌اند. هر دو مدل از یک معماری مشترک و نشانه ساز[4] یکسان بهره می‌برند که این هم‌خانواده بودن، نقش مهمی در پایداری فرآیند انتقال دانش و کاهش ناسازگاری‌های ساختاری میان مدل‌ها ایفا می‌کند. استفاده از نشانه ساز مشترک همچنین موجب شده است که کلیه ورودی‌ها به‌صورت یکنواخت پردازش شوند و نیازی به بازآموزی نشانه ساز وجود نداشته باشد.

لازم به تأکید است که در این پژوهش از مدل BERT استفاده نشده است. مدل BERT یک ترنسفورمر دوسویه مبتنی بر معماری رمزگذار[5] است که عمدتاً برای وظایف درک زبان طبیعی مانند طبقه‌بندی متن، استخراج ویژگی و پاسخ به پرسش به‌کار می‌رود. از آنجا که هدف اصلی پژوهش حاضر، تولید مرحله به مرحله کد بر اساس نشانه های قبلی است، استفاده از معماری‌های رمزگشا محور مانند CodeGen که به‌طور خاص برای وظایف مولد طراحی شده‌اند، انتخاب مناسب‌تری محسوب می‌شود.

پیاده‌سازی و ارزیابی مدل‌ها در محیط Google Colab Pro و با استفاده از پردازنده گرافیکی NVIDIA A100 انجام شده است. به منظور تضمین قابلیت بازتولید نتایج، مقدار seed برابر با ۴۲ برای تمامی کتابخانه‌های مورد استفاده تنظیم شد و تنظیمات محاسباتی قطعی در CUDA فعال گردید. این تنظیمات شامل فعال‌سازی حالت deterministic و غیرفعال‌سازی benchmark در کتابخانه cuDNN می‌باشد.

در این پژوهش، پیاده‌سازی مدل‌ها با استفاده از کتابخانه‌های PyTorch و Transformers (نسخه ۴.۳۸ یا بالاتر) انجام شده و از ابزارهایی نظیر PEFT و Accelerate برای مدیریت بهینه مدل‌ها استفاده شده است. نسخه نهایی مدل دانش‌آموز، یک مدل

[1] Decoder-only Transformer
[2] Autoregressive
[3] token
[4] tokenizer
[5] Encoder-based

CodeGen-350M ادغام‌شده[1] است که در مرحله ارزیابی صرفاً در حالت استنتاج[2] مورد استفاده قرار گرفته و هیچ‌گونه آموزش یا به‌روزرسانی پارامتر در این مرحله انجام نشده است. ازاین‌رو، پارامترهایی نظیر تعداد epoch، batch size یا زمان‌بندی بهینه‌ساز در مرحله ارزیابی تعریف نشده‌اند.

برای جبران محدودیت ظرفیت پارامتری مدل دانش‌آموز، یک سازوکار بازیابی همراه با تولید در مرحله استنتاج به مدل افزوده شده است. در این چارچوب، یک پایگاه دانش بیرونی مبتنی بر مسائل و پاسخ‌های برنامه‌نویسی ایجاد شده و نمایش برداری اسناد با استفاده از مدل‌های Sentence Transformer انجام گرفته است. بردارهای حاصل در یک ایندکس برداری با استفاده از کتابخانه FAISS ذخیره شده‌اند تا امکان بازیابی سریع نمونه‌های مرتبط فراهم شود. در زمان تولید پاسخ، اسناد بازیابی‌شده به‌صورت پویا در قالب پرامپت به ورودی مدل افزوده می‌شوند و مدل دانش‌آموز بدون افزایش تعداد پارامترها از دانش بیرونی بهره می‌گیرد.

در طراحی این چارچوب، از هیچ‌گونه الگوریتم یادگیری تقویتی، مدل پاداش یا روش‌هایی نظیر PPO استفاده نشده است. هدف اصلی، ارائه یک خط لوله سبک، قابل تکرار و عملی بوده است که بتواند با اتکا به معماری ترنسفورمر و ترکیب انتقال دانش و RAG، کیفیت تولید کد را در مدل‌های کم‌پارامتر بهبود دهد.

۳-۲- آموزش نظارت‌شده

آموزش نظارت‌شده[3] (SFT) به‌عنوان نخستین مرحله از خط لوله پیشنهادی، با هدف ایجاد یک رفتار پایه‌ای پایدار در مدل دانش‌آموز CodeGen-350M انجام شد. با توجه به محدودیت ظرفیت پارامتری مدل‌های زبانی کوچک، آغاز فرآیند آموزش با روش‌های پیچیده‌تر می‌تواند منجر به ناپایداری همگرایی شود. از این‌رو، مرحله SFT به‌عنوان گام ابتدایی طراحی شد تا مدل بتواند نگاشت اولیه میان دستور متنی و کد برنامه‌نویسی را فراگیرد و به درکی حداقلی اما منسجم از ساختار کد دست یابد [۱۱].

برای آموزش نظارت‌شده، از مجموعه داده Code Alpaca استفاده شد. این مجموعه داده نخستین‌بار توسط Taori و همکاران معرفی شده و شامل چند هزار نمونه دستور–پاسخ در حوزه برنامه‌نویسی است که با الهام از رویکرد Self-Instruct تولید شده‌اند [۳]. داده‌ها عمدتاً به زبان پایتون ارائه شده‌اند و طیفی از وظایف متداول برنامه‌نویسی، از جمله تعریف توابع، استفاده از ساختارهای شرطی و تکرار، و پیاده‌سازی الگوریتم‌های ساده را پوشش می‌دهند. قالب دستور–پاسخ این مجموعه، آن را به گزینه‌ای مناسب برای آموزش اولیه مدل‌های مولد کد تبدیل کرده است [۳].

پیش از استفاده، داده‌ها مورد پاک‌سازی قرار گرفتند. نمونه‌های دارای ورودی ناقص، پاسخ‌های چندبخشی نامنسجم، کدهای دارای خطای نحوی و مواردی که ساختار کد آن‌ها معتبر نبود حذف شدند. این فرآیند، که در مطالعات پیشین نیز به‌عنوان یک گام ضروری برای آموزش پایدار مدل‌های زبانی معرفی شده است، موجب کاهش نویز داده‌ها و افزایش کیفیت یادگیری گردید [۱۲,۱].

پس از پاک‌سازی، داده‌ها با استفاده از نشانه ساز اختصاصی CodeGen به نشانه‌های عددی تبدیل شدند. برای یکنواخت‌سازی طول توالی‌ها و امکان پردازش دسته‌ای، عمل پدینگ انجام شد و مقدار max_length به‌صورت ثابت تنظیم گردید. قالب ورودی به‌گونه‌ای طراحی شد که متن دستور و پاسخ در یک توالی واحد قرار گیرند و مدل به‌صورت یک مدل زبانی علّی (Causal Language Model) آموزش داده شود. این شیوه آموزش، که در مدل‌های مولد مبتنی بر ترنسفورمر رایج است، امکان یادگیری مرحله‌به‌مرحله تولید کد را فراهم می‌سازد [۱۳].

آموزش نظارت‌شده با استفاده از بهینه‌ساز AdamW انجام شد که به‌طور گسترده در آموزش مدل‌های زبانی بزرگ و کوچک مورد استفاده قرار می‌گیرد [۱۴]. نرخ یادگیری برابر با ۲×۱۰^{-۵} تنظیم شد و برای افزایش پایداری همگرایی، ۵ درصد از کل گام‌های آموزش به مرحله warmup اختصاص یافت. به‌منظور جلوگیری از نوسانات شدید گرادیان، از تکنیک‌های gradient clipping

[1] merged
[2] Inference
[3] Supervised Fine-Tuning

و weight decay استفاده گردید. همچنین، برای مدیریت محدودیت حافظه GPU و امکان استفاده از batch های مؤثرتر، تکنیک gradient accumulation به‌کار گرفته شد؛ رویکردی که در آموزش مدل‌های حجیم در محیط‌های کم‌منبع توصیه شده است [۱۵].

روند تغییرات مقدار Loss در طول مرحله SFT نشان داد که مدل دانش‌آموز به‌تدریج توانسته است الگوهای پایه‌ای مربوط به تبدیل دستور متنی به کد را فراگیرد. کاهش تدریجی Loss بیانگر شکل‌گیری یک رفتار اولیه پایدار در مدل است که اجرای مراحل بعدی خط لوله، از جمله انتقال دانش و بهره‌گیری از دانش بیرونی، را امکان‌پذیر می‌سازد. نتایج این مرحله با یافته‌های گزارش‌شده در پژوهش‌های مشابه درباره آموزش اولیه مدل‌های مولد کد هم‌راستا است و اهمیت SFT را به‌عنوان یک پیش‌نیاز ضروری تأیید می‌کند [۳, ۱۱].

۳-۳- انتقال دانش

پس از اجرای مرحله آموزش نظارت شده و ایجاد یک رفتار پایه‌ای پایدار در مدل دانش‌آموز، مرحله انتقال دانش به‌عنوان گام دوم خط لوله پیشنهادی اجرا شد. در این پژوهش، مرحله آموزش نظارت‌شده نقش یک warmup آموزشی را ایفا می‌کند؛ به این معنا که مدل دانش‌آموز پیش از ورود به فرآیند انتقال دانش، درکی اولیه از ساختار کد و نگاشت دستور متنی به کد کسب می‌کند. این رویکرد، که در مطالعات مرتبط با هم راستاسازی و بهبود مدل‌های زبانی نیز مورد تأکید قرار گرفته است، موجب کاهش نوسانات آموزشی و افزایش پایداری همگرایی در مراحل بعدی می‌شود [۱, ۲].

در این مرحله، مدل CodeGen-6B به‌عنوان مدل معلم و مدل CodeGen-350M به‌عنوان مدل دانش‌آموز مورد استفاده قرار گرفتند. انتخاب این دو مدل از یک خانواده معماری، با ابعاد embedding و نشانه ساز یکسان، شرایط مناسبی را برای انتقال پایدار دانش فراهم کرده است. هم‌راستایی معماری میان مدل‌ها باعث شد اختلافات ساختاری به حداقل برسد و فرآیند انتقال دانش بدون نیاز به تطبیق مجدد فضای ورودی انجام شود [۶].

در فرآیند انتقال دانش، هدف آن است که مدل دانش‌آموز رفتار پیش‌بینی مدل معلم را تا حد امکان بازتولید کند. به‌جای استفاده از برچسب‌های سخت، از توزیع احتمالات خروجی مدل معلم به‌عنوان منبع اطلاعات استفاده شد، رویکردی که در ادبیات انتقال دانش به‌عنوان یک روش مؤثر برای انتقال اطلاعات غنی مدل‌های بزرگ معرفی شده است [۸].

برای این منظور، خروجی‌های مدل معلم با اعمال دمای نرم‌سازی $T=2.0$ هموارسازی شدند. استفاده از دماهای بزرگ‌تر از ۱ موجب نرم‌تر شدن توزیع احتمالات و برجسته‌تر شدن روابط نسبی میان نشانه ها می‌شود، در حالی که مقادیر بسیار بالا می‌توانند منجر به از دست رفتن اطلاعات تمایزبخش گردند. مقدار $T=2.0$ به‌عنوان یک مقدار متعادل انتخاب شد تا امکان انتقال الگوهای تصمیم‌گیری مدل بزرگ به مدل کوچک فراهم شود، بدون آنکه توزیع خروجی بیش‌ازحد هموار گردد [۱, ۸].

پس از اعمال نرم‌سازی، فاصله میان توزیع احتمالات خروجی مدل دانش‌آموز و مدل معلم با استفاده از معیار واگرایی کولبک–لایبلر محاسبه شد. این معیار به‌طور گسترده در فرآیندهای انتقال دانش برای سنجش نزدیکی توزیع‌های خروجی به‌کار می‌رود و در این پژوهش به‌عنوان تابع هزینه مرحله انتقال دانش استفاده شده است [۸].

استفاده از برچسب‌های نرم[1] این امکان را فراهم کرد که مدل دانش‌آموز به‌جای تقلید صرف پاسخ نهایی، ساختار احتمالاتی تصمیم‌گیری مدل معلم را فراگیرد. این ساختار شامل ترجیحات نسبی، میان نشانه‌های مختلف و الگوهای تولید کد است که استخراج آن‌ها از داده‌های خام به‌تنهایی امکان‌پذیر نیست. چنین رویکردی در مطالعات مرتبط با انتقال دانش و هم‌راستاسازی مدل‌های زبانی به‌عنوان یک مزیت کلیدی معرفی شده است [۱, ۲].

[1] Soft Labels

مرحله انتقال دانش در این پژوهش بدون استفاده از الگوریتم‌های یادگیری تقویتی، مدل پاداش یا معماری critic اجرا شد. این تصمیم با هدف کاهش پیچیدگی محاسباتی و افزایش قابلیت اجرا در محیط‌های کم‌منبع اتخاذ گردید. این رویکرد در تضاد با روش‌های مبتنی بر RLHF می باشد و اجرای آن‌ها مستلزم منابع محاسباتی گسترده و فرآیندهای آموزشی پیچیده است [۱].

به دلیل اجرای مرحله آموزش نظارت شده به‌عنوان warmup ، فرآیند انتقال دانش با پایداری مناسبی همراه بود و همگرایی مدل دانش‌آموز به سمت رفتار مدل معلم به‌صورت تدریجی و کنترل‌شده انجام گرفت. نتایج کمی این مرحله، که در بخش ارزیابی ارائه شده‌اند، نشان می‌دهند مقدار KL Divergence در طول فرآیند انتقال دانش روندی نزولی داشته و بیانگر نزدیک‌تر شدن توزیع احتمالات خروجی مدل دانش‌آموز به مدل معلم است.

در مجموع، مرحله انتقال دانش نقش کلیدی در ارتقای توان مدل CodeGen-350M ایفا کرده است. ترکیب آموزش نظارت‌شده به‌عنوان warmup و انتقال دانش مبتنی بر توزیع احتمالات نرم، امکان انتقال مؤثر دانش از مدل بزرگ به مدل کوچک را فراهم کرده است. این نتایج نشان می‌دهند که بدون اتکا به روش‌های پرهزینه‌ای مانند یادگیری تقویتی و تنها با استفاده از سازوکارهای استاندارد انتقال دانش، می‌توان کیفیت تولید کد را در مدل‌های کم‌پارامتر به‌طور معناداری بهبود داد [۱, ۲, ۶].

۳-٤- بازیابی همراه با تولید

در مرحله سوم خط لوله پیشنهادی، به‌منظور جبران محدودیت ظرفیت حافظه پارامتری مدل دانش‌آموزCodeGen-350M، از سازوکار بازیابی همراه با تولید استفاده شد. در این معماری، مدل زبانی علاوه بر دانش ذخیره‌شده در پارامترهای خود، از یک پایگاه دانش بیرونی برای تولید پاسخ بهره می‌گیرد. این رویکرد به‌ویژه برای مدل‌های زبانی کوچک که توان ذخیره دانش گسترده در پارامترهای آن‌ها محدود است، امکان بهبود کامل‌بودن و انسجام منطقی خروجی را فراهم می‌سازد [۵,۱۰].

برای ایجاد پایگاه دانش بیرونی، از مجموعه MBPP[1] استفاده شد. هر نمونه این مجموعه شامل شرح مسئله، نام تابع پیشنهادی و یک پیاده‌سازی مرجع به زبان پایتون است. پیش از استفاده، داده‌ها پاک‌سازی شدند و نمونه‌های دارای توضیح ناقص، کد نامعتبر یا ساختار نامنسجم حذف گردیدند تا کیفیت اسناد بازیابی‌شده افزایش یابد.

مجموعه MBPP طیفی از وظایف پایه برنامه‌نویسی پایتون مانند تعریف توابع، استفاده از ساختارهای شرطی و تکرار و پیاده‌سازی الگوریتم‌های ساده را پوشش می‌دهد. با این حال، از نظر حجم و تنوع دامنه، این مجموعه در مقیاس آزمایشگاهی قرار دارد. بنابراین، نتایج این پژوهش عمدتاً بیانگر عملکرد روش پیشنهادی در چارچوب وظایف استاندارد MBPP بوده و ارزیابی تعمیم‌پذیری در دامنه‌های گسترده‌تر به‌عنوان مسیر پژوهشی آینده مطرح می‌شود.

برای امکان بازیابی مبتنی بر شباهت معنایی، اسناد پایگاه دانش به نمایش‌های برداری ثابت‌طول تبدیل شدند. در این پژوهش، از امبدینگ‌های کدی Jina استفاده شد که برای نمایش معنایی متون برنامه‌نویسی و کد طراحی شده‌اند. هر نمونه از دیتاستMBPP، شامل ترکیبی از شرح مسئله، نام تابع و کد مرجع، به یک بردار عددی با بعد ثابت نگاشت شد.

این نمایش برداری امکان محاسبه شباهت میان مسائل کدنویسی را نه‌تنها بر اساس شباهت سطحی متنی، بلکه بر اساس ساختار و منطق برنامه‌نویسی فراهم می‌کند و برای کاربرد تولید کد اهمیت ویژه‌ای دارد.

بردارهای امبدینگ تولیدشده در یک ایندکس برداری مبتنی بر FAISS ذخیره شدند تا امکان جست‌وجوی سریع نزدیک‌ترین همسایه‌ها فراهم گردد. برای این منظور، از ساختار HNSW استفاده شد که تعادل مناسبی میان سرعت بازیابی و مصرف حافظه ایجاد می‌کند و برای کاربردهای عملی مناسب است.

[1] Mostly Basic Python Problems

پارامترهای اصلی این بخش به این صورت تنظیم شدند که نوع ایندکس FAISS مبتنی برHNSW و فضای برداری($\mathbb{R}^d$) با بعد ثابت مطابق مدل امبدینگ استفاده شده است همچنین تعداد اسناد بازیابی‌شده برای هر ورودی $k = 3$ است و معیار شباهت هم شباهت برداری (مطابق تنظیمات پیش‌فرض پیاده‌سازی) می باشد.

پارامترهای داخلی HNSW در مقدار پیش‌فرض کتابخانه FAISS باقی ماندند تا سادگی پیاده‌سازی و پایداری سیستم حفظ شود.

فرآیند بازیابی و تزریق زمینه به مدل به این گونه است که در زمان استنتاج، ابتدا ورودی کاربر (شرح مسئله) به بردار امبدینگ تبدیل می‌شود. سپس، با جست‌وجو در ایندکسFAISS، سه سند نزدیک‌تر از نظر شباهت برداری بازیابی می‌گردند. این اسناد به‌عنوان زمینه کمکی[1] در اختیار مدل قرار می‌گیرند تا اطلاعات مرتبط با مسائل مشابه یا الگوهای حل پیشین را در فرآیند تولید پاسخ مورد استفاده قرار دهد.

برای جلوگیری از اختلاط نامناسب اطلاعات و افزایش شفافیت نقش هر بخش، اسناد بازیابی‌شده با استفاده از یک قالب پرامپت ساخت‌یافته به ورودی مدل تزریق شدند. در این قالب، متن مسئله اصلی و زمینه بازیابی‌شده به‌صورت صریح از یکدیگر جدا شده‌اند. ساختار پرامپت شامل سه بخش اصلی شرح مسئله[2]، بخش مستقل شامل اسناد بازیابی‌شده[3] با جداکننده‌های ثابت و بخش تولید پاسخ[4] می باشد.

این تفکیک صریح موجب می‌شود مدل از اسناد بازیابی‌شده به‌عنوان اطلاعات کمکی استفاده کند، بدون آنکه ساختار مسئله اصلی دچار ابهام شود. چنین طراحی‌ای مطابق با معماری‌های استاندارد RAG در ادبیات پژوهشی بوده و نقش مؤثری در بهبود کیفیت تولید دارد [۵,۱۰].

با وجود مزایای RAG، استفاده از این رویکرد با محدودیت‌هایی همراه است. نخست، افزودن مرحله بازیابی باعث افزایش تأخیر زمانی در فرآیند استنتاج می‌شود. دوم، عملکرد RAG به‌شدت به کیفیت و پوشش پایگاه دانش وابسته است و در صورت محدود بودن دامنه داده‌ها، میزان بهبود کاهش می‌یابد. سوم، امکان بازیابی اسناد نامرتبط وجود دارد که می‌تواند در برخی موارد موجب انحراف مدل و کاهش دقت خروجی شود. این محدودیت‌ها ذاتی معماری RAG هستند و باید در تفسیر نتایج و توسعه‌های آینده مدنظر قرار گیرند [۵,۱۰].

در این پژوهش، RAG به‌عنوان یک راهکار سبک و عملی برای ارتقای عملکرد مدل کوچک پس از آموزش نظارت‌شده و انتقال دانش به‌کار گرفته شد. مقایسه عملکرد مدل دانش‌آموز در حالت‌های بدون RAG و مجهز به RAG نشان می‌دهد که افزودن دانش بیرونی نقش مهمی در افزایش صحت نحوی و کامل‌بودن منطقی خروجی‌ها دارد. این نتایج، برتری رویکرد پیشنهادی را نسبت به تنظیم ساده مدل بدون بازیابی نشان می‌دهد و ضرورت استفاده از RAG را به‌عنوان مکمل انتقال دانش برجسته می‌سازد.

۳-۵- ارزیابی و پروتکل آزمایش

به‌منظور سنجش دقیق تأثیر هر یک از مؤلفه‌های خط لوله پیشنهادی، یک چارچوب ارزیابی چندمرحله‌ای طراحی شد که عملکرد مدل دانش‌آموز CodeGen-350M را در حالت‌های مختلف مورد بررسی قرار می‌دهد. هدف از این ارزیابی آن است که مشخص

[1] context
2 Instruction
3 Retrieved Context
4 Response

شود هر مرحله از خط لوله (آموزش نظارت‌شده، انتقال دانش و بازیابی همراه با تولید) تا چه حد در بهبود کیفیت تولید کد نقش داشته است.

ارزیابی مدل در سه سناریوی مجزا انجام شد:

۱) مدل دانش‌آموز پس از آموزش نظارت‌شده (SFT-only)

۲) مدل دانش‌آموز پس از آموزش نظارت‌شده و انتقال دانش، بدون استفاده ازRAG

۳) مدل دانش‌آموز پس از آموزش نظارت‌شده، انتقال دانش و با استفاده از RAG (مدل نهایی)

این تفکیک امکان تحلیل سهم هر مؤلفه را به‌صورت مستقل فراهم می‌سازد و نشان می‌دهد که بهبود نهایی حاصل از ترکیب چه مراحلی بوده است.

برای ارزیابی عملکرد مدل در تولید کد، از مجموعه MBPP استفاده شد. این مجموعه شامل مسائل استاندارد برنامه‌نویسی پایتون با ساختار مشخص ورودی و خروجی است و به‌طور گسترده در ارزیابی مدل‌های مولد کد به‌کار می‌رود. انتخاب MBPP امکان مقایسه نتایج با پژوهش‌های پیشین در حوزه تولید کد را فراهم می‌سازد [۶].

با توجه به ماهیت آزمایشگاهی مجموعهMBPP ، ارزیابی در این پژوهش بر کیفیت حل مسائل پایه و نیمه‌پایه متمرکز بوده است و تعمیم نتایج به دامنه‌های بسیار پیچیده‌تر، نیازمند بررسی‌های تکمیلی است.

به‌دلیل محدودیت معیارهای خودکار رایج مانند BLEU یا ROUGE در سنجش کیفیت کد، از معیارهایی استفاده شد که با ماهیت برنامه‌نویسی سازگارتر هستند.

۱) صحت نحوی(Syntax Validity)

در این معیار بررسی می‌شود که آیا کد تولیدشده از نظر قواعد زبان پایتون معتبر است یا خیر. برای این منظور، خروجی مدل با استفاده از تحلیل‌گر نحوی مبتنی بر AST بررسی شد. این معیار از تولید کدهایی که ظاهراً درست اما غیرقابل اجرا هستند جلوگیری می‌کند.

۲) صحت منطقی(Semantic Correctness)

در این معیار بررسی می‌شود که آیا کد تولیدشده منطق مسئله را به‌درستی پیاده‌سازی کرده است یا خیر. این ارزیابی به‌صورت مقایسه خروجی مدل با رفتار مورد انتظار مسئله انجام شد و برای مسائلی که پاسخ‌های متنوع دارند (مانند توابع بازگشتی یا لیست Comprehension) اهمیت ویژه‌ای دارد.

۳) نزدیکی توزیعی به مدل معلم

برای سنجش موفقیت مرحله انتقال دانش، مقدار KL Divergence میان توزیع احتمالات خروجی مدل دانش‌آموز و مدل معلم محاسبه شد. کاهش این مقدار نشان‌دهنده نزدیک‌تر شدن رفتار مدل دانش‌آموز به مدل بزرگ‌تر است [۱, ۸].

تمامی مدل‌ها در مرحله ارزیابی صرفاً در حالت استنتاج مورد استفاده قرار گرفتند و هیچ‌گونه به‌روزرسانی پارامتر یا تنظیم مجدد انجام نشد. تنظیمات تولید (مانند طول توالی خروجی و پارامترهای نمونه‌گیری) در تمامی سناریوها ثابت نگه داشته شدند تا مقایسه منصفانه‌ای میان مدل‌ها برقرار شود.

ارزیابی‌ها به‌صورت یکسان روی مجموعه مسائل مشخص انجام شد و نتایج به‌صورت مقایسه‌ای میان سناریوهای مختلف گزارش گردید. همچنین برای بررسی برتری روش پیشنهادی، نتایج مدل نهایی با سناریوهای پایه مقایسه شد. این سناریوهای پایه شامل مدل دانش‌آموز با تنها آموزش نظارت‌شده و مدل دانش‌آموز با آموزش نظارت‌شده و انتقال دانش، بدون RAG می باشد.

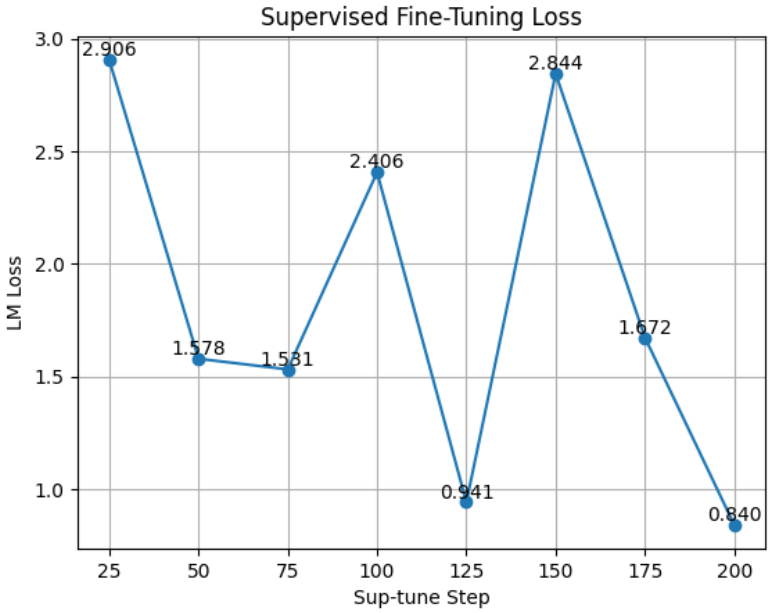
این مقایسه نشان می‌دهد که افزودن دانش بیرونی از طریق RAG ، در کنار انتقال دانش، منجر به افزایش معنادار صحت نحوی و کامل‌بودن منطقی خروجی‌ها شده است. چنین مقایسه‌ای امکان تحلیل دقیق نقش RAG را فراهم می‌سازد و برتری چارچوب پیشنهادی را نسبت به تنظیم ساده مدل‌های کوچک نشان می‌دهد [۵,۶].

٤- نتایج و تحلیل کلی

هدف از بخش نتایج، ارائه‌ی شواهد تجربی برای پاسخ به این پرسش اصلی است که هر یک از مؤلفه‌های خط لوله پیشنهادی (آموزش نظارت‌شده، انتقال دانش و بازیابی همراه با تولید) تا چه حد در بهبود عملکرد مدل زبانی کوچک CodeGen-350M در تولید کد نقش داشته‌اند. به همین منظور، نتایج به‌صورت مرحله‌به‌مرحله گزارش می‌شوند تا اثر هر بخش به‌طور مستقل قابل مشاهده و تحلیل باشد.

در این چارچوب، ابتدا نتایج مربوط به آموزش نظارت‌شده (SFT) ارائه می‌شود زیرا این مرحله به‌عنوان پایه و نقطه شروع کل خط لوله عمل کرده و نقش یک مرحله آماده‌سازی (warm-up) را برای مراحل بعدی ایفا می‌کند. سپس در بخش‌های بعدی، تأثیر انتقال دانش و افزوده‌شدن RAG بر رفتار مدل بررسی خواهد شد. این رویکرد امکان تحلیل تدریجی روند بهبود مدل و مقایسه منصفانه میان سناریوهای مختلف را فراهم می‌سازد.

برای ارزیابی عملکرد مدل دانش‌آموز CodeGen-350M در مرحله آموزش نظارت‌شده، تغییرات مقدار Loss در مدل زبانی (LM Loss) در طول گام‌های آموزش مورد بررسی قرار گرفت. نمودار ارائه‌شده روند تغییرات Loss را در گام‌های منتخب فرآیند SFT نشان می‌دهد .



شکل ۱: نمودار Loss بر حسب گام آموزش در یادگیری نظارت شده

شکل ارائه‌شده روند تغییرات LM Loss مدل CodeGen-350M را در طول گام‌های منتخب آموزش نظارت‌شده نشان می‌دهد. محور افقی نمایانگر گام‌های آموزش[1] و محور عمودی مقدار Loss مدل زبانی است. هر نقطه نشان‌دهنده مقدار میانگین Lossدر یک بازه مشخص از آموزش می‌باشد.

در ابتدای فرآیند (گام ۲۵)، مقدار Loss نسبتاً بالا و در حدود ۲/۹۰۶ مشاهده می‌شود. این مقدار بالا قابل انتظار است، زیرا مدل دانش‌آموز در این مرحله هنوز در حال تطبیق با توزیع داده‌های Instruction–Response بوده و دانش پیشینی محدودی درباره نگاشت دستورات متنی به کد دارد.

با پیشرفت آموزش تا گام‌های ۵۰ و ۷۵، کاهش قابل توجه Loss تا حدود ۱/۵۸ و ۱/۵۳ مشاهده می‌شود. این کاهش سریع نشان‌دهنده آن است که مدل در مراحل ابتدایی، الگوهای پایه‌ای کدنویسی را به‌سرعت فراگرفته است؛ الگوهایی مانند ساختار کلی توابع، نحوه استفاده از ورودی‌ها و تولید خروجی‌های نحوی معتبر. این رفتار با انتظار از یک مرحله warm-up کاملاً هم‌راستا است.

در گام ۱۰۰، افزایش مجدد Loss تا حدود ۲/۴۰۶ مشاهده می‌شود. این جهش ناگهانی، نشانه واگرایی یا ناپایداری آموزش نیست، بلکه به‌طور مستقیم به ماهیت ناهمگن داده‌های آموزشی مربوط می‌شود. در این بازه، مدل با نمونه‌هایی مواجه شده است که دارای پیچیدگی منطقی بالاتر بوده‌اند (مانند توابع بازگشتی یا دستورات چندمرحله‌ای). از آنجا که آموزش به‌صورت mini-batch انجام شده است، حضور چند نمونه دشوار در یک batch می‌تواند باعث افزایش موقتی Loss گردد.

در گام ۱۲۵، افت چشمگیر Loss تا مقدار حدود ۰/۹۴۱ مشاهده می‌شود که کمترین مقدار در کل نمودار است. این کاهش نشان می‌دهد که مدل پس از مواجهه با نمونه‌های پیچیده‌تر، توانسته است خود را با توزیع جدید داده‌ها تطبیق دهد و الگوهای پیچیده‌تر را نیز جذب کند. این رفتار بیانگر توانایی تطبیق‌پذیری مدل، حتی با ظرفیت پارامتری محدود است.

در گام ۱۵۰، مجدداً یک جهش نسبتاً شدید در Loss حدود ۲/۸۴۴ دیده می‌شود. این جهش نیز ناشی از تغییر ترکیب داده‌های batchها و ورود نمونه‌هایی با ساختار غیرمتعارف یا طول توالی بالاتر بوده است. حساسیت مدل‌های کوچک‌تر به چنین تغییراتی امری شناخته‌شده است و در اینجا نیز مشاهده می‌شود.

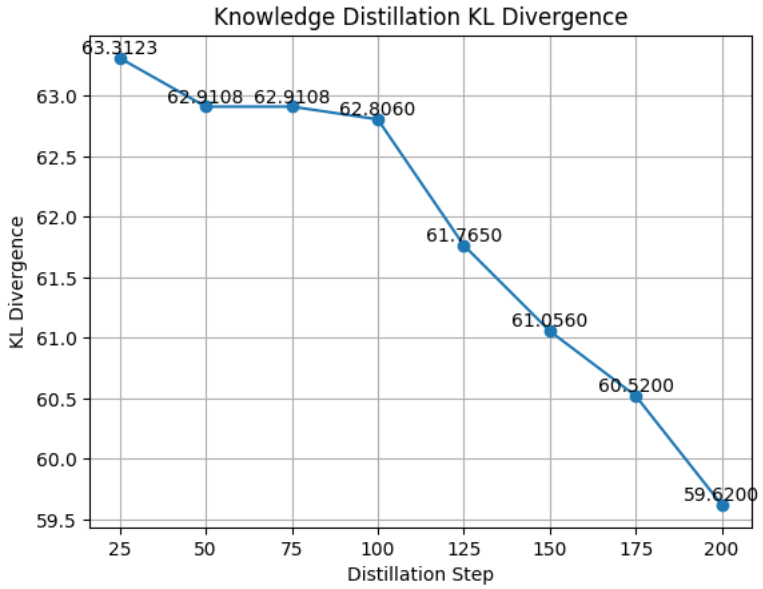
[1] Supervised Fine-Tuning Steps

پس از این جهش، روند کاهش Loss در گام‌های ۱۷۵ و ۲۰۰ ادامه یافته و در نهایت به مقدار حدود ۰/۸۴۰ در انتهای آموزش می‌رسد. رسیدن به چنین مقدار پایینی از Loss نشان می‌دهد که مدل توانسته است یک رفتار پایه‌ای پایدار و قابل اتکا در تولید کد کسب کند.

نکته کلیدی آن اســـت که تمامی جهش‌های Loss موقتی بوده‌اند **و** هیچ‌کدام منجر به واگرایی پایدار یا افزایش روندی خطا نشـده‌اند. در مقابل، کاهش نهایی Loss نسبت به مقدار اولیه، نشـان‌دهنده موفقیت مرحله آموزش نظارت‌شـده در ایفای نقش خود به‌عنوان مرحله آماده‌سازی[1] برای انتقال دانش است.

در مجموع، این نمودار نشـــان می‌دهد که اگرچه آموزش نظارت‌شـــده بر روی یک مدل کوچک با داده‌های متنوع همراه با نوسـانات مقطعی اسـت، اما روند کلی یادگیری پایدار بوده و مدل در پایان این مرحله به سـطحی از آمادگی رسـیده اسـت که امکان ورود موفق به مرحله انتقال دانش از مدل معلم را فراهم می‌سازد.

در این مرحله، اثر مرحله انتقال دانش از مدل معلم CodeGen-6B به مدل دانش‌آموز CodeGen-350M با اسـتفاده از معیار KL Divergence مورد بررسـی قرار می‌گیرد. هدف از این ارزیابی، سـنجش میزان نزدیک‌شـدن توزیع احتمالات خروجی مدل دانش‌آموز به توزیع خروجی مدل معلم اســـت؛ معیاری که به‌طور گســـترده برای ارزیابی موفقیت فرآیند انتقال دانش در مدل‌های زبانی استفاده می‌شود.



شکل ۲: نمودار KL Divergence بر حسب گام آموزش در انتقال دانش

شـــکل ارائه‌شـــده روند تغییرات مقدار KL Divergence را در طول گام‌های انتقال دانش نشـــان می‌دهد. محور افقی بیانگر گام‌های آموزش انتقال دانش و محور عمودی مقدار KL Divergence بین خروجی‌های دو مدل است.

در ابتدای این مرحله (گام ۲۵)، مقدار KL Divergence در حدود ۶۳/۳۱ مشـاهده می‌شـود که نشـان‌دهنده فاصـله قابل توجه میان رفتار مدل دانش‌آموز و مدل معلم است. این فاصله با توجه به اختلاف شدید در ظرفیت پارامتری دو مدل (۳۵۰ میلیون در برابر ۶ میلیارد پارامتر) و نیز آغاز این مرحله بلافاصله پس از آموزش نظارت‌شده، امری قابل انتظار محسوب می‌شود.

در گام‌های ابتدایی (۲۵ تا ۱۰۰)، کاهش KL Divergence نسبتاً آهسته و همراه با نوسانات محدود مشاهده می‌شود؛ به‌طوری که مقدار این معیار از حدود ۶۳/۳۱ به حدود ۶۲/۸۱ کاهش یافته اســـت. این رفتار نشـــان می‌دهد که در مراحل اولیه انتقال دانش، مدل دانش‌آموز در حال تطبیق تدریجی با توزیع نرم‌شده خروجی‌های مدل معلم است. در این بازه، نقش آموزش نظارت‌شده پیشین

[1] warm-up

به‌عنوان مرحله warm-up اهمیت ویژه‌ای دارد؛ زیرا مدل دانش‌آموز از قبل دارای یک رفتار پایه‌ای پایدار بوده و اکنون صرفاً در حال اصلاح توزیع احتمالات خود در جهت رفتار مدل بزرگ‌تر است، نه یادگیری از صفر.

از گام ۱۰۰ به بعد، روند کاهش KL Divergence شتاب بیشتری می‌گیرد. کاهش مقدار KL از حدود ۶۲/۸۱ در گام ۱۰۰ به حدود ۵۹/۶۲ در گام ۲۰۰ نشان می‌دهد که مدل دانش‌آموز به‌تدریج توانسته است الگوهای احتمالاتی پیچیده‌تری را که در خروجی‌های مدل معلم نهفته‌اند، فراگیرد. این بخش از نمودار بیانگر مرحله‌ای است که مدل دانش‌آموز نه‌تنها ساختار کلی پاسخ‌ها، بلکه اولویت‌بندی نشانه ها و روابط ظریف میان آن‌ها را نیز از مدل معلم اقتباس کرده است.

استفاده از دمای نرم‌سازی T=2.0 نقش مهمی در هموارسازی توزیع خروجی‌های مدل معلم ایفا کرده است. این تنظیم باعث شده است که اطلاعات مربوط به نشانه های با احتمال کمتر نیز در فرآیند انتقال دانش حفظ شوند و مدل دانش‌آموز بتواند از سیگنال آموزشی غنی‌تری نسبت به برچسب‌های سخت[1] بهره ببرد. کاهش پیوسته KL Divergence در این نمودار تأیید می‌کند که این انتخاب پارامتری به همگرایی پایدار کمک کرده است.

نکته مهم آن است که برخلاف برخی روش‌های پیچیده‌تر مبتنی بر یادگیری تقویتی، فرآیند انتقال دانش در این پژوهش بدون استفاده از مدل پاداش یا محیط تقویتی انجام شده است. با این حال، روند یکنواخت و نزولی KL Divergence نشان می‌دهد که انتقال دانش به‌صورت پایدار و بدون نوسانات شدید صورت گرفته است.

در مجموع، این نتایج نشان می‌دهد که مرحله انتقال دانش توانسته است فاصله توزیعی میان مدل دانش‌آموز و مدل معلم را به‌طور معناداری کاهش دهد و بخش قابل توجهی از رفتار پیش‌بینی مدل بزرگ‌تر را به مدل کوچک منتقل کند. این همگرایی توزیعی، پایه‌ای ضروری برای بهره‌گیری مؤثر از دانش بیرونی در مرحله بعدی (RAG) فراهم ساخته و نقش کلیدی در ارتقای نهایی کیفیت تولید کد توسط مدل دانش‌آموز ایفا کرده است.

حال تأثیر افزودن RAG به خط لوله پیشنهادی و مقایسه عملکرد مدل دانش‌آموز در سه حالت مختلف بررسی می‌کنیم.

۱) آموزش نظارت‌شده به‌تنهایی(SFT-only)

۲) آموزش نظارت‌شده به‌همراه انتقال دانش(SFT + KD)

۳) آموزش نظارت‌شده، انتقال دانش و استفاده از RAG (SFT + KD + RAG).

ارزیابی بر اساس سه وظیفه نمونه از مسائل متداول برنامه‌نویسی پایتون انجام شده است که هر یک نماینده سطح متفاوتی از پیچیدگی منطقی هستند. این وظایف به‌صورت زیر تعریف می‌شوند:

- **معیار ۱ (Metric 1):**

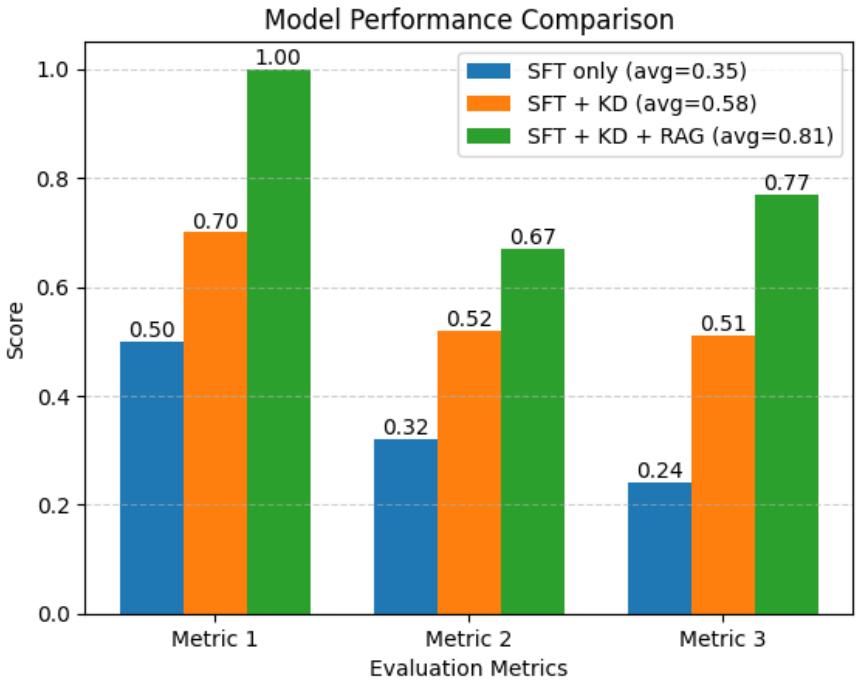
«یک تابع پایتون بنویس که عدد n ام دنباله فیبوناچی را محاسبه کند»

- **معیار ۲ (Metric 2):**

«یک تابع پایتون بنویس که دو عدد را با یکدیگر جمع کند.»

- **معیار ۳ (Metric 3):**
- «یک تابع پایتون به نام is_prime(n: int) -> bool بنویس که اگر n عدد اول است مقدار True و در غیر این صورت False بازگرداند. هیچ کتابخانه‌ای وارد نکن و اعداد $n \leq 1$ را غیراول در نظر بگیر.»
- محور عمودی نمودار، امتیاز عملکرد مدل (ترکیبی از صحت نحوی و صحت منطقی) و محور افقی، وظایف ارزیابی را نشان می‌دهد. تحلیل نتایج قسمتهای مختلف را اینگونه بیان می کنیم:

[1] hard labels



شکل ۳: نمودار امتیاز بر حسب گام معیار سنجش

حالت SFT-only

در این حالت، مدل صرفاً پس از آموزش نظارت‌شده ارزیابی شده است. نتایج نشان می‌دهد که مدل قادر به تولید ساختارهای پایه‌ای کد است، اما عملکرد آن به‌ویژه در مسائل دارای منطق تصمیم‌گیری یا بازگشت (مانند فیبوناچی و تشخیص عدد اول) محدود باقی مانده است. امتیازها در این حالت پایین‌تر بوده و نشان می‌دهند که مدل اغلب تنها اسکلت کلی تابع را تولید می‌کند و در تکمیل منطق مسئله با خطا مواجه می‌شود.

حالت SFT + انتقال دانش(KD)

با افزودن انتقال دانش، بهبود قابل توجهی در تمامی معیارها مشاهده می‌شود. در این حالت، مدل توانسته است رفتارهای پیش‌بینی دقیق‌تر مدل معلم را تا حدی فراگیرد و پاسخ‌هایی منسجم‌تر ارائه دهد. این بهبود به‌ویژه در معیار ۱ (فیبوناچی) و معیار ۳ (تشخیص عدد اول) مشهود است، زیرا این وظایف نیازمند درک روابط منطقی و شرطی هستند. با این حال، همچنان در برخی موارد جزئیات پیاده‌سازی ناقص باقی می‌ماند.

حالت SFT + انتقال دانش+ RAG (مدل نهایی)

افزودن RAG منجر به بیشترین جهش عملکردی در هر سه معیار شده است. در این حالت، مدل به نمونه‌های مرتبط از پایگاه دانش بیرونی دسترسی دارد و می‌تواند از الگوهای معتبر کدنویسی برای تکمیل پاسخ خود استفاده کند. در معیار ۱، مدل نهایی به امتیاز کامل نزدیک شده است که نشان‌دهنده توانایی تولید یک پیاده‌سازی صحیح و کامل از تابع فیبوناچی است. در معیار ۲، که یک وظیفه ساده‌تر است، اگرچه تفاوت میان حالت‌ها کمتر است، اما همچنان نسخه مجهز به RAG پاسخ‌هایی دقیق‌تر و بدون خطای نحوی ارائه می‌دهد. در معیار ۳، که شامل منطق شرطی و بررسی حالات مرزی است، استفاده از RAG موجب افزایش چشمگیر صحت منطقی شده و مدل توانسته است تمامی شرایط مسئله را به‌درستی رعایت کند.

از این نتایج می توان گفت که آموزش نظارت‌شده، پایه لازم برای تولید کد را فراهم می‌کند و انتقال دانش، رفتار مدل کوچک را به مدل بزرگ نزدیک‌تر می‌سازد و در نهایت، RAG نقش مکمل کلیدی در جبران محدودیت ظرفیت پارامتری مدل دانش‌آموز ایفا می‌کند.

افزودن RAG باعث شده است که مدل بدون افزایش تعداد پارامترها، به دانشی فراتر از حافظه داخلی خود دسترسی داشته باشد و در نتیجه، هم از نظر صحت نحوی و هم از نظر کامل‌بودن منطقی، عملکردی به‌مراتب بهتر از حالت‌های بدون RAG ارائه دهد. این نتایج به‌روشنی برتری چارچوب پیشنهادی را نسبت به تنظیم ساده مدل‌های کوچک یا حتی انتقال دانش به‌تنهایی نشان می‌دهد.

در خط لوله‌های چندمرحله‌ای، به‌ویژه در چارچوب‌هایی که شامل آموزش نظارت‌شده، انتقال دانش و بازیابی همراه با تولید هستند، این احتمال وجود دارد که خطاهای ایجادشده در مراحل اولیه به مراحل بعدی منتقل شده و به‌صورت تجمعی بر خروجی نهایی اثر بگذارند. این پدیده که به‌عنوان انباشت خطا شناخته می‌شود، می‌تواند باعث کاهش پایداری و قابلیت اعتماد مدل در صورت طراحی نامناسب خط لوله گردد.

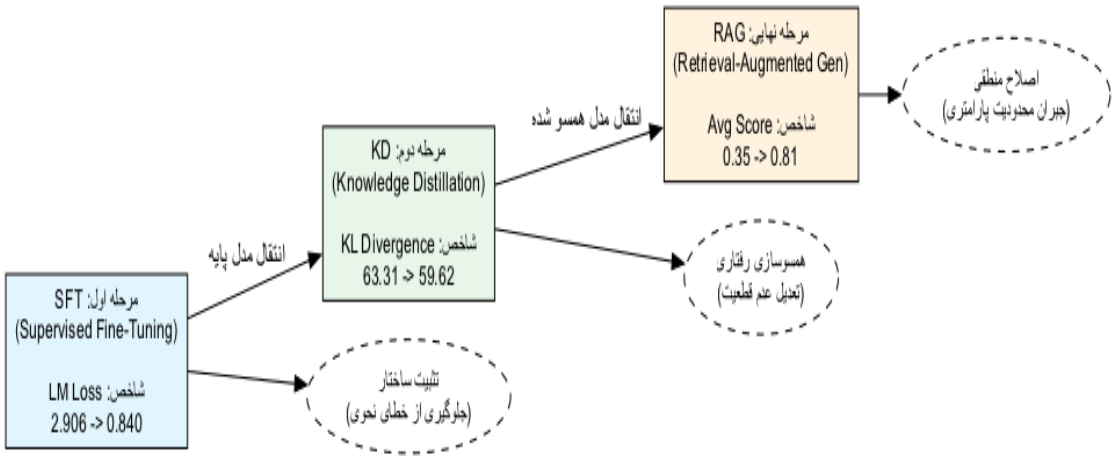
به‌منظور بررسی این موضوع، رفتار مدل دانش‌آموز در هر یک از سناریوهای ارزیابی به‌صورت مستقل تحلیل شد تا مشخص شود آیا خطاهای مراحل قبلی به‌صورت تقویتی به مرحله نهایی منتقل شده‌اند یا خیر.

در تحلیل پدیده انباشت خطا در خط لوله پیشنهادی، اثرات متقابل سه مرحله آموزش نظارت شده (SFT) ، انتقال دانش (KD) و بازیابی (RAG) بر کیفیت خروجی نهایی مورد واکاوی قرار گرفت .برخلاف سیستم‌های خطی سنتی که در آن‌ها خطای مراحل اولیه به صورت تصاعدی در مراحل بعد تقویت می‌شود، در معماری حاضر مشاهده گردید که هر مرحله به عنوان یک فیلتر اصلاح‌کننده برای مرحله قبل عمل کرده است.

دلایل عدم انباشت خطا و بهبود تدریجی عملکرد را می‌توان در موارد زیر خلاصه کرد:

- **کنترل خطای ساختاری در مرحله SFT**: در ابتدای فرآیند، مقدار **LM Loss** از **۲/۹۰۶** به **۰/۸۴۰** کاهش یافت . این کاهش نشان‌دهنده آن است که مدل دانش‌آموز پیش از ورود به مراحل پیچیده‌تر، توانسته است خطاهای نحوی[1] را به حداقل برساند و به یک پایداری ساختاری دست یابد.
- **تعدیل توزیعی در مرحله انتقال دانش:** با اجرای مرحله KD، فاصله توزیعی بین مدل معلم و دانش‌آموز (معیار **KL Divergence**) از **۶۳/۳۱** به **۵۹/۶۲** تقلیل یافت .این روند نزولی بیانگر آن است که مدل به جای تقویت خطاهای احتمالی مرحله SFT، رفتار خود را با توزیع بهینه مدل بزرگتر همسو کرده و «عدم قطعیت» در تولید توکن‌ها را کاهش داده است.
- **خنثی‌سازی خطای پارامتری توسط RAG**: در مرحله نهایی، محدودیت‌های حافظه پارامتری که عامل اصلی خطاهای منطقی در مدل‌های کوچک هستند، با تزریق دانش بیرونی جبران شد .نتایج ارزیابی نشان می‌دهد که امتیاز عملکرد مدل از **۰/۳۵** در حالت پایه به **۰/۸۱** در حالت نهایی ارتقا یافته است .این جهش عملکردی اثبات می‌کند که مکانیزم RAG نه تنها خطاهای مراحل قبل را انباشته نکرده، بلکه در بسیاری از موارد به عنوان یک «اصلاح‌گر خودکار» برای پاسخ‌های ناقص عمل کرده است.

[1] Syntax Errors



شکل ۴: فلوچارت بررسی انباشت خطا در خط لوله چند مرحله ای

همان‌طور که در تصویر مشاهده می‌شود، هر مرحله از خط لوله نقش متمایزی در جلوگیری از انباشت خطا ایفا می‌کند .در گام نخست (SFT) کاهش مقدار LM Loss از ۲/۹۰۶ به ۰/۸۴۰ منجر به «تثبیت ساختار» و رفع خطاهای نحوی شده است .در گام دوم (KD)، فرآیند انتقال دانش با کاهش فاصله توزیعی (KL Divergence) از ۶۳/۳۱ به ۵۹/۶۲، باعث «همسوسازی رفتاری» و تعدیل عدم قطعیت در مدل گردیده است .در نهایت، مرحله RAG با جبران محدودیت حافظه پارامتری، میانگین امتیاز عملکرد را از ۰/۳۵ به ۰/۸۱ ارتقا داده که نشان‌دهنده «اصلاح خطاهای منطقی» و تکمیل دانش مدل بدون بازگشت خطاست . بیضی‌های خط‌چین مکانیزم اثرگذاری هر مرحله بر ارتقای کیفیت خروجی را نشان می‌دهند.

در مجموع، طراحی مرحله‌ای خط لوله به گونه‌ای انجام شده است که مرحله SFT «صحت نحوی»، مرحله KD «نزدیکی رفتاری» و مرحله RAG «صحت معنایی و منطقی» را تضمین می‌نماید .این ترکیب هوشمندانه موجب شده است که نرخ خطا در هر مرحله توسط مؤلفه بعدی مهار گردد.

۵- نتیجه گیری

در این پژوهش، یک خط لوله چندمرحله‌ای متشکل از آموزش نظارت‌شده، انتقال دانش و بازیابی همراه با تولید با هدف ارتقای عملکرد یک مدل زبانی کوچک در تولید کد، طراحی و ارزیابی شد. نتایج تجربی نشان داد که هر یک از این مراحل، نقشی متمایز و مکمل در بهبود عملکرد نهایی مدل دانش‌آموز CodeGen-350M ایفا کرده‌اند.

در مرحله آموزش نظارت‌شده، مدل توانست الگوهای پایه‌ای نگاشت دستور متنی به کد را فراگیرد؛ با این حال، عملکرد آن در وظایف دارای منطق تصمیم‌گیری یا ساختارهای بازگشتی محدود باقی ماند. ارزیابی مدل در این حالت نشان داد که میانگین امتیاز عملکرد در سه مسئله نمونه برابر با ۰/۳۵بوده است. این نتیجه نشان می‌دهد که SFT به‌تنهایی برای ایجاد یک رفتار پایه‌ای ضروری است، اما برای حل کامل مسائل کدنویسی کافی نیست.

با افزودن مرحله انتقال دانش از مدل معلم CodeGen-6B، بهبود معناداری در عملکرد مدل دانش‌آموز مشاهده شد. کاهش یکنواخت مقدار KL Divergence نشان داد که مدل دانش‌آموز توانسته است توزیع احتمالات خروجی خود را به رفتار مدل بزرگ‌تر نزدیک کند. این همگرایی توزیعی منجر به افزایش میانگین امتیاز عملکرد مدل به ۰/۵۸شد. در این حالت، مدل در وظایفی مانند محاسبه عدد فیبوناچی و تشخیص عدد اول، پاسخ‌هایی منسجم‌تر نسبت به حالت SFT-only ارائه داد، هرچند همچنان در برخی موارد جزئیات منطقی به‌طور کامل پوشش داده نمی‌شد.

بیشترین بهبود عملکرد با افزودن مکانیزم بازیابی همراه با تولید حاصل شد. اتصال مدل دانش‌آموز به یک پایگاه دانش بیرونی مبتنی بر ایندکس FAISS و استفاده از نمونه‌های بازیابی‌شده در قالب پرامپت، موجب شد مدل بتواند از دانش بیرونی برای تکمیل

پاسخ‌های خود بهره بگیرد. در این حالت، میانگین امتیاز عملکرد مدل به ۰/۸۱ افزایش یافت که نشان‌دهنده جهش قابل توجه نسبت به دو سناریوی قبلی است.

تحلیل جزئی‌تر نتایج نشان می‌دهد که در مسئله محاسبه عددn ام دنباله فیبوناچی، امتیاز مدل از ۰/۵۰ در حالت SFT- only به ۰/۷۰ پس از انتقال دانش و نهایتاً به ۱/۰۰ در نسخه مجهز به RAG افزایش یافته است؛ در مسئله جمع دو عدد، که یک وظیفه ساده‌تر محسوب می‌شود، امتیاز از۰/۳۲ به ۰/۵۲ و سپس به ۰/۶۷ رسیده است. در مسئله تشخیص عدد اول، که نیازمند منطق شرطی و بررسی حالات مرزی است، عملکرد مدل از ۰/۲۴ در حالت پایه به ۰/۵۱ پس از انتقال دانش و در نهایت به ۰/۷۷ با استفاده از RAG بهبود یافته است.

این نتایج نشان می‌دهد که RAG نه‌تنها باعث افزایش صحت نحوی کدهای تولیدشده شده، بلکه نقش مهمی در تکمیل منطق مسئله و پوشش شرایط مرزی ایفا کرده است. در بسیاری از موارد، مدل بدون RAG تنها ساختار کلی تابع را تولید می‌کرد، در حالی که نسخه مجهز به RAG توانسته است با اتکا به نمونه‌های معتبر بازیابی‌شده، پیاده‌سازی‌های کامل‌تر و دقیق‌تری ارائه دهد.

نکته حائز اهمیت آن است که مدل دانش‌آموز با تنها ۳۵۰ میلیون پارامتر، با بهره‌گیری از انتقال دانش و RAG، به عملکردی دست یافته است که از نظر کیفی و کمی به رفتار مدل معلم ۶ میلیارد پارامتری نزدیک می‌شود. این یافته نشان می‌دهد که افزودن ماژول بازیابی می‌تواند بخش قابل توجهی از محدودیت ناشی از کوچک‌بودن مدل را جبران کند، بدون آنکه نیازی به افزایش تعداد پارامترها یا استفاده از روش‌های پرهزینه‌ای مانند یادگیری تقویتی وجود داشته باشد.

در مجموع، نتایج این پژوهش نشان می‌دهد که ترکیب SFT به‌عنوان مرحله warm-up، انتقال دانش برای همگرایی توزیعی و RAG برای تکمیل دانش بیرونی یک مسیر عملی، سبک و قابل تکرار برای ارتقای مدل‌های زبانی کوچک در وظایف تولید کد فراهم می‌کند. اهمیت این دستاورد در آن است که نشان می‌دهد حتی با منابع محاسباتی محدود نیز می‌توان به مدلی دست یافت که از نظر دقت و کارایی، با مدل‌های بسیار بزرگ‌تر قابل رقابت باشد.

مراجع

[1] L. Ouyang *et al*., “Training language models to follow instructions with human feedback,” *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 35, pp. 27730–27744, 2022.

[2] Y. Wang, Y. Kordi, S. Mishra, P. Liu, N. A. Smith, and A. Ettinger, “Self-Instruct: Aligning language models with instructions without human labels,” *arXiv preprint*, arXiv:2212.10560, 2022.

[3] R. Taori *et al*., “Stanford Alpaca: An instruction-following LLaMA model,” Stanford Center for Research on Foundation Models, Tech. Rep., 2023.

[4] T. Wolf *et al*., “Transformers: State-of-the-art natural language processing,” in *Proc. Conf. Empirical Methods in Natural Language Processing (EMNLP): System Demonstrations*, 2020, pp. 38–45.

[5] P. Lewis *et al*., “Retrieval-augmented generation for knowledge-intensive NLP tasks,” *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 33, pp. 9459–9474, 2020.

[6] E. Nijkamp *et al*., “CodeGen: An open large language model for code with multi-turn program synthesis,” *arXiv preprint*, arXiv:2203.13474, 2022.

[7] C. Raffel *et al*., “Exploring the limits of transfer learning with a unified text-to-text transformer,” *Journal of Machine Learning Research*, vol. 21, no. 140, pp. 1–67, 2020.

[8] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, “BERT: Pre-training of deep bidirectional transformers for language understanding,” in *Proc. North American Chapter of the Association for Computational Linguistics (NAACL)*, 2019, pp. 4171–4186.

[9] J. Austin *et al*., “Program synthesis with large language models,” *arXiv preprint*, arXiv:2108.07732, 2021.

[10] J. Johnson, M. Douze, and H. Jégou, “Billion-scale similarity search with GPUs,” *IEEE Transactions on Big Data*, vol. 7, no. 3, pp. 535–547, 2019.

[11] M. Chen *et al*., “Evaluating large language models trained on code,” *arXiv preprint*, arXiv:2107.03374, 2021.

[12] J. Dodge, S. Gururangan, D. Card, R. Schwartz, and N. A. Smith, “Show your work: Improved reporting of experimental results,” in *Proc. Empirical Methods in Natural Language Processing (EMNLP)*, 2019.

[13] A. Radford *et al*., “Language models are unsupervised multitask learners,” OpenAI, Tech. Rep., 2019.

[14] I. Loshchilov and F. Hutter, “Decoupled weight decay regularization,” in *Proc. Int. Conf. on Learning Representations (ICLR)*, 2019.

[15] Y. You *et al*., “Large batch optimization for deep learning: Training BERT in 76 minutes,” in *Proc. Int. Conf. on Learning Representations (ICLR)*, 2020.