دانشگاه صنعتی امیرکبیر
(پلی تکنیک تهران)

دانشکده مهندسی کامپیوتر
گرایش هوش مصنوعی

گزارش پروژه پایانی درس پردازش زبان طبیعی پیشرفته

استاد درس: دکتر ممتازی

سیددانیال قریشی‌مدینه
۴۰۳۱۳۱۰۶۱
dghoreyshi@aut.ac.ir

اسفند ۱۴۰۴

فهرست مطالب

۱ مقدمه و تعریف مسئله

۱.۱ معرفی تسک استنتاج رویدادها (Abductive Event Reasoning - SemEval-2026 Task 12)

مسئله استنتاج رویدادها یا Abductive Event Reasoning، یکی از چالش‌برانگیزترین مباحث در حوزه پردازش زبان طبیعی (NLP) است. در مسابقه SemEval-2026 Task 12، هدف طراحی سیستمی است که بتواند با دریافت یک رویداد هدف (Target Event) و مجموعه‌ای از اسناد متنی به عنوان بستر دانشی (Context)، محتمل‌ترین علت یا علت‌های مستقیم آن رویداد را از میان چهار گزینه ارائه‌شده شناسایی کند. یکی از پیچیدگی‌های اصلی این تسک، امکان وجود چند علت هم‌زمان برای یک رویداد است؛ به این معنا که مدل باید بتواند ترکیبی از گزینه‌های صحیح را استخراج نماید و صرفاً به یافتن یک پاسخ واحد بسنده نکند.

۲.۱ اهمیت استنتاج علت و معلولی در مدل‌های زبانی بزرگ (LLMs)

مدل‌های زبانی بزرگ (Large Language Models یا LLMs) در سال‌های اخیر پیشرفت‌های چشمگیری در تولید متن و درک زبان طبیعی داشته‌اند. با این حال، توانایی این مدل‌ها در انجام استدلال‌های پیچیده منطقی و کشف روابط علت و معلولی (Causal Inference) همچنان با محدودیت‌هایی روبه‌رو است. استنتاج ابداکتیو (Abductive Reasoning)، یعنی رسیدن به محتمل‌ترین توضیح برای یک مشاهده ناقص، قابلیتی است که برای درک رویدادهای دنیای واقعی، کاهش توهم (Hallucination) و افزایش اتکاپذیری سیستم‌های هوش مصنوعی حیاتی است. ارتقای این توانمندی باعث می‌شود مدل‌ها از سطح تطابق الگو (Pattern Matching) فراتر رفته و به درک معنایی و منطقی عمیق‌تری دست یابند.

۳.۱ اهداف پروژه و مرور کلی رویکرد پیشنهادی

هدف اصلی این پروژه، توسعه یک سیستم هوشمند و کارآمد برای حل تسک شماره ۱۲ از مسابقه SemEval-2026 است. برای دستیابی به این هدف و غلبه بر چالش‌های استنتاجی، یک رویکرد چندمرحله‌ای و پیشرفته اتخاذ شده است.

در گام نخست، از تکنیک بازیابی اطلاعات معنایی (Semantic RAG) برای فیلتر کردن متون طولانی و استخراج مرتبط‌ترین جملات استفاده شد. سپس عملکرد مدل پایه یعنی Qwen2.5-7B-Instruct در حالت Zero-Shot CoT مورد ارزیابی قرار گرفت که به دقت اولیه ۶۱% منجر گردید.

در گام بعدی، برای رفع ضعف‌های استدلالی مدل در مواجهه با سوالات پیچیده، استراتژی غنی‌سازی داده‌ها (Data Augmentation) مبتنی بر زنجیره افکار (Chain-of-Thought) به کار گرفته شد. در نهایت، با استفاده از معماری RAFT (Retrieval-Augmented Fine-Tuning) و تکنیک بهینه‌سازی QLoRA، مدل بر روی زیرمجموعه‌ای باکیفیت از داده‌های غنی‌شده، آموزش داده شد (Fine-Tuning). نتایج ارزیابی نشان داد که این رویکرد یکپارچه توانسته است دقت سیستم را با جهشی چشمگیر به ۸۱% ارتقا دهد که نشان‌دهنده اثربخشی بالای ترکیب استدلال منطقی و تنظیم دقیق مدل است.

۲ آزمایش‌های اولیه و ارزیابی رویکردهای مبتنی بر پرامپت (Prompting)

در فاز نخست این پروژه، پیش از ورود به مباحث پیچیده آموزش مدل، تلاش شد تا با استفاده از تکنیک‌های پیشرفته مهندسی پرامپت (Prompt Engineering)، بیشترین بازدهی ممکن از مدل‌های از پیش‌آموزش‌دیده استخراج شود. در ادامه، مسیر تکامل این آزمایش‌ها و نتایج هر یک شرح داده شده است.

۱.۲ معرفی مدل Qwen2.5-7B-Instruct به عنوان مدل پایه

برای انجام این پژوهش، مدل Qwen2.5-7B-Instruct به عنوان مدل پایه (Baseline) انتخاب گردید. دلیل این انتخاب، توازن فوق‌العاده‌ی این مدل میان اندازه (۷ میلیارد پارامتر) و قدرت استدلال بود. این مدل متن‌باز (Open-Source)، بهینه‌سازی بسیار خوبی برای پیروی از دستورالعمل‌ها (Instruction Following) دارد و به دلیل حجم مناسب، امکان استقرار و اجرای آزمایش‌ها بر روی سخت‌افزارهای محدود (مانند پردازنده‌های گرافیکی T4 در محیط Kaggle) را بدون افت شدید کیفیت فراهم می‌کند.

۲.۲ پیاده‌سازی مدل پایه با ترکیب Semantic RAG و Zero-Shot CoT

یکی از چالش‌های اصلی دادگان مسابقه، وجود متون پس‌زمینه (Context) بسیار طولانی و حاوی اطلاعات نویز بود که می‌توانست تمرکز مدل را از یافتن علت اصلی منحرف سازد. از سوی دیگر، مدل‌های زبانی خام در استنتاج‌های یک‌مرحله‌ای و بدون تفکر قبلی، دچار خطای محاسباتی می‌شوند. برای حل این دو چالش به صورت هم‌زمان، معماری خط پایه (Baseline) سیستم با ترکیب تکنیک‌های Semantic RAG و Zero-Shot CoT طراحی و پیاده‌سازی گردید.

گام اول - فیلتر کانتکست با Semantic RAG: برای استخراج شواهد کلیدی از متن، از مدل تعبیه‌سازی (Embedding) بسیار سریع و کارآمد all-MiniLM-L6-v2 استفاده شد. در پیاده‌سازی، ابتدا متن طولانیِ پس‌زمینه با استفاده از عبارات باقاعده (Regular Expressions) به جملات مجزا شکسته شد و جملات بسیار کوتاه (کمتر از ۲۰ کاراکتر) به عنوان نویز حذف گردیدند. سپس، یک کوئری جامع شامل «رویداد هدف» (Target Event) به علاوه متن هر چهار گزینه ساخته شد. پس از محاسبه شباهت کسینوسی (Cosine Similarity) میان بردار کوئری و بردارهای جملات متن، تعداد $K = 5$ جمله که بیشترین شباهت معنایی را داشتند استخراج شدند. نکته کلیدی در این پیاده‌سازی، مرتب‌سازی مجدد شاخص‌های جملاتِ استخراج‌شده (Index Sorting) بود تا ترتیب زمانی و منطقی جملاتِ اصلیِ متن به هم نریزد و کانتکستِ فیلترشده کاملاً خوانا باقی بماند.

گام دوم - تولید استدلال با Zero-Shot CoT: جملات فیلترشده در گام قبل، به عنوان کانتکست جدید وارد پرامپت مدل Qwen2.5-7B-Instruct شدند. در پرامپت سیستم (System Prompt)، با استفاده از تکنیک زنجیره افکار (Chain-of-Thought)، از مدل خواسته شد که در حالت Zero-Shot (بدون دیدن نمونه حل‌شده) ابتدا فرآیند استدلال خود را به صورت گام‌به‌گام و مختصر بنویسد و سپس خروجی نهایی را صریحاً در داخل تگ‌های XML (مانند <ANSWER>A,C</ANSWER>) قرار دهد. الزام مدل به استدلال کردن پیش از پاسخ نهایی باعث تولید توکن‌های میانی (Intermediate Tokens) می‌شود که به طور موثری فضای محاسباتی مدل را برای حل مسائل پیچیده منطقی افزایش می‌دهد. همچنین در تنظیمات تولید متن (Generation Parameters)، دمای تولید (Temperature) روی مقدار پایین ۰.۱ تنظیم شد تا خروجی‌های مدل کاملاً قطعی، منطقی و به دور از توهم (Hallucination) باشد.

ارزیابی و نتایج: اعمال این معماری بر روی دادگان ارزیابی شامل ۲۰۰ نمونه، منجر به ثبت دقت اولیه **۶۱%** (پاسخ‌گویی صحیح به ۱۲۲ سوال) گردید. این نتیجه به عنوان یک خط پایه، عملکردی قابل قبول و منسجم را نشان داد؛ اما در عین حال ثابت کرد که مدل خام، حتی با داشتن کانتکست تمیز و فرصت استدلال، همچنان در درک کامل منطق استنتاج ابداکتیو (Abductive Reasoning) دارای سقف توانایی مشخصی است و نیاز به ارتقای رویکرد دارد.

به منظور شفافیت کامل و امکان بازتولید نتایج (Reproducibility)، ساختار دقیق پرامپت سیستم (System Prompt) که برای تنظیم نقش مدل و هدایت آن به سمت استدلال گام‌به‌گام طراحی شد، در قطعه زیر آورده شده است. این پرامپت مدل را ملزم می‌کند تا ابتدا استدلال خود را بیان کرده و سپس پاسخ نهایی را در قالب تگ‌های XML تولید کند:

```
You are an expert AI specialized in Abductive Event Reasoning. First,
   briefly reason about the direct cause. Then, you MUST output the
   uppercase letter(s) of the correct option(s) inside XML tags like this:
   <ANSWER>A</ANSWER> or <ANSWER>A,C</ANSWER>. Do not output anything else
   after the tags.
```

۳.۲ ارتقای پرامپت‌ها: پیاده‌سازی Semantic RAG در ترکیب با Few-Shot CoT

برای بهبود عملکرد سیستم و تلاش برای عبور از سقف دقت مدل پایه، آزمایش دوم با استفاده از رویکرد یادگیری در بافت (In-Context Learning) طراحی گردید. در این مرحله، تکنیک Few-Shot CoT به کار گرفته شد تا با ارائه چند نمونه حل‌شده به عنوان الگو در پرامپت سیستم، مدل با ساختار استنتاج ابداکتیو (Abductive Reasoning) و فرمت مورد انتظار پاسخ‌ها آشناتر شود.

جزئیات پیاده‌سازی مهندسی: در این آزمایش، دو ارتقای اساسی در معماری کدها اعمال شد:

- **ارتقای مدل بازیابی اطلاعات (Retriever):** برای افزایش کیفیت و دقت جملات استخراج‌شده از متن اصلی، مدل تعبیه‌سازی (Embedding) سیستم از نسخه all-MiniLM-L6-v2 به مدل قدرتمندتر و بزرگ‌ترِ BAAI/bge-large-en-v1.5 ارتقا یافت. همچنین، برای اطمینان از اینکه هیچ شواهد علت و معلولی از قلم نیفتد، پارامتر تعداد جملات استخراجی به $K = 6$ (top_k=6) افزایش پیدا کرد.
- **تزریق نمونه‌های آموزشی (Demonstrations):** در پرامپت سیستم (System Prompt)، دو نمونه ساخته‌شده‌ی دستی (یک نمونه با پاسخ واحد و یک نمونه با پاسخ چندگانه) تعبیه شد. هر نمونه شامل کانتکست فرضی، رویداد هدف، گزینه‌ها، یک خط استدلال منطقی و در نهایت پاسخ محصور در تگ‌های XML بود.

پارامترهای تولید متن (Generation Config) نیز برای حفظ ثبات و قطعیتِ استدلال، روی دمای بسیار پایین (temperature=0.1) تنظیم گردید. کد مربوط به تزریق نمونه‌ها در پرامپتِ سیستم به شکل زیر پیاده‌سازی شد:

```
few_shot_examples = """
Example 1:
Context: The building's structural integrity was compromised after the
    explosion. Investigators found traces of C4.
Target Event: The roof collapsed.
Options: A) Heavy rain started. B) An explosive device detonated. C) The
    building was old. D) Firefighters arrived.
Reasoning: The context mentions an explosion and C4, which is an explosive.
     Detonation of such a device is the direct cause of the collapse.
<ANSWER>B</ANSWER>

Example 2:
...
<ANSWER>A,C</ANSWER>
"""
```

ارزیابی و تحلیل غیرمنتظره‌ی نتایج: با وجود اینکه انتظار می‌رفت تکنیک Few-Shot CoT به ساختاریافتگی بهتر و هدایت مدل در مسیر صحیح کمک کند، ارزیابی این معماری روی دادگان ۲۰۰ نمونه‌ای، نتیجه‌ای کاملاً برعکس به همراه داشت. دقت کل مدل با افت محسوس روبه‌رو شد و به ۵۵.۷۵% (مجموع امتیازات ۱۱۱.۵) تنزل یافت که حدود ۵ درصد کمتر از حالت Zero-Shot بود.

تحلیل علمی دلایل افت دقت (Error Analysis): بررسی دقیق لاگ‌ها و خطایابی خروجی‌های مدل نشان داد که این افت عملکرد ریشه در دو چالش شناخته‌شده در حوزه Prompt Engineering برای مدل‌های بزرگ دارد:

۱. **سرریز اطلاعات و گم‌شدگی در میانه (Lost in the Middle):** اضافه شدن دو نمونه طولانی و کامل به پرامپت سیستم، هم‌زمان با افزایش طول کانتکستِ خود سوال (top_k=6)، باعث افزایش چشمگیر طول پرامپت ورودی (Context Length) گردید. در این شرایط، مدل در اختصاص وزن توجه (Attention Mechanism) دچار مشکل شده و تمرکز خود را بر روی شواهد کلیدی موجود در انتهای متن از دست می‌دهد.

۲. **اثر سوگیریِ نمونه‌ها (Demonstration Bias):** مدل‌هایِ به شدت دستورپذیر (Highly Instruction-Tuned) مانند Qwen2.5-Instruct، گاهی اوقات بیش از حد به ساختارِ نمونه‌های ارائه شده وابسته می‌شوند. از آنجا که دو نمونه‌ی Few-Shot به صورت ثابت (Static) برای تمامی ۲۰۰ سوال ارائه شده بودند و ممکن

بود از نظر دامنه مفهومی با سوالات تفاوت داشته باشند، مدل به جای استدلالِ آزادانه بر اساس کانتکست جدید، تلاش می‌کرد تا کورکورانه از قالب استدلالِ نمونه‌های ثابت تقلید کند. این تقلید اجباری باعث افزایش پاسخ‌های کاملاً غلط یا از دست رفتن تطابق‌های جزئی (Partial Matches) گردید.

این یافته‌های تجربی ارزشمند ثابت کرد که در تسک پیچیده‌ای مانند استنتاج علت و معلولی، دادن فضای تفکر آزادتر به مدل (Zero-Shot CoT) اثربخشی بیشتری نسبت به تحمیل یک چارچوب استدلالیِ محدود و از پیش‌تعیین‌شده (Few-Shot CoT) دارد.

۴.۲ رویکرد پیشرفته: ترکیب Semantic RAG، تکنیک Few-Shot CoT و Self-Consistency

در گام بعدی از آزمایش‌های مهندسی پرامپت، برای جبران افت دقت مرحله قبل، تکنیک خودسازگاری (Self-Consistency) به معماری سیستم افزوده شد. در رویکردهای استانداَرد، مدل تنها یک مسیر استدلالی را تولید می‌کند (رمزگشایی حریصانه یا Greedy Decoding) که ممکن است مستعد خطای تصادفی باشد؛ اما در Self-Consistency، از مدل خواسته می‌شود برای یک سوال یکسان، چندین مسیر استدلالی و پاسخِ متفاوت تولید کند و سپس پاسخ نهایی از طریق رای‌گیری اکثریت (Majority Voting) میان خروجی‌ها انتخاب می‌شود.

جزئیات پیاده‌سازی مهندسی: در این آزمایش، معماری پایه همان معماری قبلی (مدل تعبیه‌سازی BGE-Large با $K = 6$ و پرامپت حاوی ۲ نمونه Few-Shot) بود، اما منطق استنتاج (Inference Logic) دستخوش تغییرات اساسی شد:

- **تنوع در تولید (Generation Diversity):** دمای تولید (Temperature) از مقدار ۰.۱ به ۰.۴ افزایش یافت. این تغییرِ پارامتر بسیار حیاتی است، زیرا در دماهای پایین، مدل خروجی‌های کاملاً یکسانی تولید می‌کند و سیستم رای‌گیری بی‌معنی می‌شود. دمای ۰.۴ به مدل اجازه داد تا در هر بار اجرای سوال، مسیر استدلالیِ متفاوتی را طی کند.
- **تعداد مسیرهای استدلالی (N):** به دلیل محدودیت‌های پردازشی محیط Kaggle، تعداد تکرار برای هر سوال روی $N = 3$ تنظیم شد.
- **منطق رای‌گیری اکثریت (Majority Voting):** خروجی مدل پس از ۳ بار اجرا، پردازش شده و گزینه‌های استخراج‌شده در یک لیست (votes) ذخیره گردیدند. در نهایت با استفاده از تابع Counter در پایتون، پرتکرارترین پاسخ به عنوان پیش‌بینی نهایی انتخاب شد.

ارزیابی و تحلیل نتایج: اجرای این آزمایش بر روی دادگان ۲۰۰ نمونه‌ای با دقت ۵۶.۵۰% همراه بود.

تحلیل این نتیجه نشان می‌دهد که اگرچه تکنیک Self-Consistency موفق شد با حذف برخی توهماتِ تصادفی (Hallucinations)، دقت را نسبت به حالت Few-Shot خام اندکی بهبود بخشد، اما همچنان با خط پایه Zero-Shot (۶۱%) فاصله معناداری داشت. دلیل علمی این پدیده آن است که Self-Consistency صرفاً باعث افزایش پایداریِ (Robustness) منطق فعلیِ مدل می‌شود؛ زمانی که مدل به دلیل طولانی شدن پرامپت (Lost in the Middle) و سوگیریِ نمونه‌ها (Demonstration Bias) اساساً در مسیر غلطی قرار گرفته باشد، تکرار ۳ باره‌ی آن فرآیند، تنها به تثبیتِ همان پاسخ‌های غلطِ ساختاریافته کمک می‌کند.

۵.۲ جمع‌بندی آزمایش‌های اولیه، تحلیل محدودیت‌های مدل خام و ضرورت ورود به فاز فاین‌تیونینگ

مجموعه آزمایش‌های انجام‌شده در این فاز که از ساده‌ترین پرامپت‌ها تا پیشرفته‌ترین معماری‌های مبتنی بر RAG و Self-Consistency را شامل می‌شد، به وضوح نشان داد که مهندسی پرامپت (Prompt Engineering) به تنهایی نمی‌تواند چالش‌های استنتاج ابداکتیو را در این تسک برطرف سازد.

مدل خام Qwen2.5-7B دارای یک سقف توانایی (Performance Ceiling) است. تلاش برای هدایت مدل با نمونه‌های آموزشیِ درون‌متنی (Few-Shot) به جای بهبود عملکرد، باعث سرریز اطلاعات و افت شدید دقت گردید و استفاده از تکنیک‌های جبرانی نظیر Self-Consistency نیز تنها توانست کمتر از ۱ درصد این افت را جبران کند، در حالی که هزینه محاسباتی (Computational Cost) و زمان استنتاج را به ۳ برابر افزایش داد.

بر اساس این مشاهدات تجربیِ مستدل، نتیجه‌گیری شد که برای دستیابی به یک جهش معنادار در دقت، استنتاج علت و معلولی نباید صرفاً به عنوان یک دستور (Instruction) به مدل تحمیل شود، بلکه این مهارت باید به صورت عمیق در وزن‌های شبکه عصبی (Neural Weights) درونی‌سازی گردد. این جمع‌بندی، ضرورت تغییر پارادایم از رویکردهای بدون‌آموزش (Training-free) و ورود قطعی به فاز تنظیم دقیق مدل با استفاده از معماری پیشرفته RAFT (Retrieval-Augmented Fine-Tuning) را به اثبات رساند.

۳ استراتژی غنی‌سازی داده‌ها (Data Augmentation)

همان‌طور که در نتایج آزمایش‌های اولیه مشاهده شد، مدل‌های زبانی در حالت خام فاقد توانایی درونی‌سازی‌شده برای حل مسائل استنتاج ابداکتیو در حوزه‌ی متون پیچیده هستند. از آنجا که دادگان مسابقه تنها شامل متن سوال و پاسخ نهایی (حروف گزینه‌ها) بود، فاین‌تیونینگ (Fine-Tuning) مستقیم مدل روی این داده‌ها منجر به یادگیری طوطی‌وار و کاهش قدرت تعمیم (Generalization) می‌شد. برای حل این مشکل، نیازمند دادگانی بودیم که علاوه بر پاسخ، شامل مسیر استدلال منطقی (Chain-of-Thought) نیز باشد. از این رو، استراتژی غنی‌سازی داده‌ها (Data Augmentation) برای تولید خودکار این استدلال‌ها طراحی و اجرا گردید.

۱.۳ تولید استدلال‌های منطقی (Chain-of-Thought) با استفاده از مدل‌های Llama-3

برای تولید استدلال‌های باکیفیت به عنوان دادگان آموزشی (Training Data)، از پلتفرم Groq که سخت‌افزارهای پردازش تنسور (LPU) فوق‌سریع را ارائه می‌دهد، استفاده شد. به عنوان مدل‌های معلم (Teacher Models)، از مدل‌های قدرتمند Llama-3.3-70B (برای استدلال‌های پیچیده) و Llama-3.1-8B-Instant (برای مدیریت سرعت و محدودیت‌ها) بهره گرفتیم. فرآیند تولید بدین شکل بود که برای هر نمونه از دادگان آموزشی، ابتدا کانتکست مرتبط توسط ماژول Semantic RAG استخراج شده و سپس به همراه رویداد هدف و متن دقیقِ گزینه‌ی صحیح، در اختیار مدل معلم قرار می‌گرفت تا یک استدلال منطقی و کوتاه (۲ الی ۳ جمله) برای علتِ وقوعِ آن رویداد تولید کند. این استدلال‌ها به عنوان دانشِ انتقالی (Knowledge Distillation) برای آموزش مدلِ نهایی مورد استفاده قرار گرفتند.

۲.۳ چالش یادگیری میان‌بر (Shortcut Learning) و راه‌حل ما

یکی از پدیده‌های مخرب در آموزش مدل‌های یادگیری عمیق، یادگیری میان‌بر (Shortcut Learning) است. اگر در هنگام تولید استدلال، حروف گزینه‌ها (مانند A, B, C, D) در متن پرامپت یا خروجی وجود داشته باشد، مدل معلم ممکن است استدلالی شبیه به «گزینه A درست است زیرا...» تولید کند. در مرحله فاین‌تیونینگ، مدل دانش‌آموز (Student Model) به جای یادگیریِ رابطه معنایی میان متن و رویداد، یاد می‌گیرد که صرفاً به دنبال الگوی نگارشیِ کلمه «گزینه A» بگردد!

برای جلوگیری از این مورد، یک رویکرد مهندسی پرامپتِ بسیار سخت‌گیرانه اتخاذ شد. در کدهای تولید دادگان، حروف گزینه‌ها به طور کامل حذف شده و تنها محتوای متنیِ گزینه‌ی درست (Golden Text) به مدل داده شد. پرامپت سیستم با قوانین بازدارنده (Negative Prompting) به شکل زیر پیاده‌سازی گردید:

```
#
golden_texts = [sample['options'][letter.strip()] for letter in
    golden_letters]
golden_combined_text = " AND ".join(golden_texts)
```

```
prompt = f"""You are an expert logical reasoning AI.
Based ONLY on the provided Context, explain logically why the following event:
"{golden_combined_text}"
is the direct cause of the Target Event:
"{sample['target_event']}"

CRITICAL RULES:
1. DO NOT use the words "Option", "Choice", "A", "B", "C", or "D".
2. Act as an independent analyst deducing the cause from the text.
...
"""
```

این ترفند تضمین کرد که استدلال‌های تولیدشده، صددرصد معنایی (Semantic) و به دور از هرگونه سوگیریِ ساختاری (Structural Bias) باشند.

۳.۳ چالش‌های تعامل با API و پیاده‌سازی منطق Resume

با توجه به حجم بالای داده‌های آموزشی، استخراج API با سه چالش عمده مهندسی نرم‌افزار مواجه بود:

۱. **محدودیت نرخ درخواست (Rate Limit):** سرورهای Groq دارای محدودیت‌های RPM (درخواست در دقیقه) و TPM (توکن در دقیقه) هستند.

۲. **خطاهای شبکه و توقف‌های ناگهانی:** قطعی اینترنت یا اتمام زمانِ نشست (Session Timeout) در پلتفرم Kaggle می‌توانست کل فرآیند را مختل کند.

برای غلبه بر این موانع، دو کار انجام دادیم. نخست، استفاده از کتابخانه Tenacity برای پیاده‌سازی توابع عقب‌نشینی نمایی (Exponential Backoff). با دکوراتور retry@، در صورت بروز خطای Rate Limit، سیستم به جای توقف، مدت زمان مشخصی (با ضریب فزاینده) توقف کرده و مجدداً تلاش می‌کرد. دوم، طراحی سیستم Checkpointing و پردازش جریانی (Streaming). داده‌ها به جای ذخیره در حافظه موقت (RAM)، بلافاصله پس از پردازش به صورت خط‌به‌خط در یک فایل JSONL افزوده (Append) می‌شدند. همچنین کدی نوشته شد تا در اجرای مجدد، فایل خروجی را اسکن کرده و شناسه‌ی (ID) نمونه‌های پردازش‌شده را استخراج کند تا فرآیند دقیقاً از همان نقطه‌ی توقف، ادامه یابد (Resume Logic).

۴.۳ تکنیک Data Filtering برای ایجاد High-Quality Subset

در پروژه‌های فاین‌تیونینگ، کیفیت داده‌ها به مراتب مهم‌تر از کمیت آن‌هاست (Quality over Quantity). در جریان تولید داده‌ها، برخی از درخواست‌ها با خطای قطعیِ API مواجه شده و یا مدلِ معلم خروجی نامفهومی تولید کرده بود. ورود این نویزها به مرحله آموزش می‌توانست وزن‌های شبکه (Network Weights) را تخریب کند. برای حل این مسئله، پیش از ورود به فاز آموزش، یک اسکریپت فیلتراسیون قطعی (Deterministic Filtering) روی فایل‌های خروجی اجرا شد. این اسکریپت تمام نمونه‌های تکراری (Duplicates) و ردیف‌هایی که برچسب "خطا در تولید استدلال" داشتند را به طور کامل حذف کرد. نتیجه‌ی این عملیات، استخراج یک **زیرمجموعه بی‌نقص (High-Quality Subset)** بود. آموزش مدل منحصراً بر روی این داده‌های خالص انجام گرفت که نقش بسزایی در پایداری فرآیند آموزش و کسب دقت خیره‌کننده نهایی داشت.

۴ معماری پیشنهادی: RAFT و تنظیم دقیق با QLoRA

پس از اثبات ناکارآمدی روش‌های مبتنی بر پرامپت (Prompting) در غلبه بر پیچیدگی‌های استنتاج ابداکتیو، فاز نهایی پروژه با هدف تزریق عمیقِ مهارتِ استدلال به وزن‌های شبکه عصبی آغاز گردید. در این بخش، معماری پیشنهادی

سیستم که ترکیبی از بازیابی اطلاعات و تنظیم دقیق پارامتر_کارآمد (Parameter-Efficient Fine-Tuning یا PEFT) است، تشریح می‌شود.

۱.۴ معرفی معماری RAFT (Retrieval-Augmented Fine-Tuning)

در رویکردهای سنتی، ما یا از RAG استفاده می‌کنیم (بدون تغییر وزن‌های مدل) و یا مدل را فاین‌تیون می‌کنیم (بدون ارائه کانتکست خارجی در زمان استنتاج). معماری RAFT (Retrieval-Augmented Fine-Tuning) یک پارادایم نوین است که این دو روش را با هم ترکیب می‌کند. در این تسک، هدف ما این نبود که مدل حقایق و رویدادها را حفظ کند، بلکه مدل باید یاد می‌گرفت که چگونه شواهد را از داخل یک متنِ ارائه‌شده استخراج کرده و روی آن‌ها استدلال کند. در معماری RAFT، ما مدل را مستقیماً بر روی نمونه‌هایی آموزش می‌دهیم که شامل کانتکست‌های بازیابی‌شدهٔ حاوی نویز (Retrieved Contexts) هستند. این کار به مدل می‌آموزد که در میان حجم انبوهی از اطلاعاتِ فیلترشده، دقیقاً به کدام جملات توجه کند (Attention Allocation) و شواهد نامربوط را نادیده بگیرد. این معماری، کلید اصلی پرش دقت سیستم از ۶۱% به بیش از ۸۱% بود.

۲.۴ تبدیل داده‌ها به فرمت استاندارد Chat Template

مدل‌های تنظیم‌شده برای دستورالعمل (Instruction-Tuned Models) مانند Qwen2.5-7B-Instruct، بر روی قالب‌های محاوره‌ایِ خاصی (نظیر ChatML) آموزش دیده‌اند. تغذیه داده‌های خام به این مدل‌ها در زمان فاین‌تیونینگ، باعث تخریب دانشِ قبلی آن‌ها (Catastrophic Forgetting) می‌شود. برای حفظ یکپارچگی، داده‌های غنی‌شده‌ی ما (شامل کانتکست، سوال، و استدلال‌های تولیدشده توسط Llama-3) با استفاده از تابع apply_chat_template توکنایزر (Tokenizer)، دقیقاً به فرمت استانداردِ چت تبدیل شدند. در این ساختار، نقش System وظیفه تعیین قالب، نقش User حامل کانتکست و صورت سوال، و نقش Assistant دربرگیرنده‌ی فرآیند Chain-of-Thought و پاسخ نهایی در تگ XML بود.

قطعه کد زیر، نحوه ساختاردهی این پیام‌ها را پیش از تزریق به مدل نشان می‌دهد:

```
def format_to_chat_template(example):
    messages = [
        {
            "role": "system",
            "content": "You are an expert AI specialized in Abductive Event Reasoning. First, briefly reason based on the provided Context. Then, output the uppercase letter(s) of the correct option(s) inside <ANSWER> tags."
        },
        {
            "role": "user",
            "content": f"Context: {example['context']}\n\nTarget Event: {example['target_event']}\n\nOptions:\nA) {example['options']['A']}\nB) {example['options']['B']}\nC) {example['options']['C']}\nD) {example['options']['D']}\n\nReason and output final answer in <ANSWER> tags."
        },
        {
            "role": "assistant",
            "content": f"{example['reasoning']}\n<ANSWER>{example['golden_answer']}</ANSWER>"
        }
    ]
    text = tokenizer.apply_chat_template(messages, tokenize=False)
    return {"text": text}
```

۳.۴ پیکربندی QLoRA (کوانتیزاسیون ۴ـ بیتی، تنظیمات r و lora_alpha)

آموزش کاملِ (Full Fine-Tuning) یک مدل ۷ میلیارد پارامتری در دقت ۱۶ـ بیتی نیازمند بیش از ۱۰۰ گیگابایت حافظه گرافیکی (VRAM) است که اجرای آن بر روی محیط Kaggle (کارت گرافیک T4 با ۱۶ گیگابایت حافظه) را کاملاً غیرممکن می‌سازد. برای غلبه بر این محدودیت سخت‌افزاری، از تکنیک QLoRA (Quantized Low-Rank Adaptation) استفاده شد. در این روش، وزن‌های مدلِ پایه به صورت ۴ـ بیتی فشرده می‌شوند و تنها ماتریس‌های کم‌رتبه (Low-Rank Matrices) و کوچک به عنوان آداپتور (Adapter) به شبکه اضافه شده و آموزش می‌بینند.

۱. تنظیمات کوانتیزاسیون (BitsAndBytes): مدل پایه با استفاده از فرمت ۴ـ بیتیِ NormalFloat (nf4) و تکنیک کوانتیزاسیونِ مضاعف (Double Quantization) بارگذاری شد تا ضمن کاهشِ شدید مصرف حافظه به حدود ۵ گیگابایت، افت دقت محاسباتی به حداقل برسد. نوع داده محاسباتی (Compute Dtype) نیز برای سازگاری با گرافیک‌های قدیمی‌تر، روی Float16 تنظیم گردید.

۲. پیکربندی LoRA: برای اینکه مدل ظرفیت یادگیریِ استدلال‌های پیچیده را داشته باشد، پارامترهای LoRA با دقت بالایی انتخاب شدند:

- **رتبه ماتریس (r = 16):** انتخاب رتبه ۱۶ یک تعادل عالی میان تعداد پارامترهای قابل آموزش و قدرت یادگیری فراهم کرد.
- **ضریب مقیاس‌پذیری (lora_alpha = 32):** این ضریب برای کنترل تاثیر وزن‌های جدید بر روی مدل پایه روی ۳۲ تنظیم شد.
- **ماژول‌های هدف (Target Modules):** برخلاف پیاده‌سازی‌های ساده که تنها لایه‌های توجه (Attention) را آموزش می‌دهند، آداپتورهای LoRA به تمامی لایه‌های خطی شبکه اعم از پروجکشن‌های توجه (q_proj, k_proj, v_proj, o_proj) و لایه‌های شبکه عصبی پیش‌خور (gate_proj, up_proj, down_proj) متصل شدند. این استراتژی باعث شد تا مدل توانایی تطبیق‌پذیری بسیار بالاتری با مفاهیم CoT پیدا کند.

در نهایت، فرآیند آموزش با استفاده از کتابخانه TRL و کلاس SFTTrainer با نرخ یادگیری 2×10^{-4} (Learning Rate) و زمان‌بندِ کسینوسی (Cosine Scheduler) برای دو ایپاک (Epoch) اجرا گردید.

۴.۴ تحلیل فرآیند آموزش و مانیتورینگ لاگ‌ها (Training Logs Analysis)

پس از رفع چالش‌های سخت‌افزاری و نرم‌افزاری، فرآیند تنظیم دقیق (Fine-Tuning) با موفقیت آغاز گردید. به منظور اطمینان از همگرایی (Convergence) مدل و جلوگیری از پدیده بیش‌برازش (Overfitting)، لاگ‌های آموزش در فواصل ۱۰ قدمی (Logging Steps) مانیتور شدند.

آموزش مدل به مدت ۲ ایپاک (Epoch) کامل به طول انجامید که زمان پردازش کل آن در حدود ۸۱۹۲ ثانیه (تقریباً ۲.۲ ساعت) بر روی پردازنده گرافیکی T4 بود. در این مدت، مدل حدود ۸.۱ میلیون توکن (Tokens) را پردازش نمود. تحلیل معیارهای ثبت‌شده در لاگ‌ها، روند تکامل و یادگیریِ بی‌نقص مدل را به شرح زیر اثبات می‌کند:

- **کاهش چشمگیر تابع زیان (Training Loss):** در آغاز آموزش (قدم ۱۰)، مقدار زیان برابر با ۲.۳۶۵۵ بود. این مقدار با یک شیبِ نزولیِ بسیار پایدار و بدون نوساناتِ مخرب (Spikes)، در پایان آموزش (قدم ۳۶۰) به مقدار خیره‌کننده ۰.۲۱۲۰ کاهش یافت. این افتِ شدیدَ نشان می‌دهد که مدل ساختارِ استدلال‌های Chain-of-Thought و فرمتِ خروجیِ تگ‌های XML را به طور کامل درونی‌سازی کرده است.
- **زمان‌بندی نرخ یادگیری (Learning Rate Schedule):** بررسی لاگ‌ها نشان‌دهنده عملکرد دقیقِ زمان‌بندِ کسینوسی (Cosine Scheduler) است. نرخ یادگیری در قدم‌های ابتدایی (فاز Warmup) تا حداکثر 2.0×10^{-4} افزایش یافت تا مدل بتواند از مینیمم‌های محلی (Local Minima) فرار کند. سپس به صورت تدریجی و نرم تا مقدار بسیار ناچیز 1.96×10^{-7} در قدم نهایی کاهش یافت که باعث تثبیتِ دقیقِ وزن‌های شبکه (Weights Stabilization) در انتهای آموزش گردید.

- **افزایش دقت توکن‌ها (Mean Token Accuracy):** یکی از جذاب‌ترینِ معیارهای ثبت‌شده، میانگین دقت توکن‌های پیش‌بینی‌شده در حین آموزش است. این معیار که در ابتدای آموزش تنها ۵۵.۵۳% بود، در پایان ایپاک دوم به ۹۴.۳۷% رسید. این بدان معناست که مدل در انتهای فرآیند، نه تنها استدلال‌های منطقی را با دقت بالایی تولید می‌کرد، بلکه در پیش‌بینیِ توکنِ بعدی (Next Token Prediction) برای کلماتِ کلیدیِ علت و معلولی، تقریباً به قطعیت کامل رسیده بود.

- **کاهش آنتروپی (Entropy) و افزایش قطعیت:** آنتروپی در مدل‌های زبانی نشان‌دهنده میزان شک و عدم قطعیت (Uncertainty) مدل در انتخاب کلمات است. در شروع آموزش، آنتروپی روی عدد ۱.۵۱۲ قرار داشت که نشان‌دهنده سردرگمیِ مدلِ پایه در مواجهه با پرامپت‌های جدید بود. با پیشرفت آموزش، این عدد به ۰.۲۲۷۲ سقوط کرد. این کاهش چشمگیر ثابت می‌کند که مدل از حالت «حدس تصادفی» خارج شده و به یک «تحلیل‌گر قاطع» با استدلال‌های محکم تبدیل شده است.

در قطعه زیر، خلاصه‌ای از لاگ‌های کلیدیِ ابتدا، میانه و انتهای فرآیند آموزش جهت استناد آورده شده است:

```
# Initial Phase (Warmup & High Loss)
Step: 0010 | Epoch: 0.05 | Loss: 2.3655 | LR: 1.20e-04 | Token_Acc: 0.5553
Step: 0020 | Epoch: 0.11 | Loss: 1.6483 | LR: 2.00e-04 | Token_Acc: 0.6342

# Mid Phase (Steady Convergence)
Step: 0180 | Epoch: 0.98 | Loss: 0.4073 | LR: 1.10e-04 | Token_Acc: 0.9005

# Final Phase (Stabilization & High Confidence)
Step: 0350 | Epoch: 1.91 | Loss: 0.2057 | LR: 1.16e-06 | Token_Acc: 0.9418
Step: 0360 | Epoch: 1.97 | Loss: 0.2120 | LR: 1.96e-07 | Token_Acc: 0.9420

# Overall Training Metrics
{'train_runtime': '8192s', 'train_loss': '0.5469', 'entropy': '0.2272', '
   mean_token_accuracy': '0.9437'}
```

در مجموع، تحلیل این متریک‌ها نشان می‌دهد که معماری RAFT در ترکیب با QLoRA نه تنها از نظر محاسباتی بسیار بهینه بوده، بلکه از نظر ریاضیاتی و یادگیریِ بازنمایی‌ها (Representation Learning) نیز یک همگرایی بی‌نقص و ایده‌آل را تجربه کرده است.

۵ چالش‌های فنی و راه‌حل‌های مهندسی

یکی از ارزشمندترین دستاوردهای این پروژه، مواجهه با چالش‌های پیچیده‌ی نرم‌افزاری و سخت‌افزاری در پیاده‌سازی معماری‌های نوینِ پردازش زبان طبیعی و یافتن راه‌حل‌های مهندسی برای عبور از آن‌ها بود. در این بخش، سه چالش اساسی که در مسیر تنظیم دقیق (Fine-Tuning) مدل بروز کرد و نحوه برطرف‌سازی آن‌ها به تفصیل شرح داده شده است.

۱.۵ ناسازگاری سخت‌افزاری گرافیک‌های T4 با فرمت BFloat16

شرح چالش: مدل‌های خانواده Qwen2.5 به صورت پیش‌فرض با فرمت داده‌ی اعشاری ۱۶ بیتیِ مختص هوش مصنوعی، یعنی BFloat16، آموزش دیده‌اند. با این حال، GPUهای رایگان در محیط Kaggle (مدل NVIDIA T4) مبتنی بر معماری قدیمی‌ترِ Turing هستند که از نظر سخت‌افزاری از عملیات ماتریسی با فرمت BFloat16 پشتیبانی نمی‌کنند. در زمان آغاز فرآیند آموزش و در مرحله‌ی محاسبه گرادیان‌ها توسط Accelerator، سیستم با خطای زیر متوقف می‌شد:

```
NotImplementedError: "_amp_foreach_non_finite_check_and_unscale_cuda" not
   implemented for 'BFloat16'
```

راه‌حل مهندسی: تغییر ساده‌ی پارامتر torch_dtype به تنهایی برای حل این مشکل کافی نبود، زیرا لایه‌های آداپتورِ LoRA در زمان تزریق، مجدداً کانفیگ اصلیِ مدل را خوانده و با فرمت BFloat16 ساخته می‌شدند. برای حل قطعیِ این ناسازگاری، یک رویکرد Weights Surgery پیاده‌سازی شد. پس از ساخت مدل و پیش از آغاز آموزش، با استفاده از یک حلقه بر روی تمام پارامترهای شبکه، هر پارامتری که نیاز به آموزش داشت (requires_grad=True) به صورت اجباری به Float32 (برای محاسبه دقیق گرادیان‌ها) و سایر وزن‌های باقی‌مانده از BFloat16 به Float16 تبدیل شدند.

۲.۵ تغییرات ساختاری (Breaking Changes) در به‌روزرسانی‌های جدید TRL و Transformers

شرح چالش: Hugging Face به سرعت در حال توسعه است. در زمان پیاده‌سازی این پروژه، جدیدترین نسخه‌های کتابخانه TRL (نسخه ۰.۹ به بالا) تغییراتی را معرفی کرده بودند که باعث از کار افتادنِ کدهای استانداردِ SFTTrainer می‌شد:

- پارامترهای dataset_text_field و max_seq_length از کلاس SFTTrainer حذف شده و به کلاس جدیدی به نام SFTConfig منتقل شده بودند. علاوه بر این، نام پارامتر max_seq_length به صورت ناگهانی به max_length تغییر یافته بود.
- پارامتر نام‌آشنای tokenizer کاملاً منسوخ شده و جای خود را به processing_class داده بود تا از پردازشگرهای چندوجهی (Multimodal) نیز پشتیبانی کند.
- ارسال هم‌زمان یک مدلِ از پیش‌تبدیل‌شده به PeftModel به همراه peft_config به تابع آموزش، منجر به خطای ValueError می‌شد.

راه‌حل مهندسی: با بررسی دقیقِ مستنداتِ جدید و سورس‌کدهای (Source Codes) کتابخانه TRL، معماری ساخت مدل کاملاً بازنویسی شد. به جای تبدیل دستیِ مدل، مدل خام به همراه LoraConfig به صورت مجزا به SFTTrainer تحویل داده شد تا خودِ Trainer وظیفه‌ی ترکیب را بر عهده بگیرد. همچنین تنظیمات آموزش تماماً به SFTConfig منتقل گردید و با تغییرِ نام پارامترها به processing_class و max_length، کدهای پروژه با آخرین استانداردهای لبه‌ی تکنولوژی (State-of-the-Art) همگام‌سازی شدند.

۶ ارزیابی نتایج و تحلیل

۱.۶ معیارهای ارزیابی (Evaluation Metrics)

در این پروژه، با توجه به ماهیت استنتاجی مسابقه (SemEval-2026 Task 12) و امکان وجود چند علت هم‌زمان برای یک رویداد، عملکرد مدل به صورت **ارزیابی در سطح نمونه** (Instance-level Evaluation) سنجیده می‌شود. این بدان معناست که خروجی مدل برای هر سوال به طور کاملاً مستقل با جواب‌های استاندارد همان سوال مقایسه شده و سپس میانگین این امتیازات به عنوان دقت کل (Accuracy) در نظر گرفته می‌شود.

فرض کنید مجموعه G (Golden) نشان‌دهنده گزینه‌های درستِ ارزیاب انسانی و مجموعه P (Predicted) نشان‌دهنده گزینه‌های پیش‌بینی‌شده توسط مدل برای یک نمونه‌ی مشخص باشد. تابع امتیازدهی برای هر نمونه به صورت ریاضی در رابطه زیر تعریف می‌گردد:

$$\text{Score}(P, G) = \begin{cases} 1.0 & \text{if} P = G \text{ and } |P| > 0 \\ 0.5 & \text{if} P \subset G \text{ and } |P| > 0 \\ 0.0 & \text{otherwise} \end{cases} \quad (۱)$$

بر اساس این رابطه، سه حالت برای هر پیش‌بینی متصور است:

- **تطابق کامل (Full Match) - امتیاز ۱.۰:** مدل دقیقاً تمامی علت‌های درست را تشخیص داده و هیچ گزینه اضافه‌ای در پاسخ آن وجود نداشته باشد ($P = G$).
- **تطابق نسبی (Partial Match) - امتیاز ۰.۵:** پاسخ مدل یک زیرمجموعه‌ی سره و غیرتهی از جواب‌های درست باشد ($P \subset G$).
- **نادرست (Incorrect) - امتیاز ۰.۰:** مدل یک گزینه کاملاً غلط را انتخاب کرده باشد، یا ترکیبی از گزینه‌های درست و غلط را برگرداند ($P \not\subset G$).

در نهایت، دقت کل سیستم برابر است با میانگین امتیازات کسب شده در تمامی N نمونه‌ی موجود در دادگان ارزیابی:

$$\text{Accuracy} = \frac{1}{N}\sum_{i=1}^{N} \text{Score}(P_i, G_i) \quad (۲)$$

۲.۶ مقایسه جامع نتایج: ارزیابی مدل نهایی بر روی دادگان dev_set

پس از اتمام فرآیند آموزش، به منظور سنجش قدرت تعمیم‌پذیری (Generalization) مدل و اطمینان از عدم وقوع بیش‌برازش (Overfitting)، مدل فاین‌تیون‌شده بر روی دادگان رسمی توسعه مسابقه (dev_set) که شامل 400 نمونه کاملاً جدید و دیده نشده بود، مورد ارزیابی قرار گرفت. این ارزیابی با استفاده از معماری کامل سیستم، یعنی ترکیب مدل فاین‌تیون‌شده، Semantic RAG و تولید استدلال منطقی (CoT) انجام پذیرفت.

اجرای این ارزیابی بر روی 400 نمونه، حدود 1 ساعت و 16 دقیقه زمان برد (میانگین 11.46 ثانیه برای پردازش هر نمونه به همراه بازیابی اطلاعات و تولید استدلال). نتایج نهایی این ارزیابی یک دستاورد خیره‌کننده را به همراه داشت: سیستم موفق به کسب مجموع امتیاز 328.5 از 400 گردید که معادل **دقت 82.12%** (Accuracy: 0.8213) است.

۳.۶ تحلیل خطا (Error Analysis) بر روی نمونه‌های شکست‌خورده

با وجود دقت بسیار بالای 82.12%، بررسی نمونه‌هایی که امتیاز صفر دریافت کرده‌اند، بینش عمیقی در خصوص رفتار مدل و محدودیت‌های سیستم امتیازدهی ارائه می‌دهد. در فرمول ارزیابی این مسابقه، سیستم در برابر انتخاب‌های اضافه به شدت مجازات‌گر (Strict) عمل می‌کند؛ به این معنا که اگر مدل علت‌های درست را پیدا کند اما حتی یک علت نامربوط را به آن‌ها بیفزاید، کل امتیاز آن نمونه صفر خواهد شد.

تحلیل کیفی خطاهای استخراج‌شده از لاگِ ارزیابی دادگان dev_set، خطاها را در دو دسته اصلی طبقه‌بندی می‌کند:

1. پدیده بیش‌تخصیصی یا جامع‌نگری افراطی (Over-attribution / Over-generation): بیشترین سهم خطاهای مدل مربوط به مواردی است که مدل، پاسخ صحیح (Golden) را به درستی تشخیص داده، اما گزینه‌های دیگری را نیز به عنوان علت‌های مکمل و هم‌زمان دخیل دانسته است.

- **شناسه q-2057:** پاسخ صحیح B بوده است، اما مدل A,B را پیش‌بینی کرده است.
- **شناسه q-2072:** پاسخ صحیح D بوده است، اما مدل B,D را پیش‌بینی کرده است.
- **شناسه q-2092:** پاسخ صحیح A,B بوده است، اما مدل A,B,C را پیش‌بینی کرده است.

دلیل این رفتار آن است که مدل در مواجهه با کانتکست‌های پیچیده‌ی خبری یا سیاسی، تمامی پیش‌زمینه‌ها و عوامل موثرِ جانبی را نیز در زنجیره استدلال خود (Reasoning Chain) دخیل می‌کند. از دیدگاه معنایی، خروجی مدل لزوماً «غلط» نیست، بلکه از دیدگاه برچسب‌گذارِ انسانیِ مسابقه، گزینه‌های اضافه نقشِ علتِ «مستقیم» (Direct Cause) را نداشته‌اند.

2. تداخل علت ریشه‌ای با علت محرک (Root Cause vs. Proximate Cause): دسته دوم خطاها بسیار جالب و قابل‌تأمل هستند؛ در این حالت، زمانی که با یک زنجیره متوالی از اتفاقات روبه‌رو هستیم، مدل فریبِ رویداد آغازگر (Trigger) را می‌خورد و جرقه اولیه را به جای علتِ نهایی و بی‌واسطه انتخاب می‌کند.
برای درک بهتر این چالش، نمونه q-2059 مثال بسیار روشنی است:

- پیش‌بینی مدل: A (سخنرانی ترامپ در میان جمعیت)
- پاسخ صحیح: D

در این نمونه (که به ماجرای استیضاح اشاره دارد)، مدل با یک نگاه کل‌نگر، سخنرانی تهییج‌کننده را به عنوان علت ریشه‌ای و موتور محرکِ وقایعِ پس از آن تشخیص داده است. با این حال، از دیدگاه سخت‌گیرانه‌ی طراحان مسابقه، تنها علتِ رسمی و حقوقیِ استیضاح (یعنی گزینه D) به عنوان پاسخِ درست پذیرفته می‌شود. بروز این دست از خطاها به خوبی نشان می‌دهد که تفکیکِ حلقه‌های به هم پیوسته در یک زنجیره طولانیِ علت و معلولی، تا چه حد دارای پیچیدگیِ ذاتی است و مرز باریک میان «علت اولیه» و «علت نهایی»، چالشی عمیق در تسک‌های استنتاجی محسوب می‌شود.

۴.۶ مقایسه جامع نتایج آزمایش‌ها: از Prompting تا Fine-Tuning

برای درک بهتر روند تکامل معماری سیستم، نتایج ارزیابیِ تمامی رویکردهای پیاده‌سازی‌شده بر روی دادگان اولیه ۲۰۰ نمونه‌ای، در جدول ۱ گردآوری شده است.

جدول ۱: مقایسه عملکرد معماری‌های مختلف بر روی دادگان ارزیابی مسابقه

معماری / رویکرد (Approach)	مجموع امتیاز	دقت (Accuracy)	تغییر
Baseline: Semantic RAG + Zero-Shot CoT	122.0	61.00%	–
Prompting: Semantic RAG + Few-Shot CoT	111.5	55.75%	-5.25%
Prompting: Few-Shot CoT + Self-Consistency	113.0	56.50%	-4.50%
Proposed: RAFT + QLoRA Fine-Tuning	**163.5**	**81.75%**	**+20.75%**

همان‌طور که در جدول ۱ مشاهده می‌شود، تلاش برای حل مسئله با استفاده از رویکردهای مهندسی پرامپت (Prompt Engineering) با شکست مواجه شد. تزریق نمونه‌های آموزشی (Few-Shot) به دلیل بروز پدیده سرریز اطلاعات و سوگیری نمونه‌ها (Demonstration Bias)، دقت را به ۵۵.۷۵% کاهش داد. استفاده از مکانیزم پرهزینه‌ی رای‌گیری اکثریت (Self-Consistency) نیز تنها توانست دقت را به ۵۶.۵۰% برساند که همچنان پایین‌تر از خط پایه بود.
با این حال، تغییر استراتژی به سمت تنظیم دقیق (Fine-Tuning) با استفاده از معماری RAFT و دادگانِ غنی‌شده با استدلال‌های مدل Llama-3، یک دستاورد شگرف به همراه داشت. مدل فاین‌تیون‌شده توانست با کسب امتیاز ۱۶۳.۵، دقت سیستم را به ۸۱.۷۵% ارتقا دهد. این مورد نشان می‌دهد که مدل با موفقیت مهارت استنتاج ابداکتیو را درونی‌سازی کرده و به جای تقلید کورکورانه از الگوها، قادر به تحلیل عمیق روابط علت و معلولی در متون ناشناخته است.

۵.۶ تحلیل خطا (Error Analysis) روی نمونه‌های شکست‌خورده

با وجود دقت بسیار بالای ۸۱.۷۵%، بررسی ۱۸.۲۵% خطای باقی‌مانده اطلاعات ارزشمندی برای بهبودهای آتی فراهم می‌کند. با استخراج و بررسی لاگ‌های خروجی (فایل predictions.jsonl)، خطاها عموماً در دو دسته زیر طبقه‌بندی می‌شوند:

۱. **استخراج ناقص در رویدادهای چندعلتی (Partial Extraction):** در بسیاری از نمونه‌ها که رویداد هدف دارای دو علت هم‌زمان بود (مثلاً گزینه‌های A,C)، مدل موفق به کشف یکی از علت‌ها می‌شد اما علت دوم را نادیده می‌گرفت و امتیاز 0.5 دریافت می‌کرد. دلیل این امر، تمایل ذاتی مدل‌های زبانی به توقف تولید (Early Stopping) پس از یافتن اولین پاسخ منطقی است.

۲. **استنتاج‌های نیازمند دانش جهانی (World Knowledge Deficit):** در نمونه‌هایی که پاسخ صراحتاً در متن نیامده بود و نیازمند ترکیب اطلاعاتِ متن با دانش عمومی بود، مدل دچار خطا می‌شد. به عنوان مثال در نمونه q-22 مربوط به محدودیت‌های شیوع ویروس کرونا (Coronavirus lockdowns)، مدل نتوانست ارتباط غیرمستقیم میان توقف اقتصاد جهانی و سیاست‌های قرنطینه محلی را به درستی ترسیم کند و گزینه‌ی اشتباهی را برگزید.

۷ نتیجه‌گیری و کارهای آینده

۱.۷ نتیجه‌گیری

در این پروژه، سیستم هوشمندی برای حل چالش کشف علت و معلولی رویدادها در متون (Abductive Event Reasoning) در چارچوب مسابقه SemEval-2026 توسعه یافت. آزمایش‌ها نشان داد که مدل‌های زبانیِ پایه، با وجود ابعاد بزرگ، در استنتاج‌های ابداکتیوِ پیچیده فاقد قطعیت و پایداری لازم هستند. برای حل این مشکل، یک جریان‌کار (Workflow) مهندسی‌شده شامل تولید داده‌های غنی مبتنی بر Chain-of-Thought با استفاده از مدل‌های قدرتمند، مقابله با پدیده Shortcut Learning و در نهایت پیاده‌سازی معماری پیشرفته RAFT به همراه QLoRA طراحی گردید. با غلبه بر چالش‌های سطح پایینِ سخت‌افزاری (نظیر ناسازگاری گرافیک‌های T4 با فرمت BFloat16)، مدل Qwen2.5-7B با موفقیت تنظیم دقیق شد. نتایج نشان داد که این رویکرد یکپارچه توانسته است با ارتقای ۲۰.۷۵ درصدی دقت نسبت به خط پایه، سیستم را به دقت استثنایی ۸۱.۷۵% برساند و پتانسیل بالای تکنیک‌های مبتنی بر PEFT در درونی‌سازی مهارت‌های استدلالی را به اثبات برساند.

۲.۷ کارهای آینده (Future Work)

برای ارتقای سیستم در پژوهش‌های آتی، راهکارهای زیر پیشنهاد می‌گردد:

- **بازیابی هیبریدی (Hybrid Retrieval):** ترکیب جستجوی متراکم معنایی (Dense Retrieval با مدل‌های Embedding) و جستجوی پراکنده و کلیدواژه‌محور (نظیر BM25) برای بهبود فاز RAG و کاهش خطای گم‌شدگی اطلاعات کلیدی.

- **تکنیک Direct Preference Optimization (DPO):** به جای فاین‌تیونینگ نظارت‌شده (SFT)، می‌توان از DPO استفاده کرد تا به مدل آموزش داد که چرا یک مسیر استدلالی از مسیر دیگر بهتر است. این کار خطای مدل در رویدادهای چندعلتی را به شدت کاهش می‌دهد.

- **توسعه روی مدل‌های بزرگ‌تر:** پیاده‌سازی همین معماری بر روی مدل‌های متن‌باز با پارامترهای بیشتر نظیر Llama-3-14B یا Qwen-14B در صورت دسترسی به سخت‌افزارهای قدرتمندتر، می‌تواند سقف دقت را به بالای ۹۰% ارتقا دهد.