

Lecture 20: Tool Use and Language Agents

[Language Models & Tools](#)

[LMs are powerful for text generation tasks. But ...](#)

[Tools benefit language models a lot](#)

[What Is A Tool Anyway?](#)

[Tool Basics: Definition](#)

[Tool Basics: Functionality](#)

[The Basic Tool Use Paradigm](#)

[Scenarios of LM Tool Using](#)

[What if tools are unavailable?](#)

[How does TroVE make tools?](#)

[How Can TroVE Help?](#)

[How to evaluate tool use?](#)

[Trade-offs in tool usage: Computation Cost](#)

[In Summary](#)

[Language Model as Agents](#)

[What are agents?](#)

[How to get started in LLM Agents?](#)

[Tasks and Applications for LLM Agents](#)

[Why do we want agents?](#)

[How do People Interact with Computers?](#)

[Tool Integrations into Chatbots](#)

[Robots](#)

[Games](#)

[Software Development](#)

[UI Automation](#)

[Training-free Methods for Building Agents](#)

[Quan sát từ môi trường](#)

[Khả năng lập kế hoạch và suy luận](#)

[Khả năng sử dụng công cụ](#)

[Tạo mã để thực thi nhiệm vụ](#)

[Evaluation Environment and Benchmark](#)

[Evaluation of LLM Agents](#)

[Keys to Agent Benchmarks](#)

[WebArena Environment Design](#)

[Collecting Realistic Intents](#)

[Example Tasks in WebArena](#)

[Outcome/ Execution-based Evaluation](#)

[Observation & Action Space](#)

[Prompting LLM as Agent](#)
[WebArena is Challenging](#)
[Failures: Not Knowing How](#)
[Failures: Not Being Accurate](#)
[Failures: Trivial Errors](#)
[Training Methods for Improving Agents](#)
[Learning of LLM Agents](#)
[In-context Learning](#)
[Supervised Fine-tuning](#)
[Reinforcement Learning](#)
[Resources](#)

Language Models & Tools

LMs are powerful for text generation tasks. But ...

Mô hình ngôn ngữ hiện nay rất mạnh mẽ trong việc giải quyết nhiều nhiệm vụ, đặc biệt là trong việc tạo ra văn bản. Tuy nhiên, câu hỏi đặt ra là liệu mô hình ngôn ngữ có đủ khả năng để xử lý mọi tình huống hay không. Theo quan điểm cá nhân của tôi, câu trả lời là không.

Có một số tình huống mà mô hình ngôn ngữ không hoạt động hiệu quả. Đầu tiên, khi được yêu cầu thực hiện các phép toán phức tạp, mô hình ngôn ngữ có thể không thực hiện một cách chính xác hoặc hiệu quả. Ví dụ, phương pháp phổ biến là sử dụng "Chain of Thought" để xử lý chi tiết, nhưng điều này có thể dẫn đến việc không đưa ra được câu trả lời đúng. Ngược lại, nếu bạn sử dụng một công cụ máy tính, bạn có thể nhập trực tiếp biểu thức và nhận được kết quả ngay lập tức.

Một ví dụ khác là khi cần truy cập thông tin thực tế mà mô hình ngôn ngữ không có sẵn. Chẳng hạn, nếu bạn hỏi "What is the current time?", thời gian hiện tại có thể không nằm trong dữ liệu huấn luyện của mô hình. Tùy thuộc vào thời điểm bạn đặt câu hỏi, câu trả lời sẽ khác nhau. Do đó, cách duy nhất để mô hình ngôn ngữ trả lời câu hỏi này là sử dụng một công cụ bên ngoài, chẳng hạn như gọi API "get time" để lấy thời gian hiện tại.

Những hạn chế này đã thúc đẩy sự quan tâm của mọi người đối với việc sử dụng các công cụ bổ sung để nâng cao khả năng của mô hình ngôn ngữ.

Tools benefit language models a lot

Trong lĩnh vực nghiên cứu về mô hình ngôn ngữ, có nhiều công trình tập trung vào việc tích hợp công cụ (tool) với các mô hình này. Một trong những nghiên cứu tiên phong là "toolformer", đề xuất sử dụng 5 công cụ như máy tính và công cụ tìm kiếm Wiki để mở rộng khả năng của mô hình ngôn ngữ.

Tuy nhiên, các nghiên cứu sau đó đã sử dụng và đánh giá nhiều loại công cụ khác nhau trên các bộ dữ liệu đa dạng:

- Công cụ phần mềm: Như trong "toolformer" và "ART", sử dụng máy tính và công cụ tìm kiếm Wiki.
- API: Một số nghiên cứu sử dụng các API web như "get weather" hoặc "get time".
- Mô hình neural: Có công trình sử dụng các mô hình từ Hugging Face Hub làm công cụ, bao gồm cả mô hình ngôn ngữ và các mô hình chuyên biệt cho từng tác vụ.
- Hàm tự định nghĩa: Một số nghiên cứu tập trung vào việc sử dụng các hàm được định nghĩa cục bộ và được thiết kế bởi chuyên gia làm công cụ.

Sự đa dạng này trong việc định nghĩa và sử dụng "công cụ" có thể gây ra một số nhầm lẫn.

What Is A Tool Anyway?

Trong phần này, chúng ta sẽ khám phá ba khía cạnh chính liên quan đến công cụ. Đầu tiên, chúng ta sẽ tìm hiểu về khái niệm cơ bản của công cụ, định nghĩa và các chức năng chính của chúng. Tiếp theo, chúng ta sẽ xem xét các kịch bản sử dụng công cụ, bao gồm những công cụ nào hiện có, nhiệm vụ nào có thể áp dụng công cụ này và phương pháp sử dụng. Cuối cùng, chúng ta sẽ thảo luận về khía cạnh đánh giá, bao gồm các tiêu chí đánh giá và lợi ích thực nghiệm, nếu có, của các công cụ này.

Tool Basics: Definition

Theo định nghĩa, công cụ là một chương trình mà mô hình ngôn ngữ có thể sử dụng để thực hiện một số chức năng nhất định. Để một chương trình được coi là công cụ, nó cần thỏa mãn hai thuộc tính: tính ngoại vi và tính chức năng.

Thứ nhất, tính ngoại vi có nghĩa là công cụ phải nằm ngoài mô hình ngôn ngữ. Điều này tương tự như định nghĩa về việc sử dụng công cụ của động vật, nơi công cụ là một đối tượng môi trường được sử dụng từ bên ngoài.

Thứ hai, tính chức năng đề cập đến khả năng của chương trình có thể áp dụng lên các đối tượng khác trong môi trường và thay đổi trạng thái của môi trường hoặc tạo ra một kết quả đầu ra. Ví dụ đơn giản là trong một môi trường có một tấm vải trắng (canvas) và một công cụ là cọ vẽ. Chức năng của cọ vẽ là tô màu lên tấm vải và tạo ra một bức tranh.

Kết hợp hai thuộc tính này, chúng ta có thể định nghĩa công cụ là một giao diện hàm cho chương trình máy tính hoạt động bên ngoài mô hình ngôn ngữ. Mô hình ngôn ngữ sử dụng công cụ bằng cách tạo ra các lệnh hàm và các tham số đầu vào cho công cụ đó.

Tool Basics: Functionality

Sau khi hiểu rõ định nghĩa, chúng ta có thể tóm tắt ba chức năng chính của các công cụ:

1. Công Cụ Nhận Thức: Được sử dụng để thu thập dữ liệu từ môi trường mà không thay đổi dữ liệu đó. Ví dụ, công cụ tìm kiếm có thể được coi là công cụ nhận thức khi nó tìm kiếm và thu thập thông tin liên quan đến truy vấn của bạn. Một ví dụ khác là API thời tiết, nơi bạn có thể lấy thông tin thời tiết từ web.
2. Công Cụ Hành Động: Được sử dụng để thực hiện hành động trong môi trường và thay đổi trạng thái của nó. Ví dụ, khi bạn coi một bức tranh là một đối tượng trong môi trường, việc sử dụng cọ vẽ để thay đổi bức tranh là một hành động thay đổi trạng thái môi trường.
3. Công Cụ Tính Toán: Thực hiện các hoạt động tính toán, không chỉ giới hạn ở các phép toán số học. Ví dụ, một công cụ dịch thuật có thể được coi là công cụ tính toán khi nó chuyển đổi ngôn ngữ từ ngôn ngữ này sang ngôn ngữ khác.

Các chức năng này không hoàn toàn tách biệt; một công cụ có thể có nhiều chức năng. Ví dụ, công cụ tìm kiếm Wikipedia có thể vừa là công cụ nhận thức khi thu thập tài liệu từ web, vừa là công cụ tính toán khi tính toán điểm tương đồng của truy vấn với các tài liệu khác và xếp hạng chúng.

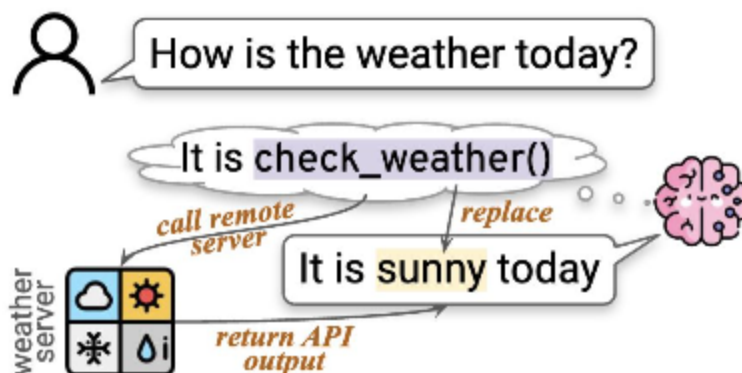
Agent có thể sử dụng công cụ nhận thức để thu thập thông tin từ môi trường và công cụ hành động để thực hiện hành động trong môi trường. Tuy nhiên, một mô hình ngôn ngữ chỉ sử dụng công cụ tính toán mà không sử dụng công cụ nhận thức hay hành động thì không được coi là agent theo định nghĩa này.

Hiện tại, việc sử dụng công cụ trong xây dựng agent chưa thực sự phát triển. Các bộ dữ liệu chủ yếu hỗ trợ việc chia nhỏ nhiệm vụ thành nhiều bước, mỗi bước chỉ sử dụng một công cụ. Tuy nhiên, đây có thể là một hướng nghiên cứu thú vị trong tương lai.

The Basic Tool Use Paradigm

Trong phần này, chúng ta sẽ đi sâu vào chi tiết về các kịch bản và nhiệm vụ mà mô hình ngôn ngữ có thể áp dụng. Chúng ta bắt đầu với công cụ cơ bản sử dụng Paradigm, cho phép mô hình ngôn ngữ sử dụng công cụ một cách hiệu quả.

Tóm lại, đây là sự chuyển đổi giữa chế độ tạo văn bản và chế độ thực thi công cụ. Ví dụ, khi người dùng hỏi "Thời tiết hôm nay thế nào?", mô hình ngôn ngữ bắt đầu với quá trình tạo văn bản tiêu chuẩn. Khi mô hình cảm thấy cần sự hỗ trợ từ công cụ, nó sẽ tạo ra các token để hình thành biểu thức gọi công cụ, chẳng hạn như "kiểm tra thời tiết". Sau khi biểu thức này hoàn thành, nó sẽ kích hoạt máy chủ thực thi từ xa, ở đây là máy chủ thời tiết, và chuyển sang chế độ thực thi công cụ. Máy chủ sẽ thực thi lệnh gọi và trả về kết quả, ví dụ như "trời nắng", và trả lại cho mô hình ngôn ngữ.



Mô hình ngôn ngữ sau đó thay thế lệnh gọi API bằng kết quả thực thi và chuyển lại sang chế độ tạo văn bản để tiếp tục tạo các token còn lại. Sau khi hoàn tất quá trình này, mô hình sẽ trả về phản hồi cuối cùng cho người dùng ban đầu. Quá trình này khá trực quan và phương pháp để dạy mô hình sử dụng công cụ cũng khá đơn giản.

Có hai cách tiếp cận chính: Thứ nhất là suy luận trong thời gian thực, cung cấp hướng dẫn ngôn ngữ tự nhiên và ví dụ trong ngữ cảnh để hướng dẫn mô hình thực hiện quá trình này. Thứ hai là học qua đào tạo, nơi mô hình được huấn luyện với các ví dụ về truy vấn ngôn ngữ tự nhiên và giải pháp công cụ tương ứng.

Scenarios of LM Tool Using

Việc sử dụng các công cụ hỗ trợ đã được nhiều người áp dụng trong nhiều tình huống khác nhau. Dưới đây là năm kịch bản chính:

1. Truy cập tri thức: Mục tiêu của việc này là giải quyết vấn đề về giới hạn tri thức mà các mô hình có thể ghi nhớ hoặc lưu trữ trong quá trình huấn luyện. Ví dụ, thời gian hiện tại có thể không được chia sẻ trong dữ liệu huấn luyện. Các mô hình có thể truy cập tri thức từ các nguồn khác nhau như cơ sở tri thức có cấu trúc với đồ thị tri thức. Người dùng có thể sử dụng các công cụ như SQL executor để truy vấn dữ liệu có cấu trúc và nhận kết quả cuối cùng. Ngoài ra, các công cụ tìm kiếm cũng được sử dụng để tìm kiếm thông tin trên Internet. Các mô hình truy xuất có thể được xem như là một cách để truy cập tri thức.
2. Hoạt động tính toán: Điều này nhằm giải quyết vấn đề mà các mô hình không thể thực hiện các phép tính phức tạp một cách hiệu quả. Ví dụ, người dùng có thể sử dụng máy tính để giải các bài toán toán học hoặc sử dụng trình thông dịch Python để thực hiện các chương trình phức tạp hơn. Ngoài ra, có thể tận dụng các phần mềm hiện có như Google Sheets để thực hiện các hành động cần thiết.
3. Tương tác với thế giới thực: Khả năng của các mô hình ngôn ngữ bị giới hạn trong dữ liệu huấn luyện. Để mở rộng khả năng này, các mô hình có thể được sử dụng để điều hướng web hoặc truy cập thông tin thực tế như thời tiết hoặc vị trí hiện tại. Chúng cũng có thể quản lý lịch và email để tự động hóa công việc.

4. Xử lý các loại dữ liệu phi văn bản: Các mô hình ngôn ngữ chủ yếu xử lý văn bản, nhưng nếu được kết nối với các công cụ khác, chúng có thể truy cập và tương tác với các loại dữ liệu khác. Ví dụ, API như TAD Image cho phép truy cập và xử lý hình ảnh, hoặc Spotify API để nghe nhạc. Các công cụ như Visual QA có thể được sử dụng để trả lời câu hỏi về dữ liệu hình ảnh.
5. Sử dụng các mô hình chuyên biệt: Người dùng có thể tải các mô hình QA hoặc dịch thuật chuyên biệt để thực hiện các tác vụ cụ thể. Ví dụ, mô hình Visual QA có thể được sử dụng để trả lời câu hỏi về dữ liệu hình ảnh.

Một điểm chung của các công cụ này là chúng đều được thiết kế bởi các chuyên gia trước khi được áp dụng vào các tác vụ cụ thể. Với sự hỗ trợ của các công cụ này, chúng ta có thể dễ dàng áp dụng chúng vào các nhiệm vụ mà không cần phải thiết kế lại từ đầu. Tuy nhiên, cũng có những nhiệm vụ không có sẵn các công cụ được thiết kế trước.

What if tools are unavailable?

Một câu hỏi thú vị đặt ra là liệu chúng ta có thể tự động tạo ra các công cụ, chẳng hạn như sử dụng các mô hình ngôn ngữ, mà không cần phụ thuộc vào các chuyên gia con người hay không. Trong nghiên cứu của bài báo "TroVE: Inducing Verifiable and Efficient Toolboxes for Solving Programmatic Tasks" của Wang và cộng sự (2024), thì câu trả lời là có.

Để tóm tắt một cách ngắn gọn, phương pháp tiêu chuẩn để giải quyết các bài toán lập trình thường là bạn sẽ được cung cấp một bài toán bằng ngôn ngữ lập trình và yêu cầu mô hình ngôn ngữ tạo ra một chương trình. Sau đó, bạn sẽ thực thi kết quả để có được câu trả lời cuối cùng. Một cách tương tự, bạn có thể có một chuỗi các ví dụ văn bản và truyền chúng vào mô hình ngôn ngữ, từ đó mô hình sẽ tạo ra các giải pháp cho từng bài toán. Tuy nhiên, các chương trình này thường có độ dài khá lớn và có thể gặp phải một số vấn đề.

Một trong những vấn đề chính là khả năng xảy ra lỗi lập trình. Ví dụ, chỉ cần gõ sai một ký tự cũng có thể khiến toàn bộ giải pháp chương trình trở nên sai lệch. Do đó, động lực của chúng tôi là: nếu chúng ta yêu cầu các mô hình ngôn ngữ không chỉ tạo ra chương trình mà còn tạo ra một bộ công cụ hỗ trợ, thì việc giải quyết bài toán sẽ trở nên đơn giản hơn rất nhiều.

Chẳng hạn, nếu có một công cụ tính toán tỷ lệ thay đổi, thì giải pháp chỉ cần là một lệnh gọi hàm, và điều bạn cần làm chỉ là xác định các tham số đầu vào cho hàm đó. Điều này không chỉ giúp giảm thiểu các lỗi mà chúng ta đã thấy, mà còn mang lại lợi ích cho con người, vì việc xác minh giải pháp sẽ trở nên dễ dàng hơn.

How does TroVE make tools?

Trong nghiên cứu bài báo TroVE của Wang và cộng sự (2024), phương pháp TroVE gồm ba bước chính: Create mode, Import mode, và Skip mode.

1. Create mode:

- Đầu tiên, họ bắt đầu bằng cách khởi tạo và chuẩn bị các ví dụ. Khi gặp một ví dụ mới có tính năng chưa thấy trước đây, mô hình sẽ tạo ra một công cụ có thể tái sử dụng.
 - Sau đó, mô hình sử dụng công cụ vừa tạo để sinh ra giải pháp cho ví dụ đó.
 - Để cải thiện chất lượng của các cặp (ví dụ và giải pháp), họ thực hiện thao tác sampling và chỉ chọn các cặp tốt nhất. Giải pháp cùng với công cụ được thêm vào hộp công cụ (toolbox) để sử dụng cho các ví dụ sau này.
2. Import mode:
 - Giả sử trong hộp công cụ đã có nhiều chức năng. Khi mô hình thấy một ví dụ sử dụng các chức năng tương tự, thay vì tạo lại công cụ mới, mô hình sẽ trực tiếp nhập công cụ từ hộp công cụ.
 - Như vậy, mô hình sẽ tái sử dụng công cụ và sinh ra giải pháp cho ví dụ mới, phù hợp với chế độ Import mode.
 3. Skip mode:
 - Trong trường hợp bài toán quá dễ, mô hình có thể bỏ qua việc tạo công cụ mới và trực tiếp đưa ra giải pháp mà không cần công cụ.

Ví dụ: Khi gặp ví dụ mới với chức năng chưa thấy trước đây, mô hình sẽ tạo ra công cụ mới và sử dụng công cụ đó để giải quyết bài toán. Tuy nhiên, nếu chức năng này đã tồn tại trong hộp công cụ, mô hình chỉ cần nhập và sử dụng lại công cụ đó, giúp tiết kiệm thời gian và tài nguyên. Và trong các trường hợp đặc biệt, mô hình có thể bỏ qua việc sử dụng công cụ và trực tiếp giải quyết bài toán.

Phương pháp này cho phép tạo ra các công cụ chính xác và có thể tái sử dụng, đồng thời cải thiện hiệu suất giải quyết các nhiệm vụ lập trình một cách đáng kể.

How Can TroVE Help?

Kết quả của phương pháp TroVE được Wang và cộng sự (2024) công bố thật ấn tượng. Sau đây là tổng quan về kết quả nghiên cứu và phương pháp:

Accuracy ↑						Complexity ↓						Verification ↑					
Method	MATH _{algebra}		TabMWP		GQA		prealg	precal	TABLEQA			VISUAL					
	acc ↑	# lib ↓	acc ↑	# lib ↓	acc ↑	# lib ↓			TabMWP	WTQ	HiTab		GQA				
w/ additional supervision																	
LATM	0.30	-	0.09	-	0.29	-											
CRAFT	0.68	282	0.88	181	0.45												
w/ additional rectification & iteration																	
Creator	0.65	875	0.81	4,595	0.34												
w/o supervision, rectification, or iteration																	
TroVE	0.72	16	0.92	38	0.44												

Method	Accuracy ↑		Time (s) ↓		31-43% faster
	avg	std	avg	std	
	0.77	0.109	25.5		
	0.88	0.024	30.7		
	0.87	0.057	17.5		

Table 3. Comparing with existing methods using GPT-4. We report the baseline results as reported in Yuan et al. (2023). We do not report the complexity metric since none of these methods report it (our results in Table 2).

Method	Accuracy ↑		Time (s) ↓	
	avg	std	avg	std
10% more accurate	0.77	0.109	25.5	
	0.88	0.024	30.7	
	0.87	0.057	17.5	
			31-43% faster	

Table 5. Human accuracy and time in verifying model-produced solutions with three methods experimented.

1 MATH, TABLEQA, and VISUAL tasks.

- Cải thiện độ chính xác và hiệu suất: Phương pháp TroVE giúp cải thiện đáng kể độ chính xác so với các phương pháp cơ sở (Baselines). Mặc dù đạt được độ chính xác cao hơn, kích thước thư viện của phương pháp vẫn duy trì ở mức nhỏ, giúp giảm thiểu tài nguyên sử dụng.
- Giảm độ phức tạp của giải pháp: Số lượng thao tác, đại diện cho độ phức tạp của giải pháp, của phương pháp TroVE được đo lường và so sánh. Kết quả cho thấy, giải pháp của phương pháp này đơn giản hơn nhiều so với các phương pháp khác.
- Nghiên cứu xác minh bởi con người: Nhóm nghiên cứu đã thực hiện một nghiên cứu thú vị về xác minh bởi con người. Trong nghiên cứu này, con người được yêu cầu xác minh tính đúng đắn của giải pháp và đo lường thời gian xác minh. Kết quả cho thấy phương pháp TroVE giúp quá trình xác minh chính xác hơn 10% và nhanh hơn từ 30 đến 40%.

How to evaluate tool use?

Trong phần này, chúng ta sẽ thảo luận về cách đánh giá và những lợi ích thực nghiệm của việc sử dụng các công cụ. Hiện tại, chúng ta đang sử dụng hai loại chuẩn mực để đánh giá:

1. Tái sử dụng các chuẩn mực hiện có: Các chuẩn mực này thường được áp dụng cho các tác vụ yêu cầu lý luận, chẳng hạn như các tập dữ liệu toán học như "math dataset" hoặc "big bench dataset". Ngoài ra, còn có các tác vụ phức tạp hơn yêu cầu dữ liệu có cấu trúc như bảng hoặc cơ sở tri thức, với các tập dữ liệu như "table, KG" cho các bảng phức tạp hơn. Cuối cùng, có những tác vụ liên quan đến các phương thức khác như "visual QA", nơi người dùng có thể tạo ra hoặc tái sử dụng các công cụ lập trình để thực hiện các câu hỏi về hình ảnh hoặc chỉnh sửa hình ảnh.
2. Chuẩn mực API tổng hợp: Những chuẩn mực này chủ yếu tập trung vào các công cụ API. Quá trình tạo ra các chuẩn mực này thường giống nhau: tìm kiếm các trang web cung cấp nhiều API công cộng, sau đó lập trình API và thu thập siêu dữ liệu liên quan. Tuy nhiên, khi phân tích các tập dữ liệu này, chúng tôi phát hiện hai vấn đề chính:
 - Vấn đề tự nhiên: Khi xem xét sâu hơn về cách các ví dụ sử dụng API được chọn, chúng tôi nhận thấy rằng các công cụ được chọn có thể không được sử dụng cùng nhau trong thực tế. Hơn nữa, các ví dụ được tạo ra có thể không phản ánh đúng cách sử dụng tự nhiên của các công cụ này.
 - Khả năng thực thi của công cụ: Mặc dù công cụ được coi là các chương trình có thể thực thi, nhưng thực tế không phải lúc nào cũng vậy. Hơn một nửa các tập dữ liệu mà chúng tôi phân tích cho thấy các công cụ không thể thực thi. Điều này có thể liên quan đến cách mà các ví dụ được tạo ra, khi mà các ví dụ thường chỉ được tổng hợp mà không thực thi thực tế.

Để đánh giá các tập dữ liệu này, hiện tại chúng ta sử dụng một số tiêu chí cơ bản:

1. Tỷ lệ hoàn thành tác vụ: So sánh phản hồi do mô hình tạo ra với phản hồi tham chiếu đã được chú thích.

2. So sánh lựa chọn: Đối với những tác vụ mà công cụ không thể thực thi, ta so sánh các lựa chọn hoặc biểu thức để kiểm tra độ chính xác của phản hồi.
3. Khả năng sử dụng công cụ: Đánh giá khả năng sử dụng của công cụ để khuyến khích tính năng tổng quát và hiệu quả trong thực tiễn.

Tuy nhiên, vẫn còn nhiều khía cạnh quan trọng chưa được đề cập trong các tiêu chí đánh giá này. Một số khía cạnh quan trọng có thể bao gồm:

- Hiệu suất: So sánh chi phí tính toán khi sử dụng các công cụ.
- Chất lượng công cụ: Đánh giá tốc độ phản hồi và hiệu quả tính toán của công cụ.
- Độ tin cậy: Đặc biệt là đối với các công cụ không ổn định, cần có cách để người dùng nhận biết độ chính xác của công cụ.
- Kiểm tra tái lập: Cần có các giải pháp tham chiếu có thể chạy song song với các giải pháp do mô hình tạo ra.
- Bảo mật thông tin: Đánh giá mức độ tin cậy của các công cụ API, đặc biệt khi người dùng cần cung cấp thông tin cá nhân.

Những khía cạnh này đều là những vấn đề thú vị nhưng chưa được nghiên cứu sâu trong lĩnh vực công cụ hiện nay.

Trade-offs in tool usage: Computation Cost

Trong nghiên cứu này, các nhà nghiên cứu đã thực hiện một phân tích chi tiết về hiệu suất và chi phí tính toán của các phương pháp khác nhau. Họ đã so sánh từ hai khía cạnh: thứ nhất là những tác vụ nào hưởng lợi nhiều nhất từ các công cụ, và thứ hai là phương pháp nào sử dụng công cụ hiệu quả nhất.

What tasks benefit the most from tools?

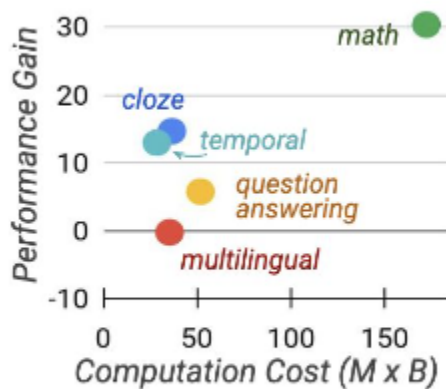


Figure 5: Compute & performance gain with ToolFormer.

What methods are efficient in tooling?

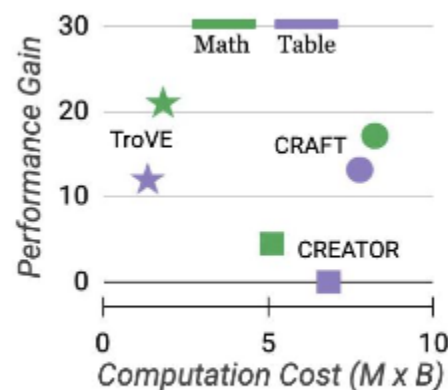


Figure 6: Comparing different tool-making methods.

Ví dụ, trong hình minh họa, các bộ dữ liệu lớn cho thấy sự cải thiện đáng kể về hiệu suất. Tuy nhiên, đối với các tác vụ đa ngôn ngữ, mặc dù không phải là không có cải thiện, nhưng hiệu suất lại giảm và vẫn tiêu tốn nhiều tài nguyên tính toán. Điều này cho thấy rằng các tác vụ đa ngôn ngữ có thể không luôn luôn được hưởng lợi từ việc sử dụng công cụ.

Ngoài ra, các nhà nghiên cứu cũng xem xét các phương pháp nào hiệu quả trong việc sử dụng công cụ ngay cả trên cùng một bộ dữ liệu. Ví dụ, khi so sánh trên các bộ dữ liệu MTH và table, có những phương pháp sử dụng ít tài nguyên tính toán hơn nhưng vẫn đạt được mức cải thiện tương tự. Điều này nhấn mạnh tầm quan trọng của việc đánh giá toàn diện để có cái nhìn rõ ràng hơn về hiệu quả của các phương pháp.

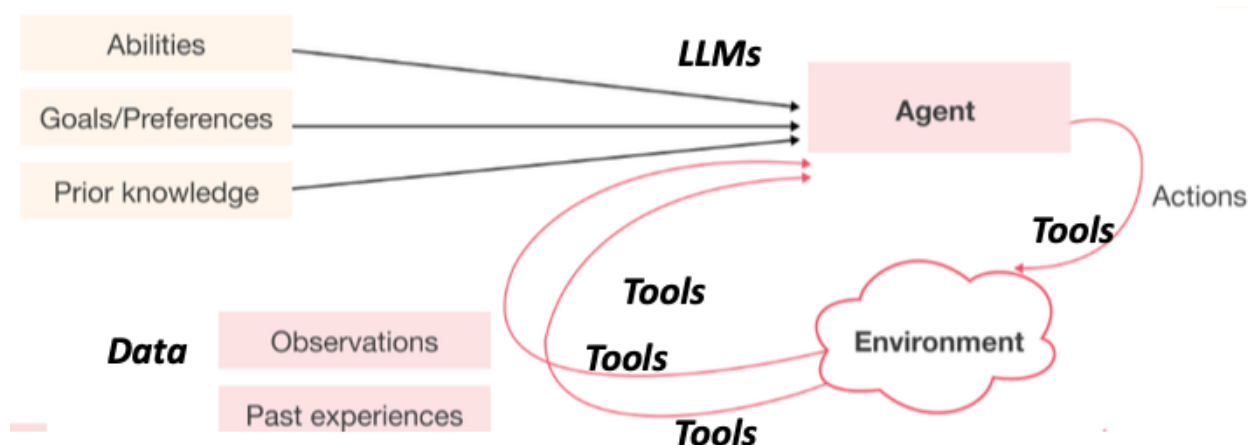
In Summary

- Tool Basics: definition & functionality
- Scenarios: what tools, what tasks, what methods
- Evaluation, empirical benefit, future directions

Language Model as Agents

What are agents?

Agents có thể được hiểu là bất kỳ thứ gì có khả năng nhận thức môi trường thông qua các cảm biến và tác động lên môi trường thông qua các cơ cấu chấp hành. Trong sơ đồ này, bạn có thể thấy agent sẽ nhận thông tin từ môi trường và thực hiện các hành động dựa trên thông tin đó. Agents có thể có các khả năng, kiến thức, mục tiêu hoặc sở thích nhất định.



Vì chủ đề hôm nay là về các agent trong mô hình ngôn ngữ, chúng ta thường sử dụng các mô hình ngôn ngữ lớn làm agent. Các công cụ đã được đề cập có thể được sử dụng như các cảm biến hoặc cơ cấu chấp hành. Ví dụ, việc phát nhạc có thể được xem như một cơ cấu chấp hành. Các khả năng, kiến thức trước đó hoặc kinh nghiệm có thể được coi là dữ liệu hoặc dữ liệu huấn luyện cho mô hình ngôn ngữ của bạn.

How to get started in LLM Agents?

Để bắt đầu với các agent trong mô hình ngôn ngữ, tôi sẽ trình bày bốn giai đoạn để xây dựng một LLM Agent. Đầu tiên, tôi sẽ đề cập đến một số nhiệm vụ và ứng dụng. Thứ hai, tôi sẽ giới thiệu một số phương pháp không cần huấn luyện để xây dựng các agent, giúp bạn có thể sử dụng với các mô hình dựa trên API. Tiếp theo là môi trường đánh giá và Benchmark, một chủ

đề cực kỳ quan trọng trong nghiên cứu. Cuối cùng, tôi sẽ nói ngắn gọn về một số phương pháp huấn luyện để cải thiện các agent. Vì đây là một lĩnh vực đang phát triển, một số phương pháp huấn luyện có thể chưa phải là tốt nhất.

Tasks and Applications for LLM Agents

Why do we want agents?

Hãy tưởng tượng nếu mọi việc có thể được thực hiện chỉ bằng cách nói chuyện, giống như cách mà các "human agents" (tác nhân con người) hoạt động. Ví dụ, bạn nói chuyện với một "real estate agent" để mua nhà.

How do People Interact with Computers?

Hiện tại, khi tương tác với máy tính, chúng ta thường sử dụng giao diện đồ họa hoặc viết mã bằng tay qua bàn phím và chuột. Nhưng trong tương lai, nếu mọi thứ có thể được thực hiện chỉ bằng cách nói chuyện với các trợ lý ảo như Alexa hay Google Assistant, thì điều đó sẽ tiết kiệm thời gian, tự nhiên hơn, dễ tiếp cận và không cần phải vượt qua các "learning curve" (đường cong học tập) của các chương trình phức tạp.

Hiện nay, có một số agent giúp thực hiện các nhiệm vụ thông qua giao diện ngôn ngữ tự nhiên, chẳng hạn như Siri, Google Assistant, và Alexa. Ví dụ, bạn có thể đặt báo thức, và đó chính là một agent vì nó thực hiện việc đặt báo thức cho bạn. Ngoài ra, còn có các công cụ lập trình ngôn ngữ tự nhiên như GitHub Copilot, giúp bạn viết mã. Bạn chỉ cần nói "I want to sort my list in descending order" và nó sẽ tạo ra mã thực hiện điều đó cho bạn.

Tool Integrations into Chatbots

Nhiều người hiện nay sử dụng Chat GPT trong cuộc sống hàng ngày. Khi tính năng tích hợp plugin được giới thiệu, công cụ này đã trở nên mạnh mẽ hơn. Các tích hợp này cho phép chatbot thực hiện nhiều tác vụ hữu ích như đặt lịch hẹn PRS hoặc tạo đơn hàng trên Instacart.

Robots

Trong lĩnh vực robot, một ứng dụng tiềm năng của các agent là điều hướng bằng ngôn ngữ tự nhiên. Ví dụ, agent có thể quan sát môi trường xung quanh giống như trong Google Street View và thực hiện các chỉ dẫn ngôn ngữ tự nhiên để di chuyển qua các con phố.

Một tập dữ liệu phổ biến cho nhiệm vụ này là Alward, nơi agent được đặt trong một môi trường mô phỏng và nhận mô tả bằng văn bản, chẳng hạn như "bạn đang ở giữa phòng, xung quanh có giường và tủ ngăn kéo".

Nhiệm vụ của agent là tìm và tương tác với một đối tượng cụ thể, ví dụ như đồng hồ báo thức trên bàn. Mô hình ngôn ngữ lý tưởng sẽ dự đoán hành động cần thực hiện, chẳng hạn như di

chuyển đến bàn, và sau đó cập nhật quan sát khi đến vị trí mới. Qua đó, ta có thể thấy cách agent tương tác với môi trường xung quanh thông qua việc quan sát và thực hiện hành động.

Games

Trong lĩnh vực trò chơi, có nhiều ứng dụng và tiêu chuẩn đánh giá thú vị. Một ví dụ điển hình là Mind Dojo, nơi người dùng có thể tạo ra một agent có khả năng lắng nghe hướng dẫn bằng ngôn ngữ tự nhiên và thực hiện các nhiệm vụ trong trò chơi Minecraft. Ngoài ra, DeepMind gần đây đã phát triển một công trình mang tên Sema, cho phép người dùng đưa ra hướng dẫn bằng ngôn ngữ tự nhiên để lập trình một trò chơi bắn thiên thạch, giúp tự động bắn hạ thiên thạch trong trò chơi.

Software Development

Trong lĩnh vực phát triển phần mềm, các agents trong mô hình ngôn ngữ đang được ứng dụng ngày càng rộng rãi. Gần đây, một startup tên là Devon đã phát triển một "kỹ sư phần mềm AI". Hệ thống này bao gồm một trình soạn thảo mã, một terminal để thực thi lệnh, và một trình duyệt web để tìm kiếm tài liệu.

Lý tưởng nhất, nếu mọi thứ được tự động hóa, bạn chỉ cần đưa ra một chỉ dẫn tự nhiên, chẳng hạn như "hoàn thành bài tập 711 cho tôi", và agent sẽ thực hiện công việc đó trong không gian làm việc này. Trong kịch bản này, hệ thống sẽ quan sát chuỗi lệnh bao gồm terminal, trình duyệt web, và trình soạn thảo mã. Các hành động có thể thực hiện bao gồm: ra lệnh cho terminal, tìm kiếm trên trình duyệt web, hoặc viết mã trong trình soạn thảo.

UI Automation

Trong lĩnh vực tự động hóa giao diện người dùng (UI), có thể thực hiện các tác vụ như duyệt web, phát nhạc, hoặc phóng to thu nhỏ giao diện. Những thao tác này liên quan đến việc điều hướng giao diện đồ họa người dùng (GUI).

Training-free Methods for Building Agents

Vừa rồi, chúng ta đã có cái nhìn tổng quan về các nhiệm vụ và ứng dụng của agents, câu hỏi đặt ra là: liệu chúng ta có thể phát triển các phương pháp để xây dựng những LM agents hiệu quả hay không? Với một mô hình ngôn ngữ mạnh mẽ trong tay, chủ đề này trở nên vô cùng hấp dẫn. Để bắt đầu, tôi sẽ giới thiệu một số phương pháp không cần huấn luyện để phát triển các agent này.

Trong phần này, chúng ta sẽ khám phá cách biến một mô hình ngôn ngữ thành một agent thông minh có khả năng tương tác với môi trường. Để làm được điều này, một agent cần có khả năng nhận diện và xử lý các quan sát từ môi trường hiện tại, có thể là văn bản, hình ảnh, âm thanh hoặc dữ liệu có cấu trúc.

Quan sát từ môi trường

Một agent thường nhận đầu vào từ nhiều loại dữ liệu khác nhau. Ví dụ, trong một trò chơi, agent có thể nhận dữ liệu từ văn bản mô tả môi trường xung quanh, hình ảnh chụp màn hình hiện tại, hoặc âm thanh để nhận biết những gì đang xảy ra xung quanh mà không thể nhìn thấy trên màn hình. Đối với các ứng dụng web, dữ liệu có thể là cấu trúc HTML của trang web.

Khả năng lập kế hoạch và suy luận

Để thực hiện các nhiệm vụ phức tạp, mô hình ngôn ngữ cần có khả năng lập kế hoạch và suy luận. Một trong những phương pháp phổ biến là "Chain of Thought" (Chuỗi suy nghĩ), cho phép mô hình tạo ra các bước suy luận để thực hiện nhiệm vụ. Ví dụ, nếu nhiệm vụ là đặt một lọ tiêu lên ngăn kéo, mô hình sẽ suy luận rằng cần tìm lọ tiêu trước, sau đó mới đặt nó lên ngăn kéo.

Khả năng sử dụng công cụ

Ngoài việc lập kế hoạch và suy luận, mô hình ngôn ngữ cần có khả năng tương tác với môi trường thông qua các hành động. Điều này đòi hỏi mô hình phải tạo ra các lệnh hành động có thể thực thi, như gọi API để thay đổi trạng thái của môi trường và nhận lại quan sát mới để tiếp tục quá trình.

Một ví dụ thực tế là sử dụng mô hình ngôn ngữ để quyết định có nên đi leo núi hay không. Mô hình có thể gọi API để lấy thông tin thời tiết tại vị trí hiện tại, sau đó đưa ra quyết định dựa trên dữ liệu thời tiết nhận được.

Khi có nhiều API, việc cung cấp tất cả thông tin cho mô hình có thể vượt quá khả năng xử lý. Một giải pháp là sử dụng bộ nhớ ngoài để truy vấn và chỉ cung cấp những API cần thiết nhất dựa trên ngữ cảnh hiện tại.

Tạo mã để thực thi nhiệm vụ

Một cách tiếp cận khác là để mô hình ngôn ngữ tạo mã lập trình để thực hiện nhiệm vụ. Ví dụ, mô hình có thể tạo mã Python để lên lịch một cuộc họp, sau đó thực thi mã này để hoàn thành nhiệm vụ.

Evaluation Environment and Benchmark

Evaluation of LLM Agents

Đầu tiên, cần nhấn mạnh rằng việc đánh giá các LLM agent là một thách thức lớn. Nhiều nghiên cứu hiện tại sử dụng các môi trường đơn giản và các nhiệm vụ cơ bản, dẫn đến việc hiệu suất dễ dàng đạt đến mức bão hòa. Ví dụ, khi yêu cầu ChatGPT kiểm tra thời tiết để quyết định có nên đi hiking hay không, hoặc đặt lịch họp qua Google Calendar API, các tác vụ này đều rất đơn giản và dễ dàng đạt độ chính xác 100%. Tuy nhiên, điều này không phản ánh được tiến bộ thực sự trong nghiên cứu về các LLM agent.

Các benchmark đánh giá hiện tại thường là môi trường không trạng thái (stateless) và không tương tác. Ví dụ, "mind to web" tập trung vào việc chuyển đổi các trang web qua các hành động cụ thể và đánh giá dựa trên độ chính xác của chuỗi hành động. Tuy nhiên, cách đánh giá này có thể không lý tưởng vì nó không cho phép sự linh hoạt trong thứ tự thực hiện các hành động, dẫn đến việc bỏ lỡ những cơ hội mà agent thực hiện đúng nhưng không theo thứ tự đã định trước.

Ngoài ra, còn có các môi trường tương tác nhưng thường là ngắn hạn (short Horizon), như "webshop" - một phiên bản đơn giản hóa của Amazon. Trong môi trường này, các tác vụ thường chỉ yêu cầu một vài bước để hoàn thành, ví dụ như nhập "I love 711" vào ô văn bản và nhấn submit. Mặc dù các môi trường này cung cấp sự tương tác, nhưng chúng vẫn khá đơn giản và không phản ánh được độ phức tạp của các tình huống thực tế.


Instruction: I am looking for x-large, red color women faux fur lined winter warm jacket coat, and price lower than 70.00 dollars

Current Query: women fur jacket coat


Results

Page 1 (1-10) of 50 total results


[Back to Search](#) [Next >](#)



B09KPT8G37
Women Faux Fur Lined Jacket Coat
Winter Warm Thick Fleece Outwear
Trench Zipper Plus Size Long
Amazon Dash Product



B07ZXBGDXF
Women's Coat, FORUW Winter Faux
Fur Fleece Outwear Warm Lapel
Biker Motor Aviator Jacket



B098XT34EY
Fjackets Real Lambskin Sherpa
Jacket - Mens Leather Jacket
★★★★★ 4.7

Current Action: click[Fjackets Real Lambskin...]

Keys to Agent Benchmarks

Trong việc xây dựng các benchmark cho agent, có một số yếu tố quan trọng cần xem xét. Đầu tiên, cần có một môi trường tương tác, vì nếu không có môi trường, việc đánh giá chỉ dừng lại ở việc kiểm tra xem hành động có đúng hay không, mà không xem xét kết quả thực thi cuối cùng.

Thứ hai, cần có sự đa dạng trong chức năng. Nếu chỉ tập trung vào một lĩnh vực như shopping, có nguy cơ là sẽ "overfit" vào chức năng đó. Nội dung cũng cần phong phú và thực tế để giúp agent có thể thích ứng tốt hơn với các trang web hiện đại, từ đó cải thiện khả năng chuyển giao hiệu suất từ các benchmark sang các trang web thực tế.

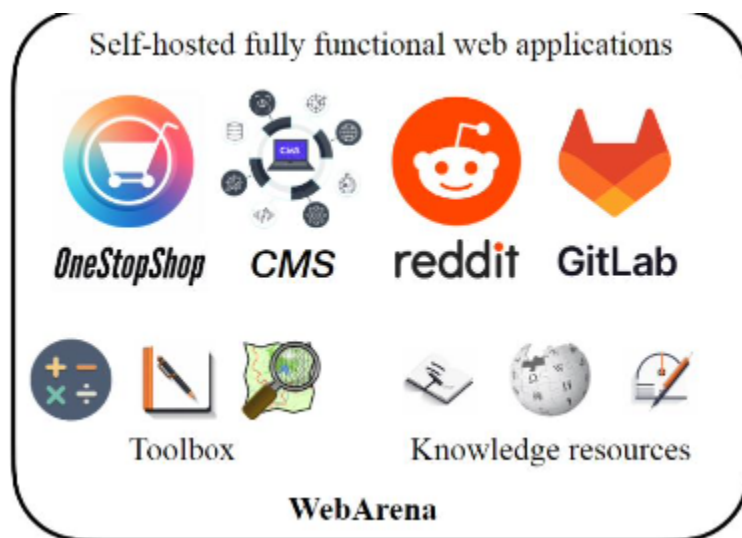
Môi trường cần phải tương tác, dễ mở rộng và có thể tái tạo. Việc tái tạo là rất quan trọng trong cộng đồng nghiên cứu, vì vậy chúng ta không nên sử dụng các trang web trực tiếp làm môi trường thử nghiệm do chúng thường xuyên thay đổi. Chẳng hạn, nếu bạn đạt 90% độ chính xác hôm qua, hôm nay có thể chỉ còn 20% vì các trang web đã thay đổi.

Chúng ta cũng nên hướng tới các nhiệm vụ dài hạn với độ khó đủ để thách thức agent. Việc kết hợp nhiều trang web, như "Reddit" để tìm kiếm đánh giá, cũng là một điểm cộng. Cuối cùng, cần có các chỉ số đánh giá đáng tin cậy để khuyến khích đạt được mục tiêu cuối cùng thay vì chỉ thỏa mãn một phần, đồng thời thúc đẩy agent thực hiện nhiệm vụ đúng cách thay vì chỉ làm theo các hành động được cung cấp.

WebArena Environment Design

Trong lĩnh vực phát triển web, một trong những công trình mới nhất là Arena, một môi trường nhằm đáp ứng các yêu cầu đã được đề ra trước đó, cụ thể là một "setbox internet". Đây là một giải pháp mã nguồn mở, sẵn sàng cho sản xuất, cho phép triển khai các trang web và dữ liệu từ các trang web thực tế phổ biến. Chúng tôi thực hiện việc thu thập dữ liệu từ các trang như Amazon và đưa vào một trang web giả lập Amazon.

Arena cũng dễ dàng phân phối nhờ vào việc sử dụng Docker, giúp cho việc tái tạo môi trường trở nên thuận tiện hơn. Mặc dù việc lựa chọn trang web có thể bị giới hạn, chúng ta cố gắng làm cho nó đa dạng bằng cách bao gồm nhiều loại trang web khác nhau, như một trang web mua sắm, một trang quản lý nội dung, và một diễn đàn tương tự như Reddit. Ngoài ra, chúng tôi còn tích hợp Wikipedia và một số công cụ khác, bao gồm cả bản đồ, trong bộ công cụ đánh giá này.



Collecting Realistic Intents

Việc thu thập các ý định thực tế của người dùng là rất quan trọng. Một cách hiệu quả để thu thập những ý định này là kiểm tra lịch sử trình duyệt của chính chúng ta hoặc của người khác. Sau đó, chúng ta có thể phân loại chúng thành ba loại chính:

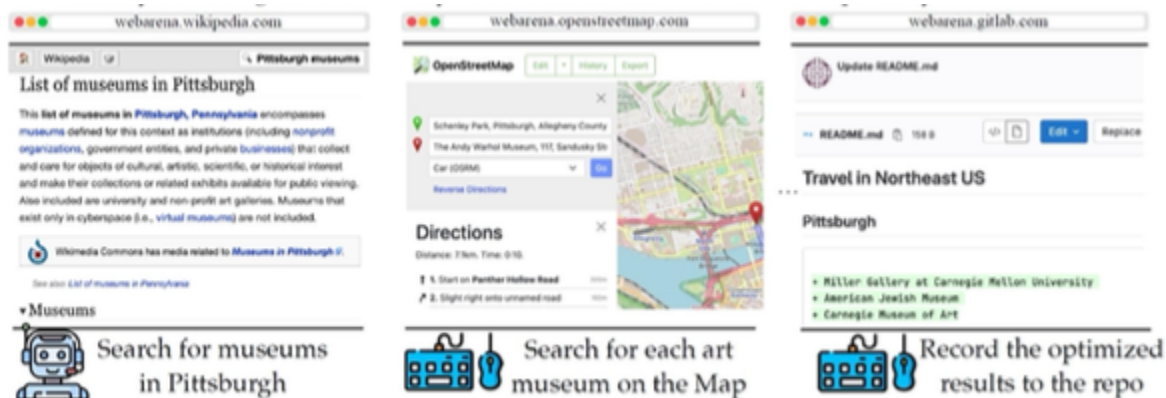
1. Tìm kiếm thông tin: Phần lớn hoạt động trên internet của chúng ta là để tìm kiếm thông tin. Ví dụ, khi bạn muốn biết lần cuối cùng mình mua dầu gội là khi nào để nhắc nhở bản thân.

- Điều hướng: Đây là những hoạt động giúp chúng ta tìm đường hoặc truy cập vào các tài nguyên cụ thể. Ví dụ, bạn muốn kiểm tra các yêu cầu hợp nhất (merge requests) được giao cho mình khi bắt đầu một ngày làm việc mới.
- Thao tác nội dung và cấu hình: Những tác vụ này thường yêu cầu bạn thay đổi môi trường ở một mức độ nào đó. Như đã thảo luận trước đây, một agent không chỉ có thể thu thập thông tin từ môi trường mà đôi khi còn phải thực hiện các hành động ngược lại. Ví dụ, nếu bạn muốn đăng câu hỏi "Có cần thiết phải có xe hơi ở New York City không?" trên một subreddit mà bạn nghĩ sẽ nhận được câu trả lời, bạn đang thực hiện một thay đổi nhỏ trong môi trường trực tuyến của mình.

Việc hiểu rõ và phân loại các ý định này giúp cải thiện khả năng tương tác và đáp ứng của các hệ thống thông minh.

Example Tasks in WebArena

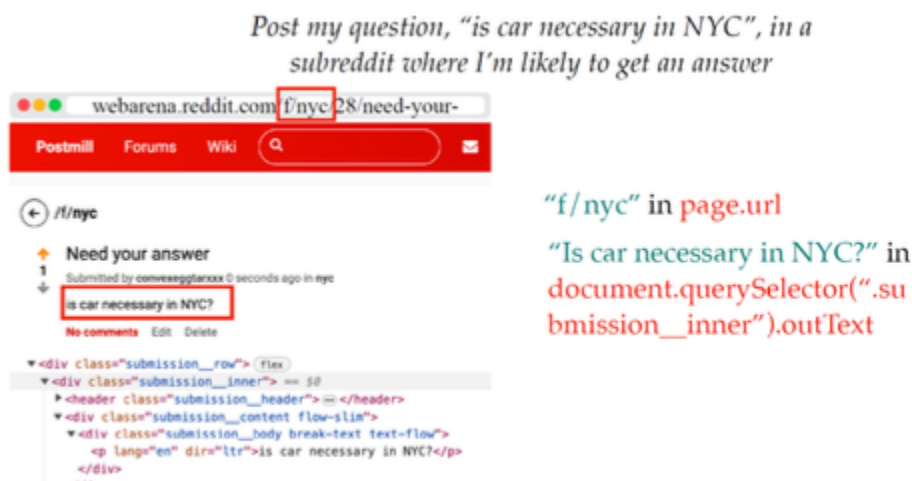
Dưới đây là một ví dụ về một nhiệm vụ phức tạp trong một bài kiểm tra: tạo kế hoạch thăm các bảo tàng ở Pittsburgh với khoảng cách lái xe tối thiểu, bắt đầu từ Shy Park và ghi lại thứ tự trong kho lưu trữ du lịch Đông Bắc Hoa Kỳ. Nhiệm vụ này yêu cầu nhiều bước, bao gồm tìm kiếm thông tin trên nhiều trang web. Đầu tiên, bạn cần tìm kiếm các bảo tàng ở Pittsburgh, có thể sử dụng Google hoặc Wikipedia. Sau đó, bạn tìm kiếm từng bảo tàng nghệ thuật trên phần mềm bản đồ và cuối cùng kiểm tra khoảng cách lái xe tối thiểu. Nhiệm vụ này đòi hỏi cả khả năng suy luận toán học để thu thập và tối ưu hóa khoảng cách lái xe, sau đó sắp xếp chúng trong kho lưu trữ.



Outcome/ Execution-based Evaluation

Trong quá trình thực hiện các tác vụ phức tạp, chúng ta không nên chỉ dựa vào đánh giá dựa trên chuỗi hành động. Mục tiêu của chúng ta là xác thực trực tiếp tính đúng đắn của kết quả thực thi. Ví dụ, câu hỏi "Lần cuối tôi mua dầu gội là khi nào?" có thể được trả lời trực tiếp bằng một ngày cụ thể, vì chúng ta đã biết trước dữ liệu có trong bộ đánh giá. Đối với những câu hỏi phức tạp hơn, chẳng hạn như "Có cần thiết phải có ô tô ở New York City không?", chúng ta cần kiểm tra nội dung trong trang web hoặc tài liệu HTML để xác nhận. Những công cụ xác thực này có thể được xem như các bài kiểm tra đơn vị trong phát triển phần mềm, chỉ kiểm tra kết

quả cuối cùng của môi trường để đảm bảo tính chính xác, từ đó giảm bớt sự phụ thuộc vào việc kiểm tra từng hành động.



Observation & Action Space

Trong các tác vụ liên quan đến không gian quan sát và hành động, có thể sử dụng nhiều mô hình khác nhau. Bạn có thể chụp ảnh màn hình của trang web hoặc sử dụng không gian văn bản như mã nguồn HTML thô. Ngoài ra, có thể sử dụng phiên bản có cấu trúc hơn gọi là "cây truy cập" (accessibility tree), một cấu trúc dạng cây để biểu diễn cấu trúc của trang web. Đối với không gian hành động, chúng ta có thể sử dụng bàn phím để nhập liệu hoặc chuột để nhấp, di chuột và cuộn, cũng như sử dụng trình duyệt để quay lại trang trước.

Prompting LLM as Agent

Trong bài viết này, chúng ta đã thảo luận về việc sử dụng phương pháp gợi ý và học từ ngữ cảnh với ít ví dụ (Few-shot in-context learning) để cung cấp hướng dẫn tổng quát và một vài ví dụ minh họa. Để đánh giá hiệu suất của GPT-4 trong các nhiệm vụ phức tạp, chúng ta sẽ cung cấp cho GPT-4 một số hướng dẫn cụ thể. Chúng ta giới thiệu nó như một công cụ trí tuệ nhân tạo tự động và mô tả không gian quan sát mà nó sẽ làm việc. Chúng ta cũng chỉ ra các hành động mà nó có thể thực hiện và cung cấp một số ví dụ minh họa.

You are an autonomous intelligent agent tasked with navigating a web browser. You will be given web-based tasks. These tasks will be accomplished through the use of specific actions you can issue.

You can observe the following information:

...

You can do the following actions:

...

...

Không gian quan sát được mô tả như một cấu trúc dạng cây, là phiên bản đã được lọc của HTML, đại diện cho trang web. Thông tin bao gồm URL, mô tả nhiệm vụ và ví dụ về đầu ra. Chúng ta áp dụng phương pháp suy luận theo chuỗi tư duy, từng bước chỉ ra những gì cần làm và hành động, dựa trên không gian hành động đã được cung cấp trước đó. Các hành động này là những gì mà hệ thống có thể thực hiện trong môi trường đã định.

Example input:

OBSERVATION:

[1744] link 'HP CB782A#ABA 640 Inkjet Fax Machine (Renewed)'

[1757] button 'Add to Cart'

URL: <http://onestopmarket.com/office-products/office-electronics.html>

OBJECTIVE: What is the price of HP Inkjet Fax Machine

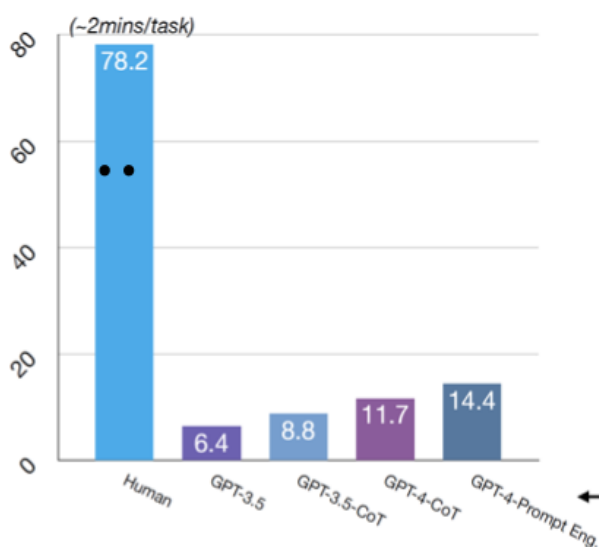
PREVIOUS ACTION: None

Example output:

Let's think step-by-step. This page lists ... the next action I will perform is `click [1744]`
(Optional chain-of-thought reasoning) (Issued action)

WebArena is Challenging

Bạn có thể thấy WebArena là một thách thức lớn. Khi yêu cầu con người thực hiện các nhiệm vụ này, họ có thể đạt độ chính xác khoảng 78% trong vòng 2 phút. Tuy nhiên, với các mô hình ngôn ngữ tiên tiến như GPT-3.5 hay GPT-4, ngay cả khi sử dụng các kỹ thuật prompt nâng cao, chỉ có thể giải quyết khoảng 14% bài kiểm tra.



Mặc dù phương pháp "chain of thought" có thể hỗ trợ, nhưng lợi ích mang lại vẫn còn hạn chế và GPT-4 vẫn kém xa so với hiệu suất của con người. Kỹ thuật prompt thường nhấn mạnh sự

nhạy cảm của các mô hình lớn đối với những thay đổi nhỏ trong hướng dẫn, và đôi khi việc này cũng không hề đơn giản.

Failures: Not Knowing How

Trong một số trường hợp thất bại của mô hình ngôn ngữ, đôi khi mô hình không biết nên nhấn nút nào. Ví dụ, khi yêu cầu "hiển thị cho tôi những khách hàng đã bày tỏ sự không hài lòng về một chiếc áo khoác có khóa kéo", con người có thể dễ dàng nhận ra cần truy cập trang sản phẩm trong danh mục và kiểm tra đánh giá, hoặc vào phần đánh giá và tìm kiếm áo khoác đó. Tuy nhiên, mô hình như GPT-4 có thể thiếu kiến thức thông thường và có thể chuyển hướng sai, như vào phần khách hàng. Điều này cho thấy mô hình ngôn ngữ chưa có khả năng suy luận hoặc lập kế hoạch tốt khi thiếu thông tin cơ bản.

Failures: Not Being Accurate

Trong quá trình sử dụng các mô hình ngôn ngữ, đôi khi chúng có thể không chính xác. Ví dụ, khi bạn yêu cầu nhập ngày đến hạn, mô hình có thể nhập sai định dạng. Nếu trang web không được thiết kế tốt, quá trình này có thể bị dừng lại do định dạng không chính xác. Lý tưởng nhất là bạn nên sử dụng các công cụ kiểm tra trạng thái như vdet. Tuy nhiên, đôi khi mô hình ngôn ngữ có thể tự động tìm cách nhập văn bản mà không cần sự can thiệp của người dùng.

“... and set the due date to 2023/12/23”

Due date

2023/12/23



“... and set the due date to 2023-12-13”

Due date

2023-12-13



Failures: Trivial Errors

Trong bài báo nghiên cứu, chúng ta nhận thấy rằng các lỗi nhỏ nhặt có thể gây ra vấn đề lớn. Ví dụ, với mô hình GPT-4, khoảng 21% ví dụ gặp lỗi do việc gõ lặp lại. Điều này có thể liên quan đến hiện tượng "hallucination" trong các mô hình ngôn ngữ lớn, khi chúng tự động nhập một loạt các từ không liên quan như "DMV area" nhiều lần.

Ngoài ra, có những lỗi không hề đơn giản. Một ví dụ thú vị là khi yêu cầu mô hình GPT-4 thực hiện tác vụ trên trang GitLab, nó không hiểu từ "myself" và chỉ nhập từ này như một chuỗi ký tự. Thực tế, mô hình cần phải xác định danh tính của người dùng và nhập "me" hoặc tên người dùng của chính nó vào trường này. Điều này cho thấy sự phức tạp trong việc xử lý ngữ cảnh và danh tính cá nhân của các mô hình ngôn ngữ lớn.

Training Methods for Improving Agents

Trong phần này, tôi sẽ đề cập đến một số phương pháp huấn luyện để cải thiện hiệu suất của các agent. Trước đó, chúng ta đã thảo luận về các ứng dụng, cũng như một số kỹ thuật prompt và các tiêu chuẩn đánh giá tiên tiến. Mặc dù đã có môi trường thử nghiệm và áp dụng các kỹ thuật như "Chain of Thought prompting", nhưng hiệu suất vẫn chưa đạt yêu cầu.

Learning of LLM Agents

Chủ đề chính của phần này là các phương pháp học cho LLM agents. Tôi sẽ tập trung vào ba loại hình học chính:

1. Học trong ngữ cảnh (In-context learning): Một số người có thể cho rằng đây chỉ là gợi ý, nhưng việc cung cấp các ví dụ minh họa tốt hơn có thể giúp khai thác tối đa phương pháp này.
2. Tinh chỉnh có giám sát (Supervised fine-tuning): Phương pháp này dựa trên việc học từ các chuyên gia. Nếu có dữ liệu chất lượng về cách con người thực hiện một nhiệm vụ, chúng ta có thể sử dụng chúng để cải thiện mô hình.
3. Học tăng cường (Reinforcement learning): Đối với các tác nhân tương tác với môi trường, học tăng cường là một kỹ thuật phổ biến. Nếu có một môi trường đủ tốt, mô hình có thể học từ chính môi trường đó.

In-context Learning

Học trong ngữ cảnh (In-context Learning) là một phương pháp mà mô hình ngôn ngữ có thể thực hiện nhiệm vụ chỉ bằng cách điều chỉnh theo các ví dụ đầu vào và đầu ra mà không cần tối ưu hóa các tham số khác. Điều này thường xảy ra khi chúng ta không có quyền truy cập vào các tham số hoặc việc huấn luyện quá tốn kém. Tuy nhiên, đây vẫn là một phương pháp phổ biến trong học máy.

Ví dụ, khi thực hiện các bài kiểm tra trên nền tảng web, chúng ta cung cấp một số ví dụ về quan sát và hành động cần thực hiện cho một nhiệm vụ cụ thể.

```
example_user
OBSERVATION:
[1744] link 'HP CB782A#ABA 640 Inkjet Fax Machine (Renewed)'
[1749] StaticText '$279.49'
[1757] button 'Add to Cart'
[1760] button 'Add to Wish List'
[1761] button 'Add to Compare'
URL: http://onestopmarket.com/office-products/office-electronics.html
OBJECTIVE: What is the price of HP Inkjet Fax Machine
PREVIOUS ACTION: None
```

```
example_assistant
Let's think step-by-step. This page lists the information of HP Inkjet Fax Machine, which is the product identified in the objective. Its price is $279.49. I think I have achieved the objective. I will issue the stop action with the answer. In summary, the next action I will perform is `` ` stop [$279.49]` ``
```

Trong học trong ngữ cảnh, chúng ta có thể đưa ra các ví dụ về quan sát của người dùng, chẳng hạn như một trang web được biểu diễn dưới dạng cây tài liệu HTML đã được rút gọn. Phần ví dụ của assistant sẽ đóng vai trò như nhãn đầu ra, thể hiện cách thực hiện suy luận theo chuỗi và định dạng hành động cần thực hiện.

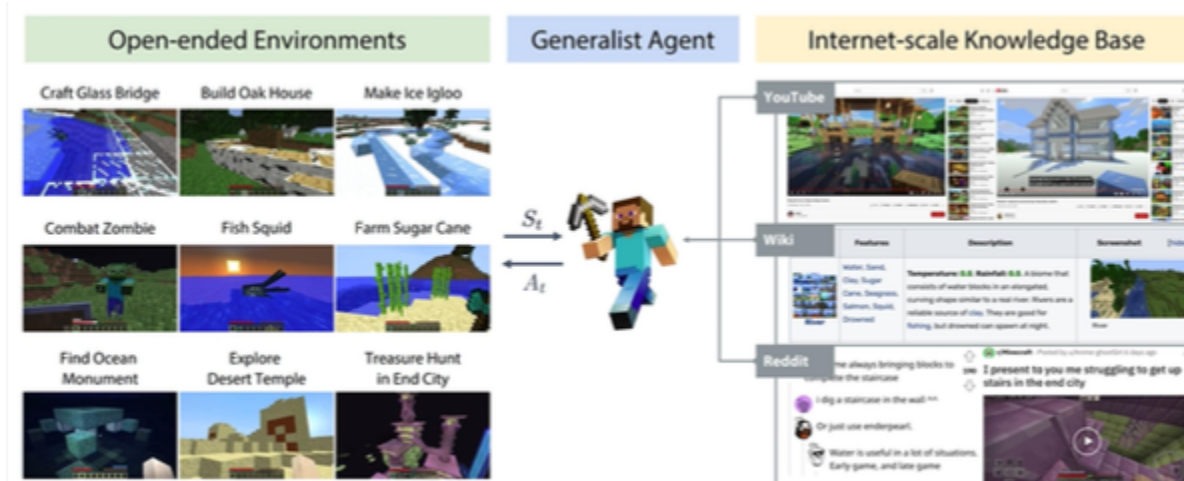
Với việc cung cấp hướng dẫn rõ ràng, không gian hành động và một số ví dụ điển hình về cặp quan sát-hành động, mô hình ngôn ngữ có thể tự động nhận diện định dạng và tạo ra các hành động theo định dạng đã chỉ định. Học trong ngữ cảnh đôi khi rất hiệu quả trong việc điều chỉnh mô hình ngôn ngữ theo yêu cầu cụ thể của bạn.

Supervised Fine-tuning

Để tinh chỉnh có giám sát, đầu tiên chúng ta thu thập một lượng lớn dữ liệu từ các chuyên gia thông qua việc thích ứng của con người. Ví dụ, bạn có thể có các cặp nhiệm vụ như quan sát ý định, hành động, và sau đó tối ưu hóa chúng bằng cách sử dụng hàm mất mát cross-entropy. Nhiều nghiên cứu hiện tại cố gắng tối ưu hóa các mô hình ngôn ngữ bằng cách thu thập các chú thích từ con người.

```
task_intent, [(obs_1, action_1), ..., (obs_N, action_N)]
```

Phương pháp này hoạt động rất hiệu quả nhưng đòi hỏi nhiều tài nguyên và không thể học được nhiều từ các chuỗi hành động thất bại. Chẳng hạn, nếu bạn có một chuỗi hành động thành công và một chuỗi thất bại, bạn có thể không tận dụng được chuỗi thất bại, ngay cả khi chỉ có bước cuối cùng là sai. Có một số kỹ thuật tăng cường dữ liệu, ví dụ như trong trò chơi Minecraft, bạn có thể thực hiện tăng cường dữ liệu dựa trên video YouTube, bài viết Wikipedia, hoặc các thảo luận trên Reddit.



Reinforcement Learning

Trong lĩnh vực học tăng cường, hiện nay có nhiều phương pháp mới đang được nghiên cứu và phát triển. Trước đây, một số nghiên cứu tập trung vào việc học từ phản hồi của con người. Tuy

nhien, hiện tại, có những hướng đi mới không cần đến phản hồi từ con người. Thay vào đó, chúng ta có thể thay thế các phần thưởng (rewards) bằng môi trường thực tế.

Ví dụ, khi bạn có quyền truy cập vào một môi trường như Web Arena, việc xác định một nhiệm vụ có thành công hay không có thể được thực hiện tự động thông qua môi trường đó. Điều này mang lại một dạng phản hồi tự nhiên từ môi trường, giúp cải thiện quá trình học tập của mô hình.

Nếu bạn quan tâm đến các nghiên cứu đang diễn ra trong lĩnh vực này, có thể tham khảo thêm các bài báo (Ouyang et al. 2022, Song et al. 2024, Yao et al. 2023) trong phần tài liệu tham khảo dưới đây để có cái nhìn sâu hơn về những tiến bộ mới nhất.

Resources

1. <https://phontron.com/class/anlp2024/lectures/#language-agents-march-28>
2. [TroVE: Inducing Verifiable and Efficient Toolboxes for Solving Programmatic Tasks Wang et al. 2024](#)
3. [Chain-of-Thought Prompting Elicits Reasoning in Large Language Models Wei et al. 2022](#)
4. [Toolformer: Language Models Can Teach Themselves to Use Tools](#)
5. [ReAct: Synergizing Reasoning and Acting in Language Models Yao et al. 2023](#)
6. [PAL: Program-aided Language Models Gao et al. 2022](#)
7. [Mind2Web: Towards a Generalist Agent for the Web Deng et al. 2023](#)
8. [WebShop: Towards Scalable Real-World Web Interaction with Grounded Language Agents Yao et al. 2023](#)
9. [WebArena: A Realistic Web Environment for Building Autonomous Agents Zhou et al. 2023](#)
10. [MineDojo: Building Open-Ended Embodied Agents with Internet-Scale Knowledge Fan et al. 2022](#)
11. [Training language models to follow instructions with human feedback Ouyang et al. 2022](#)
12. [Trial and Error: Exploration-Based Trajectory Optimization for LLM Agents Song et al. 2024](#)
13. [Retroformer: Retrospective Large Language Agents with Policy Gradient Optimization Yao et al. 2023](#)