به نام خدا

دانشگاه تهران

دانشکدگان فنی

دانشکده مهندسی برق و کامپیوتر

درس سیستم های چند عاملی پیشرفته

پروژه ی درس مرحله ی 1

نام و نام خانوادگی: ارشیا مخلص

شماره دانشجویی: 810104247

آذر ماه 1404

فهرست

سوال 1.

در سال های اخیر، عامل های مبتنی بر LLM از یک «چت بات قوی» به اجزای هوشمندی تبدیل شده اند که می توانند مثل یک عضو تیم فکر کنند، برنامه بریزند، ابزار استفاده کنند و در کنار عامل های دیگر کار جمعی انجام دهند. مخزن هایی مثل Multi-Agent-Papers و Awesome Multi-Agent Papers تصویر روشنی می دهند از این که جامعه تحقیقاتی چطور از عامل تکی شروع کرده و به طراحی جوامع بزرگ عامل ها، چارچوب های چندعاملی و بنچمارک های پیچیده برای سنجش همکاری، رقابت و هماهنگی رسیده است.

مقدمه

در سطح پایه، LLM-based agentیعنی مدلی که فقط متن تولید نمی کند، بلکه در یک «پوسته عامل» قرار می گیرد و می تواند اهداف را بفهمد، کار را به گام های کوچک بشکند، با محیط و ابزارها تعامل کند و نتایج را ارزیابی کند. هرچه وظیفه پیچیده تر و بلندمدت تر می شود، یک عامل تنها کمتر کافی است؛ به همین دلیل کارها به سمت چندعامل رفته اند تا مثل یک تیم انسانی، نقش ها و مسئولیت ها تقسیم شوند و از تعامل، نقد متقابل و تخصصی سازی سود ببرند.

پایه LLM-BASED AGENTS

در لایه ی پایه، کارهایی مثل ReAct نشان می دهند چگونه می توان استدلال زنجیره ای و عمل (مثلاً وب جستجو، اجرای کد، فراخوانی API) را در یک حلقه ادغام کرد تا عامل بتواند در محیطی مثل WebShop یا ALFWorld کار انجام دهد. فهرست هایی مثل Awesome-Agent-Papers و مخازن «LLM_MultiAgents_Survey_Papers»مجموعه ای بزرگ از این عامل های تک نقشی را مستند می کنند که واحد اصلی بسیاری از سیستم های چندعاملی بعدی هستند.

همکاری چندعاملی(MULTI-AGENT COLLABORATION)

در سطح بعد، کارها به همکاری چندعامل می رسند؛ جایی که چند LLM یا چند شخصیت از یک LLM در قالب نقش های متفاوت کنار هم قرار می گیرند. نمونه های شاخص در README اولیه شامل MetaAgents برای هماهنگی اجتماعی و تشکیل تیم در سناریوی نمایشگاه کار، GameGPTبرای توسعه بازی با نقش هایی مثل مدیر، مهندس و بازبین، LLM-Coordinationبرای ارزیابی توان هماهنگی و Theory of Mind در بازی هایی مثل Hanabi وOvercooked ، و Co-NavGPT برای ناوبری تصویری چندرباتی با استفاده از مدل های دید-زبان است. در کنار این ها، کارهایی مثلCAMEL ، AutoAgents، Multi-Agent Collaboration، و AutoGen الگوهای عمومی تری از گفتگو، نقد متقابل، self-playو تقسیم کار میان عامل ها ارائه می دهند.

چارچوب ها و پلتفرم ها(FRAMEWORKS & BENCHMARKS)

برای اینکه این ایده ها فقط در مثال های پراکنده نمانند، مجموعه ای از چارچوب ها و بنچمارک ها طراحی شده اند : AutoGenو AgentScope و MALLM بسترهای عمومی برای تعریف عامل ها، نقش ها و جریان مکالمه و ابزارها فراهم می کنند، در حالی کهAgentBench ، MultiAgentBench، Collab-Overcooked، و SmartPlayروی سنجش توان عامل ها در همکاری، رقابت، برنامه ریزی و تعامل با محیط تمرکز دارند. این

بنچمارک ها معمولاً سناریوهای بازی، حل مسئله، QA، استفاده از ابزار و ارزیابی امنیت/ریسک را در بر می گیرند و معیارهایی مثل موفقیت در کار، پایداری همکاری و کیفیت استدلال را می سنجند.

سامانه های کاربردی دامنه محور

بخش بزرگی از لیست به کاربردهای دامنه ای اختصاص دارد؛ در نرم افزار و کدنویسی، کارهایی مثلChatDev ، MetaGPT، CodeRو AutoSafeCoder چند عامل با نقش های مهندس، طراح، بازبین و تست را برای تحلیل نیازمندی، طراحی، پیاده سازی و دیباگ کنار هم قرار می دهند. در پزشکی، Agent Hospital، MDAgents، Zodiacو کارهای مشابه از عامل ها برای تصمیم سازی تشخیصی، آموزش پزشکی و شبیه سازی بیمارستان استفاده می کنند. در رباتیک و ناوبری، RoCo، Co-NavGPTو MALMM با تکیه بر LLM/VLM به عنوان برنامه ریز سطح بالا، چند ربات را برای ناوبری و دستکاری هماهنگ می کنند. حوزه های دیگر مثل سیستم های توصیه گر(AgentCF) ، مالی(TwinMarket) ، حقوق و دادگاه های شبیه سازی شده (AgentCourt) و برنامه ریزی شهری (urban planning agents) هم نشان می دهند که الگوی چندعاملی تقریباً به هر حوزه ای که تصمیم گیری و تعامل پیچیده دارد قابل تعمیم است.

شبیه سازی اجتماعی و جوامع عامل ها

گروه دیگری از کارها هدفشان حل یک وظیفه مهندسی نیست، بلکه مطالعه رفتار جمعی و اجتماعی عامل ها است؛ Generative Agents، AgentSociety، OASIS، TwinMarketو ده ها کار مشابه، شهرها، بازارها یا جوامع بزرگ مصنوعی می سازند که در آن صدها تا میلیون ها عامل LLM زندگی روزمره، تصمیم اقتصادی، کنش سیاسی یا تعامل اجتماعی را شبیه سازی می کنند. این خط پژوهشی برای مطالعه شکل گیری هنجارها، ظهور همکاری پایدار، تکامل فرهنگی و حتی هم ترازی اجتماعی مدل ها اهمیت دارد و نشان می دهدLLM ها فراتر از حل مسائل تکی، می توانند واحدهای سازنده «جامعه های مصنوعی» باشند.

ارزیابی و بهبود مدل

برای نگه داشتن این سیستم ها در مسیر درست، بخش بزرگی از کارها روی ارزیابی و بهبود تمرکز دارد؛ رویکردهایی مثلmulti-agent debate ، Agent-as-a-Judge، ChatEvalو AutoDefense سعی می کنند از خود عامل ها به عنوان منتقد، داور یا مهاجم/دفاع کننده استفاده کنند تا هم کیفیت استدلال و factuality بالا برود و هم ریسک هایی مثل jailbreak یا تصمیم غلط شناسایی شود. بنچمارک هایی مثلAgentBench ، Collab-Overcooked، Decryptoو ارزیابی های Theory of Mind برای چندعامل کمک می کنند بفهمیم این سامانه ها تا چه حد می توانند نیات دیگران را درک کنند، برنامه مشترک بسازند و در محیط های پرریسک قابل اعتماد عمل کنند

جمع بندی و مسیرهای آینده

در مجموع، اینREADME ها و فهرست ها نشان می دهند که LLM-based multi-agent systems از «چند مدل حرف زن» فراتر رفته اند و در حال تبدیل شدن به معماری های جدی برای حل مسائل پیچیده، شبیه سازی جوامع و خودکارسازی جریان های کاری هستند، اما هنوز چالش های مهمی مثل مقیاس پذیری، پایداری همکاری، Theory of Mindقابل اعتماد، ایمنی و ارزیابی دقیق باز هستند. همین چالش ها نقطه شروع خوبی برای پروژه های جدید است: از طراحی پروتکل های بهتر برای ارتباط عامل ها، تا ساخت بنچمارک های دقیق تر برای هماهنگی و طراحی چارچوب هایی که بتوانند این سیستم ها را در کاربردهای واقعی امن و پایدار نگه دارند.

سوال 2.

هدف کلی مقاله چیست؟

هدف کلی این مقاله ارزیابی توانایی عامل های مبتنی بر LLM برای همکاری چندعاملی در یک مأموریت متنی جست وجو و خنثی سازی بمب است، به خصوص از نظر برنامه ریزی مشترک و استنتاج Theory of Mind (استدلال درباره وضعیت ذهنی دیگر عامل ها). مقاله می خواهد نشان دهد که این عامل ها بدون آموزش ویژه و به صورت صفر-شات می توانند رفتاری نزدیک به روش های پیشرفته MARL و برنامه ریزی داشته باشند، اما هنوز به خاطر خطا در مدیریت کانتکست بلندمدت و هالوسینیشن از وضعیت محیط، برنامه ریز یا هم تیمی ایده آلی نیستند.

عامل ها چه هستند و محیط بازی چگونه است؟

در این تحقیق سه عاملLLM محور با نام هایAlpha ، Bravoو Charlie تعریف شده اند که هر کدام نقش یک متخصص در تیم خنثی سازی بمب را بازی می کنند و با استفاده از یک مدل زبانی) مثل GPT-3.5 یا (GPT-4 تصمیم می گیرند. محیط بازی یک دنیای متنی است که روی یک گراف از اتاق ها (نودها) ساخته شده؛ در هر اتاق ممکن است بمب های چندمرحله ای رنگی وجود داشته باشد و هر عامل فقط اتاق فعلی، ابزارهای خودش، پیام های تیم و وضعیت کلی امتیاز را می بیند و می تواند بین حرکت به اتاق های مجاور، بازرسی بمب و استفاده از سیم برهای رنگی انتخاب کند تا بمب ها را به طور هماهنگ خنثی کنند.

عامل ها چگونه با هم ارتباط برقرار می کنند؟

ارتباط بین عامل ها کاملاً متنی است: در هر راند، هر عامل علاوه بر انتخاب اکشن، یک پیام متنی برای تیم می فرستد که بلافاصله از طریق کانال چت داخلی به صورت broadcast در مشاهدات دور بعد همه اعضا نمایش داده می شود. خود محیطِ بازی و یک لایه ی «text game interface» این پیام ها را از متن طبیعی به اعمال انتزاعی) مثلاً «move to room X» یا («inspect bomb Y» رمزگشایی می کند و در عوض، وضعیت محیط را دوباره به صورت توضیح متنی (محتوای اتاق، نتیجه ی عمل، موقعیت هم تیمی ها، پیام های قبلی) به عامل ها برمی گرداند.

آیا عامل ها با هم همکاری می کنند؟ چگونه و چرا؟

بله، همکاری صریح و ضمنی هر دو شکل می گیرد: عامل ها وظایف را بین خود تقسیم می کنند، یک عامل) معمولاً Alphaنقش رهبر را می گیرد و به بقیه می گوید به کدام اتاق بروند یا کدام بمب را بررسی/خنثی کنند و دیگران هم غالباً طبق این پیشنهادها عمل می کنند. این همکاری به این دلیل ضروری است که اطلاعات و ابزارها بین عامل ها پخش شده (هرکس تعدادی سیم بر رنگی دارد و فقط اتاق فعلی را می بیند)، بنابراین برای رسیدن به حداکثر امتیاز باید هم در فضا (پخش شدن در اتاق های مختلف) و هم در توالی برش های رنگی بمب ها با هم هماهنگ شوند؛ نتایج نشان می دهد تیم های مبتنی بر GPT-4 با این همکاری می توانند همه بمب ها را در زمان محدود خنثی کنند.

منظور از THEORY OF MIND در این مقاله چیست؟

Theory of Mind در این مقاله یعنی توانایی عامل برای استدلال درباره «آن چه خود و دیگران می دانند یا نمی دانند»، در سه سطح:

Introspection: توصیف وضعیت دانشی و باوری خود عامل.

First-order ToM: استنتاج این که یک هم تیمی در مورد وضعیت جهان چه باوری دارد) مثلاً آیا Bravo می داند بمب در اتاق ۳ است یا نه.(

Second-order ToM: استنتاج این که هم تیمی درباره وضعیت ذهنی خود عامل چه فکری می کند) مثلاً آیا Charlie می داند که Alpha از محتوای اتاق ۵ خبر دارد.(

در طول مأموریت، بعد از بعضی اکشن ها از عامل ها سؤالات ToM پرسیده می شود) الهام گرفته از تست–Sally Anne) و پاسخ های متنی آن ها توسط انسان ها با توجه به تاریخچه کامل تعاملات برچسب گذاری می شود.

حافظه در این مقاله چگونه مدیریت و نگهداری می شود؟

حافظه ی عامل ها از دو بخش اصلی تشکیل شده است:

History: در ورودی مدل: بخشی از تاریخچه ی راندهای قبلی (قوانین بازی، چند راند اخیر شامل مشاهدات، اعمال و پیام ها) تا حد ظرفیت ورودی مدل در prompt نگه داشته می شود.

Belief State: صریح) در حالت :(GPT-4 + Belief بعد از هر مشاهده، عامل با پرامپت خاصی وادار می شود یک متن ساخت یافته به روز کند که باورهای مهمش درباره ی محیط) مثلاً «Bomb 1 is in Room 0 with sequence Red», «Room 3 connected to Room 0») را ذخیره می کند؛ این متن در پرامپت راندهای بعد تکرار می شود تا محدودیت طول context جبران شود و خطاهای ناشی از فراموشی و هالوسینیشن کاهش یابد.

نتایج اصلی مقاله چیست؟

از نظر عملکرد وظیفه: تیم مبتنی بر ChatGPT (gpt-3.5) همه بمب ها را خنثی نمی کند و میانگین امتیاز حدود ۴۳ می گیرد، در حالی که تیم GPT-4 و GPT-4+Belief امتیاز کامل ۹۰ می گیرند؛ GPT-4+Belief تعداد راندها را نسبت به GPT-4 عادی تقریباً نصف می کند (۲۸/۳ → ۱۲/۳ راند) و درصد اکشن معتبر را از حدود ۷۲٪ به ۸۶٪ می رساند، که آن را از نظر کارایی به MAPPO نزدیک می کند.

نسبت به خط مبناها CBS Planner :(برنامه ریز متمرکز با اطلاعات کامل) در ۶ راند به جواب بهینه می رسد و MAPPO پس از میلیون ها گام آموزش در حدود ۱۱ راند، در حالی که عامل های LLM بدون آموزش خاص و با تصمیم گیری کاملاً غیرمتمرکز به عملکردی نسبتاً نزدیک می رسند.

از نظر ToM: دقت ToM در ChatGPT برای introspection حدود ۷۹٪، برای first-order حدود ۴۲٪ و برای second-order حدود ۱۲٪ است؛ در GPT-4 این اعداد به حدود ۸۰٪، ۶۰٪ و ۶۴٪ می رسند و در GPT-4+Belief به حدود ۹۷٪، ۸۰٪ و ۶۹٪ افزایش می یابند، که نشان می دهد نمایش صریح باورها هم عملکرد تیمی و هم استنتاج ToM را بهبود می دهد.

از نظر کیفی: الگوهایی مثل رهبری Alpha) emergent) دستور می دهد، بقیه تبعیت می کنند(، کمک به هم تیمی، حل تعارض و اشتراک اطلاعات در رفتار تیم های GPT-4 دیده می شود، ولی دو نوع شکست سیستماتیک هم مشاهده می شود: نادیده گرفتن محدودیت های بلندمدت (مثلاً حرکت به اتاق غیرمجاور) و هالوسینیشن درباره وضعیت بازی (مثلاً فرض وجود بمبی که نیست یا مرحله ای که دیده نشده).

یک محیط/بازی مشابه که بتوان آزمایش را تکرار کرد چیست؟

یک محیط مشابه می تواند هر سناریوی جست وجو و نجات متنی یا شبیه سازی شده ای باشد که در آن:
جهان به اتاق ها/نواحی گراف مانند تقسیم شده،

اطلاعات و ابزار بین عامل ها توزیع شده،
موفقیت نیاز به توالی درست اعمال و هماهنگی بین اعضای تیم دارد.
برای مثال می توان یک «ماموریت نجات گروگان» متنی طراحی کرد که در آن چند عامل باید در یک ساختمان، اتاق ها را جست وجو کنند، کلیدها و ابزارها را تقسیم کنند، درها را باز کنند و گروگان ها را نجات دهند؛ هر عامل فقط اتاق فعلی خود را می بیند و از طریق چت متنی با بقیه هماهنگ می شود، دقیقاً مشابه ساختار گرافی و ارتباطی این مقاله. یک گزینه دیگر بسط همین سناریو بمب خنثی کنی به تعداد بیشتر اتاق ها و بمب ها است، یا پیاده سازی آن در یک محیط متنی مانند TextWorld یا ALFWorld با قوانین مشابه.

دو شکل مقاله چه چیزی را نشان می دهند؟

شکل اصلی فریم ورک :(Figure 1) این شکل معماری کلی سیستم را نشان می دهد؛ در سمت راست «Task Environment»(محیط بمب ها و اتاق ها) قرار دارد، که از طریق «Text Game Interface» به توصیف های متنی و بازخورد درباره اعمال تبدیل می شود؛ در سمت چپ سه عامل(Alpha) LLM ، Bravo، (Charlie دیده می شوند که هر کدام با مدل زبانی، belief stateمتنی و تاریخچه ی تعامل خود کار می کنند و در هر راند بر اساسobservation ، belief stateرا به روزرسانی کرده، اکشن و پیام به تیم تولید می کنند. این شکل جریان حلقه ای: محیط ← مشاهده متنی ← به روزرسانی باور ← انتخاب عمل و پیام ← رمزگشایی به عمل ← به روزرسانی محیط را به صورت شماتیک نمایش می دهد.
شکل رفتار و خطاها :(Figure 2) این شکل چند پنل مثالی از لاگ تعامل را نشان می دهد؛ در پنل بالا-چپ نمونه ای از شکست در مدیریت کانتکست بلندمدت (درخواست حرکت به اتاق غیرمجاور و خطای محیط) آمده است، در بالا-راست نمونه ای از رفتارهای تعاملی مثبت مثل رهبری emergent و تقسیم وظایف در پیام هایAlpha ، Bravoو Charlie را می بینیم، و در پنل های پایین، پاسخ های GPT-4+Belief و ChatGPT به پرسش های ToM (introspection، first-order، second-order) کنار هم گذاشته شده تا نشان دهد GPT-4+Belief چطور با توجه به تاریخچه ی موقعیت و پیام ها، استنتاج دقیق تری از «چه کسی چه چیزی می داند» انجام می دهد.

سوال 3.

هدف کلی این مقاله چیست؟

هدف کلی مقاله این است که نشان دهد می توان با استفاده ازLLM های از پیش آموزش دیده و چند مثال زنجیره استدلال(chain-of-thought) ، آن ها را به عامل هایی تبدیل کرد که در بازی های استراتژیک بتوانند به صورت انعطاف پذیر و قابل تعمیم، درباره ی وضعیت ها، پاداش ها و باورهای دیگران استدلال کنند، بدون این که نیاز به آموزش مجدد روی هر بازی جدید باشد. نویسندگان می خواهند نشان دهند که با یک «prompt compiler» ساختاریافته، LLMقادر است در انواع بازی های ماتریسی و مذاکره، استراتژی های نزدیک به نظریه ی بازی و انسان مانند تولید کند.

عامل ها چه هستند و محیط بازی چگونه است؟

در این کار عامل ها همانLLM هایی هستند) مثل code-davinci-002 و نسخه های متنی (davinci/curie که نقش بازیکن را در بازی های استراتژیک بر عهده می گیرند و با تکیه بر پرامپت های حاوی دموهای استدلال، حرکت بعدی خود را انتخاب می کنند. دو نوع محیط اصلی تعریف می شود:

بازی های ماتریسی :(Matrix Games) بازی های ۲×۲ و بزرگ تر (۳×۳، ۳×۴، چندبازیکنه) مثل Prisoner's Dilemma, Chicken, Stag Hunt, Matching Pennies و نسخه های دومرحله ای، که در آن ها هر بازیکن یک اکشن از مجموعه اکشن های محدود انتخاب می کند و ترکیب اکشن ها یک پاداش عددی برای هر بازیکن تولید می کند.

بازی های مذاکره :(Deal or No Deal) سناریوهایی که دو بازیکن باید یک «گلدان» شامل چند نوع آیتم (کلاه، کتاب، توپ) را بین خود تقسیم کنند، در حالی که هر کدام ارزش متفاوتی برای هر آیتم دارند و از ارزش طرف مقابل خبر ندارند؛ در برخی آزمایش ها خود مدل نقش «broker» (داور) را دارد و باید یک تقسیم «منصفانه» پیشنهاد دهد.

عامل ها چگونه با هم ارتباط برقرار می کنند؟

در بیشتر آزمایش ها، ارتباط بین عامل ها ضمنی است و از طریق ساختار بازی و مدل سازی ذهنی طرف مقابل انجام می شود؛ یعنی LLM در استدلال متنی خود در نظر می گیرد که «حریف چه هدفی دارد و چه اکشنی خواهد زد» و بر همان اساس بهترین پاسخ را انتخاب می کند (level-k / Theory-of-Mind) سبک .(در محیط مذاکره، ارتباط به صورت دیالوگ رفت وبرگشتی است: انسان یا عامل مقابل یک پیشنهاد متنی برای تقسیم آیتم ها می دهد، LLMبا استدلال متنی درباره ارزش ها و باورها پاسخ می دهد(accept/reject/propose) ، و این رفت وبرگشت در چند دور ادامه می یابد؛ در سناریوی «broker» هم LLM براساس اطلاعات دو طرف، پیشنهاد متن محورِ جدید تولید می کند.

آیا عامل ها با هم همکاری می کنند؟ چگونه و چرا؟

این مقاله هم سناریوهای رقابتی و هم تعاونی را پوشش می دهد، اما در هر دو حالت نوعی «استدلال متقابل» بین عامل ها وجود دارد که به همکاری یا سازش منجر می شود. در برخی بازی های ماتریسی مثل ,welfare» «welfareیا وقتی هدف «کمینه کردن تفاوت پاداش ها «(equality) است، LLMعملاً به دنبال حداکثرسازی رفاه

مشترک یا تقسیم منصفانه پاداش است، که نوعی همکاری است. در بازی مذاکره ی واقعی هم نتایج نشان می دهد که عامل نه تنها پاداش خود را بالا می برد، بلکه پاداش انسان را هم کمی بهتر از baseline می کند و در ارزیابی کاربران «منطقی، سازش گر و شبیه انسان» دیده می شود، یعنی به طور ضمنی همکاری/سازش را یاد می گیرد تا به توافق برسد.

منظور از THEORY OF MIND در این مقاله چیست؟

هرچند مقاله مستقیماً روی واژه ی «Theory of Mind» تمرکز ندارد، ولی مفهوم آن را در قالب سه مؤلفه ی استراتژیک پیاده می کند:

Search: جست وجو در درخت بازی با فرض هایی درباره ی این که حریف چه هدفی دارد و چه اکشنی انتخاب می کند.

Value Assignment: این که مدل بفهمد خودِ بازیکن و حریف چه پاداشی از هر حالت می گیرند (مثلاً حریف می خواهد «max» خودش را زیاد کند یا «کمک» کند).

Belief Tracking: نگه داشتن باور روی اطلاعات پنهان مثل حالت جهان یا ترجیحات حریف، مثلاً از روی این که «حریف ۹ بار از ۱۰ بار کتاب را به توپ ترجیح داده، پس کتاب را بیشتر دوست دارد.«

در عمل، این یعنی مدل تلاش می کند باورهای حریف و اهدافش را از روی رفتار و پاداش ها استنتاج کند و بر اساس آن استراتژی خود را تنظیم کند، که جوهره ی Theory of Mind در محیط های استراتژیک است.

حافظه در این مقاله چگونه مدیریت و نگهداری می شود؟

حافظه در این کار بیشتر در قالب «استدلال متنی درون پرامپت» است:

یک «prompt compiler» برای هر بازی یک دمو تولید می کند که در آن مراحل جست وجو، محاسبه ی امیدریاضی پاداش ها، مقایسه ی اکشن ها و به روزکردن باورها روی حالت های پنهان یا ارزش ها به صورت متن chain-of-thought نوشته شده است.

هنگام حل بازی جدید، LLM همین الگوی متنی را روی داده های جدید تکرار می کند: برای هر حالت/اکشن پاداش ها را حساب می کند، روی hidden states یا ترجیحات حریف belief به روز می کند و در چند دور «propose → evaluate → revise» (به خصوص در مذاکره) پیشنهاد را اصلاح می کند؛ این متن استدلالی کوتاه مدت حافظه ی کاری عامل است.

در تنظیمات پیچیده تر، از «factored reasoning» و ابزارهایی مثل search(...) و mean(...) به صورت نمادین در متن استفاده می شود تا استدلال روی زیرمسئله ها و زیر درخت ها شکسته شود و مشکل محدودیت context کمتر شود.

نتایج اصلی مقاله چیست؟

در بازی های ماتریسی با پاداش های جدید، LLM با پرامپت استراتژیک توانست در تقریباً همه ی موارد بهترین پاسخ نظریه ی بازی را پیدا کند (دقیقاً ۱٫۰۰ دقت در بسیاری از تنظیمات)، در حالی که ۰-shot، ۲ shot-معمولی و CoT ساده عملکرد خیلی ضعیف تری داشتند.

در بازی هایی با قوانین پیچیده تر (درخت های دومرحله ای، ماتریس های بزرگ تر، چندبازیکنه)، استفاده از «factored search» باعث شد مدل بتواند به این ساختارهای جدید بدون مثال اضافی تعمیم بدهد و با دقت بالا

بهترین پاسخ را انتخاب کند، در حالی که بدون فاکتورکردن، دقت به خاطر محدودیت context و پیچیدگی افت می کرد.

در سناریوهای با اهداف جدید (مثلاً max/help ، welfare، daxity) ، مدل فقط با دیدن دموهایی که هدف را توضیح می دهند و بدون آموزش مجدد توانست در اکثر موارد با دقت نزدیک ۱/۰۰ بهترین پاسخ را برای هدف جدید انتخاب کند، که نشان دهنده ی تعمیم به «اهداف انتزاعی» است.

در نقش «broker» درDeal or No Deal ، عامل استدلالی توانست تفاوت با تقسیمِ ایده آل عادلانه را بیش از دو برابر نسبت به baseline کاهش دهد (یعنی پیشنهادات بسیار نزدیک به optimum equality یا Rawlsian fairness تولید کند).

در نقش مذاکره کننده ی مستقیم با انسان ها، روش پیشنهادی هم پاداش خود و هم پاداش انسان را کمی بالا برد و در ارزیابی کاربران، از نظر «شبیه انسان بودن، معقول بودن، سازش گر و کمتر تهاجمی بودن» از baselines بهتر نمره گرفت، یعنی هم از نظر کارایی و هم از نظر کیفیت اجتماعی رفتار، برتر بود.

یک محیط/بازی مشابه که بتوان آزمایش را در آن تکرار کرد چیست؟

هر محیط نظریه ی بازی که با چند بازیکن، جدول پاداش و شاید اطلاعات پنهان تعریف می شود، مناسب تکرار این سبک آزمایش است؛ مثلاً:

نسخه ای از بازی «Diplomacy» یا «Hanabi» که به جای اجرای کامل، موقعیت ها به صورت توصیف متنی وpayoff های ساده شده به مدل داده شود تا بهترین حرکت یا مذاکره را انتخاب کند.

یک بازار ساده دو یا سه کالایی که چند عامل باید قیمت گذاری یا پیشنهاد خرید/فروش بدهند؛ LLM می تواند با همان ساختار search / value / belief استراتژی قیمت یا پیشنهاد را پیدا کند.

سناریوهای ساده ی مزایده (auction) که در آن چند bidder با ارزش های پنهان در یک حراج تک شیء یا چندشیء شرکت می کنند و مدل باید استراتژی پیشنهاد را انتخاب کند.

نکته ی اصلی این است که بتوان محیط را به صورت «درخت یا ماتریس پاداش» و چند متن دمو برای جست وجو و محاسبه ی ارزش ها به LLM داد تا همان روش روی آن اعمال شود.

دو شکل مقاله چه چیزی را نشان می دهند؟

شکل ۱: در سمت چپ یک درخت بازی جزئی مشاهده ای با دو بازیکن Gopher) و (Bob و حالت های پنهان s_1, s_2 و اکشن های a_1, a_2, b_1, b_2 نشان داده شده که در برگ های آن پاداش های دو بازیکن نوشته شده است؛ در سمت راست، یک نمونه از استدلال متنی مرتبط آورده شده که در آن با رنگ های مختلف سه جزء) «search»جست وجوی شاخه ها(، «value»(محاسبه ی امیدریاضی پاداش ها) و «belief» (شکل دادن باور روی حالت جهان از روی اقدام حریف) هایلایت شده تا نشان دهد prompt compiler چگونه-chain-of thoughtرا ساختار می دهد.

شکل ۳ (برای Deal or No Deal): پنل (a) یک اپیزود نمونه را نشان می دهد که در آن تعداد آیتم ها و ارزش های Alice و Bob نمایش داده شده و پیشنهاد اولیه حریف مشخص است؛ پنل (b) نمونه ای از استدلال متنی مربوط به همین اپیزود را نشان می دهد که در آن با رنگ صورتی بخشی که روی «ارزش ها» حساب می کند، با نارنجی «جست وجوی پیشنهادهای جایگزین» و با خاکستری «به روزرسانی باورها درباره ترجیحات حریف» مشخص شده است. این شکل ها کمک می کنند بصری ببینیم که سه مؤلفه ی search / value / belief چگونه داخل متن زنجیره استدلالی جاگذاری شده اند.