Lecture 12: RL from Human Feedback

Giới thiệu

Trong bài viết này, chúng ta sẽ tìm hiểu về “Learning from Human Feedback”. Ban đầu, tôi đã đề cập đến “Reinforcement Learning from Human Feedback” vì đây là thuật ngữ phổ biến hiện nay. Tuy nhiên, thực tế còn có nhiều phương pháp khác để học từ phản hồi của con người.

Phần đầu tiên, chúng ta sẽ khám phá các cách thu thập phản hồi từ con người cho các mô hình sinh văn bản. Trọng tâm sẽ đặt vào các tác vụ sinh nội dung, bởi chúng phức tạp hơn nhiều so với các bài toán phân loại thông thường. Sau khi thảo luận kỹ về phương pháp thu thập, chúng ta sẽ chuyển sang cách học từ những tín hiệu phản hồi này.

Maximum Likelihood Training

Hiện tại, phương pháp huấn luyện phổ biến nhất là Maximum Likelihood. Mục tiêu của phương pháp này là tối đa hóa khả năng dự đoán từ tiếp theo trong một câu dựa trên các từ trước đó. Điều này giúp xác định tổn thất (loss) của đầu ra dựa trên đầu vào, trong đó đầu vào có thể là một đoạn văn bản và đầu ra là câu trả lời tương ứng.

$$l(Y|X) = -\ log\ P(Y|X) = -\sum_{t} log\ P\left(y_t|\ X,\ y_{<t}\right)$$

Tuy nhiên, phương pháp Maximum Likelihood Training gặp phải nhiều vấn đề. Dưới đây là ba ví dụ cụ thể về những vấn đề này.

Problem 1: Some Mistakes are Worse than Others

Mục tiêu cuối cùng là tạo ra đầu ra chất lượng, và một số dự đoán sai có thể gây ra vấn đề lớn hơn cho chất lượng đầu ra. Ví dụ, trong hệ thống nhận dạng giọng nói hoặc dịch thuật, nếu bạn muốn câu "please send this package to Pittsburgh" nhưng hệ thống lại chuyển thành "please send a package to Pittsburgh", thì đây không phải là vấn đề lớn. Tuy nhiên, nếu hệ thống chuyển thành "please send this package to Tokyo", thì đây là một vấn đề nghiêm trọng vì gói hàng có thể đến sai địa điểm.

Ngoài ra, nếu hệ thống thay đổi câu thành "bleeping send this package to Pittsburgh" thay vì "please", điều này có thể gây ra vấn đề trong hệ thống dịch vụ khách hàng, vì khách hàng có thể không hài lòng và không quay lại. Từ góc độ của phương pháp Maximum likelihood, tất cả các từ này chỉ là các token, và việc sai một token này được coi là tương đương với sai một token khác. Đây chính là một vấn đề cần được xem xét.

Problem 2: The “Gold Standard” in MLE can be Bad

Một vấn đề khác là tiêu chuẩn vàng trong Maximum Likelihood Estimation có thể không phải lúc nào cũng tốt, đôi khi không đạt được kết quả mong muốn. Các tập dữ liệu (corpora) thường chứa những đầu ra mà chúng ta không muốn mô hình ngôn ngữ tạo ra, chẳng hạn như các bình luận độc hại trên Reddit hay thông tin sai lệch.

Một khía cạnh khác mà nhiều người chưa nghĩ đến là phần lớn dữ liệu trực tuyến hiện nay được tạo tự động, ví dụ như từ dịch máy. Nhiều bản dịch trực tuyến có nguồn gốc từ Google Dịch năm 2016, khi chất lượng dịch còn kém hơn nhiều so với hiện tại, dẫn đến các bản dịch tự động có chất lượng thấp.

Problem 3: Exposure Bias

Vấn đề cuối cùng là "exposure bias". Hiện tượng này xảy ra khi phương pháp huấn luyện MLE không xét đến tính cần thiết của việc sinh nội dung và quá phụ thuộc vào ngữ cảnh chuẩn.

Khi xem xét lại phương trình MLE, phần y_<T luôn được coi là chính xác và là đầu ra tốt. Điều này dẫn đến việc mô hình phụ thuộc quá mức vào các đầu ra tốt. Một ví dụ điển hình là hiện tượng mô hình lặp lại nội dung nhiều lần. Chẳng hạn, khi bạn nói "I am going to Pittsburgh", khả năng cao là từ "Pittsburgh" sẽ xuất hiện lại trong các câu tiếp theo vì chủ đề đang nói về Pittsburgh.

Bản chất của exposure bias là do mô hình chưa từng được tiếp xúc với các lỗi trong quá khứ nên không thể xử lý chúng hiệu quả. Để khắc phục vấn đề này, chúng ta có thể sử dụng các thuật toán huấn luyện thay thế. Cụ thể là sinh ra nhiều đầu ra khác nhau, đánh giá thấp một số đầu ra và phạt mô hình khi nó tạo ra những đầu ra kém chất lượng.

Measuring how “Good” an Ouput Is

How to Measure output “Goodness”?

Có nhiều phương pháp đánh giá chất lượng đầu ra của một hệ thống. Đầu tiên là đánh giá khách quan (objective assessment), áp dụng cho các nhiệm vụ có câu trả lời đúng rõ ràng. Ngoài ra, còn có các chú thích chủ quan từ con người (human subjective annotations), nơi bạn có thể yêu cầu con người thực hiện chú thích. Một phương pháp khác là dự đoán của máy về sở thích của con người (machine prediction of human preferences). Cuối cùng, có thể đánh giá thông qua việc sử dụng trong một hệ thống khác hoặc trong một nhiệm vụ tiếp theo (downstream task).

Objective Assessment

Phương pháp đánh giá khách quan hoạt động dựa trên việc so sánh kết quả với một đáp án đã được chú thích là chính xác. Ví dụ điển hình là khi giải các bài toán toán học, trả lời các câu hỏi trắc nghiệm khách quan, hay trong các tác vụ phân loại văn bản.

Trong trường hợp các bài toán toán học, mỗi bài toán thường chỉ có một đáp án đúng duy nhất. Điều này giúp việc đánh giá trở nên đơn giản hơn. Tuy nhiên, trong thực tế, chúng ta cần xử lý nhiều loại vấn đề khác không có câu trả lời khách quan như vậy.

Human Evaluation

Trong các generation task mà không có câu trả lời khách quan, một trong những tiêu chuẩn vàng của chúng ta là đánh giá từ con người. Ví dụ, khi xử lý một đoạn văn bản nguồn như một prompt hoặc văn bản đầu vào cho dịch máy, chúng ta có thể có một hoặc nhiều giả thuyết. Sau đó, chúng ta yêu cầu người gán nhãn đưa ra điểm số hoặc thực hiện một số loại chú thích khác cho các giả thuyết này.

Human Feedback: Direct Assessment

Có nhiều phương pháp đánh giá đầu ra dựa trên phản hồi con người, một trong số đó là đánh giá trực tiếp. Đánh giá trực tiếp là người đánh giá sẽ cho điểm trực tiếp dựa trên chất lượng đầu ra. Ví dụ, nếu câu dịch "please send this package to Tokyo" được chấm 2/10, thì câu "please send a package to Tokyo" có thể được chấm 8/10. Tuy nhiên, việc này mang tính chủ quan và khó đạt được sự nhất quán nếu không có tiêu chí rõ ràng và người đánh giá có kỹ năng.

Một lợi thế của đánh giá trực tiếp là cung cấp cái nhìn tổng quan về chất lượng, nhưng nhược điểm là khó khăn trong việc duy trì sự nhất quán. Thường thì điểm số được gán dựa trên các tiêu chí mong muốn như độ trôi chảy, mức độ phản ánh chính xác ngữ nghĩa của đầu vào (adequacy), tính thực tế (factuality), và sự mạch lạc (coherence). Đặc biệt, trong dịch máy, độ trôi chảy có thể được đánh giá mà không cần nhìn vào đầu vào, nhưng để đánh giá adequacy, cần phải hiểu cả đầu vào và đầu ra.

Tính thực tế có thể được đánh giá dựa trên đầu vào cụ thể hoặc dựa trên kiến thức chung, yêu cầu người đánh giá phải kiểm tra thông tin trực tuyến. Các tiêu chí đánh giá này cũng phụ thuộc vào từng nhiệm vụ cụ thể, ví dụ như dịch máy, đối thoại, chatbot, hay tóm tắt văn bản.

Human Feedback: Preference Ratings

Phương pháp đánh giá ưu tiên (preference ratings) yêu cầu người dùng so sánh hai hoặc nhiều đầu ra từ các mô hình khác nhau hoặc từ các lần tạo khác nhau của một mô hình và xác định cái nào tốt hơn. Ví dụ, khi so sánh hai câu "Please send this package to Tokyo" và "Please send a package to Pittsburgh", nhiều người sẽ đồng ý rằng câu thứ hai tốt hơn.

Tuy nhiên, phương pháp này có hạn chế là không thể xác định rõ ràng liệu các hệ thống có thực sự tốt hay xấu. Khi so sánh các hệ thống kém, có thể tìm thấy cái tốt hơn, nhưng điều đó không có nghĩa là nó đã sẵn sàng để triển khai. Tương tự, khi so sánh các hệ thống tốt, sự khác biệt có thể rất nhỏ.

Một cách tiếp cận khác là đánh giá song song (side-by-side assessment), cho phép chấm điểm trực tiếp từng đầu ra với các giá trị thập phân, giúp phân biệt rõ ràng hơn giữa các hệ thống.

Một vấn đề khác của đánh giá ưu tiên là giới hạn số lượng hệ thống mà con người có thể so sánh trước khi bị quá tải. Để giải quyết, có thể sử dụng các thuật toán xếp hạng như ELO hoặc TrueSkill (Sakaguchi et al. 2014), vốn được phát triển để xếp hạng người chơi cờ vua hoặc game. Các thuật toán này sử dụng kết quả từ các trận đấu cặp để đưa ra xếp hạng tổng thể cho các hệ thống. Đây là một phương pháp hữu ích để có được xếp hạng hệ thống mà chỉ cần thực hiện các đánh giá cặp đôi.

Human Feedback: Error Annotation

Phương pháp đánh giá đầu ra dựa trên phản hồi từ con người cuối cùng là chú thích lỗi. Phương pháp này đặc biệt hữu ích trong nhiều trường hợp, đặc biệt là trong dịch máy.

Một công cụ phổ biến trong dịch máy là các chỉ số chất lượng đa chiều (Multi-dimensional Quality Metrics). Công cụ này cho phép chú thích các đoạn văn bản trong đầu ra, mỗi đoạn được đánh giá theo mức độ nghiêm trọng và loại lỗi. Có khoảng tám loại lỗi khác nhau, chẳng hạn như lỗi vi phạm quy ước ngôn ngữ hoặc lỗi về độ chính xác. Các lỗi này có thể được phân loại từ nhỏ đến lớn, với sự phân biệt quan trọng giữa lỗi nhỏ và lớn.

Ưu điểm của phương pháp này là cung cấp phản hồi chi tiết hơn, giúp xác định hệ thống có nhiều lỗi về độ chính xác hay lỗi về quy ước ngôn ngữ. Ngoài ra, nó còn giúp tăng tính nhất quán trong đánh giá, vì việc xác định một từ có đúng hay không thường dễ hơn so với việc đánh giá tổng thể một đầu ra. Tuy nhiên, nhược điểm là quá trình này có thể rất tốn thời gian, đặc biệt khi phải chú thích từng lỗi trong các đầu ra dài.

An Alternative: Automatic Evaluation

Ngoài việc thu thập phản hồi từ con người, chúng ta còn có phương pháp đánh giá tự động (automatic evaluation). Phương pháp này hoạt động dựa trên ba thành phần chính: nguồn đầu vào, các giả thuyết, và một hệ thống tự động tạo điểm số. Trong một số trường hợp, chúng ta cũng có thể sử dụng một đầu ra tham chiếu - là một tiêu chuẩn vàng được tạo bởi con người để làm chuẩn so sánh.

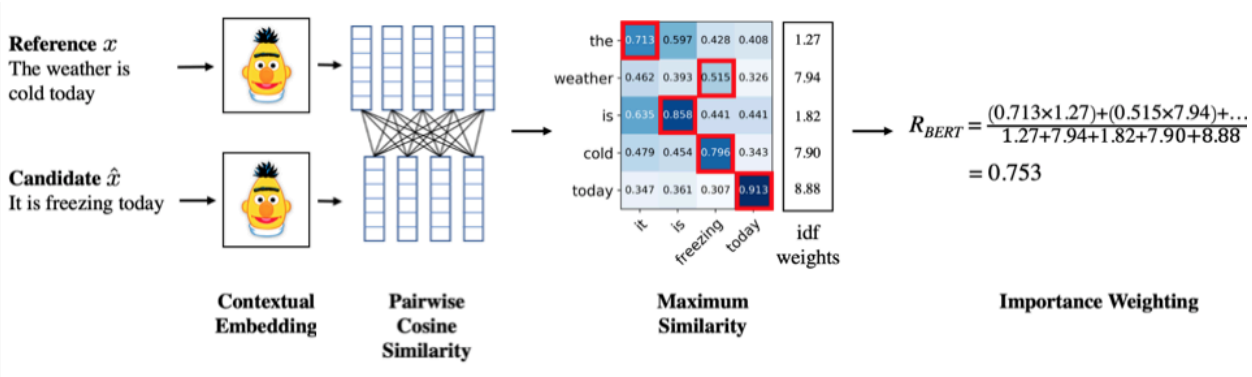
Mục tiêu chính của đánh giá tự động là dự đoán được sở thích hoặc điểm số mà con người sẽ đánh giá, vì đánh giá của con người vẫn được xem là tiêu chuẩn quan trọng nhất. Thuật ngữ này được gọi khác nhau tùy lĩnh vực: trong dịch máy và tóm tắt văn bản, nó được gọi là "automatic evaluation"; trong khi đó, trong lĩnh vực học tăng cường, chatbot hay AGI, nó thường được gọi là "reward model" (mô hình phần thưởng) - phản ánh góc nhìn về việc học từ

phản hồi. Tuy nhiên, bản chất của các phương pháp này đều giống nhau: dự đoán mức độ tốt của đầu ra và xác định mức độ thưởng cho mô hình khi tạo ra đầu ra đó. Có 3 paradigms khác nhau cho việc đánh giá tự động.

Embedding-based Evaluation

Đầu tiên, đánh giá dựa trên embedding là một phương pháp tính toán không giám sát, dựa trên sự tương đồng giữa embedding của đầu ra mà mô hình tạo ra và một đầu ra tham chiếu mà bạn đã tạo. Ví dụ, nếu đầu ra tham chiếu là "the weather is cold today" và đầu ra của mô hình là "it is freezing today", thì đây có thể được coi là một đầu ra khá tốt.

Chúng ta sẽ sử dụng một mô hình embedding, ví dụ như BERT, để tạo embedding cho mỗi token trong chuỗi. Giả sử có năm token trong chuỗi tham chiếu và bốn token trong chuỗi đầu ra, chúng ta sẽ có năm embedding và bốn embedding tương ứng. Tiếp theo, chúng ta tính toán độ tương đồng cosine từng cặp giữa tất cả các embedding này, tạo ra một ma trận độ tương đồng cosine.



Sau đó, chúng ta có thể lấy giá trị tương đồng lớn nhất (ARG Max) theo hàng hoặc cột. Các hàng tương ứng với các token trong tham chiếu, do đó, việc tìm kiếm token tương tự trong đầu ra cho mỗi token trong tham chiếu giống như một phương pháp dựa trên recall. Ngược lại, nếu xem xét các cột, đây là một metric dựa trên precision, vì nó cho biết có bao nhiêu token trong đầu ra có sự tương đồng với tham chiếu.

Chúng ta có thể tính toán recall và precision cho tất cả các token và đưa vào một công thức giống như F-measure. Ngoài ra, có thể sử dụng trọng số tf-idf để tăng trọng số cho các từ ít xuất hiện, vì chúng thường là các từ mang nội dung quan trọng. Ví dụ, sai lầm khi chuyển từ "Pittsburgh" thành "Tokyo" sẽ nghiêm trọng hơn so với sai lầm từ "this" thành "um".

Regression-based Evaluation

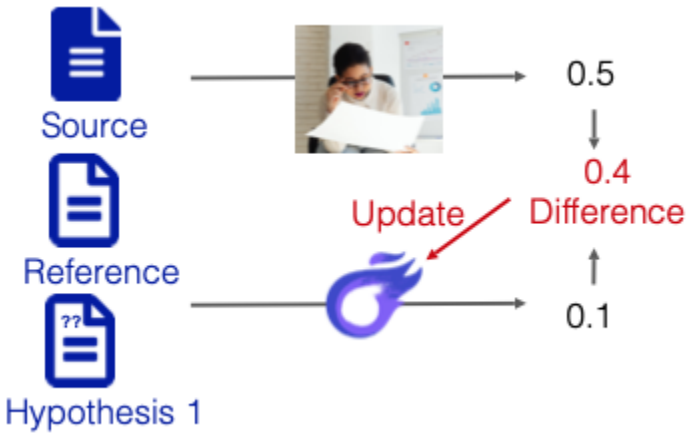
Phương pháp đánh giá dựa trên hồi quy thường được sử dụng trong môi trường học có giám sát. Để thực hiện phương pháp này, chúng ta cần thu thập một lượng lớn đánh giá từ con người. Những đánh giá này có thể được thực hiện theo hai cách:

- Đánh giá trực tiếp: Người đánh giá cho điểm trực tiếp
- Đánh giá theo cặp (Pairwise judgments): So sánh từng cặp kết quả

Tùy thuộc vào phương pháp đánh giá mà ta sẽ sử dụng hàm mất mát (loss) khác nhau:

- Với đánh giá trực tiếp: Sử dụng hàm mất mát dựa trên hồi quy như minimum squared error
- Với đánh giá theo cặp: Sử dụng hàm mất mát dựa trên xếp hạng, tăng trọng số cho các kết quả có điểm cao và giảm trọng số cho các kết quả có điểm thấp

Một ví dụ điển hình là mô hình COMET (Rei et al. 2020) - từng là SOTA trong lĩnh vực dịch máy. COMET hoạt động hiệu quả nhờ có được một lượng lớn dữ liệu đánh giá được tích lũy qua nhiều năm trong lĩnh vực dịch máy. Quy trình hoạt động của mô hình như sau:



- Thu thập dữ liệu đánh giá từ con người
- So sánh kết quả đầu ra của mô hình COMET với đánh giá của con người
- Tính toán sự khác biệt và cập nhật tham số mô hình

Tuy nhiên, phương pháp này có một hạn chế lớn: nó đòi hỏi một lượng lớn dữ liệu huấn luyện. Đối với nhiều tác vụ khác, việc không có đủ dữ liệu huấn luyện khiến cho việc áp dụng phương pháp này trở nên kém khả thi.

QA-based Evaluation

Gần đây, xu hướng đánh giá mới đang được áp dụng là sử dụng mô hình ngôn ngữ để đánh giá chất lượng đầu ra. Một ví dụ điển hình là GEMBA (Kocmi and Federmann 2023). Cách thức hoạt động của nó như sau:

- Yêu cầu GPT-4 chấm điểm bản dịch từ ngôn ngữ nguồn sang ngôn ngữ đích
- Thang điểm từ 0-100, trong đó: 0: không giữ được ý nghĩa và 100: hoàn hảo về nghĩa và ngữ pháp
- Đầu vào bao gồm: văn bản nguồn, bản dịch tham chiếu (nếu có), và bản dịch cần đánh giá

- Đầu ra là một điểm số đánh giá

Mặc dù phương pháp này cho kết quả khá tốt, đặc biệt khi sử dụng mô hình ngôn ngữ mạnh, nhưng nó vẫn tồn tại một số hạn chế:

1. Tính không ổn định:

- Kết quả phụ thuộc nhiều vào prompt được sử dụng
- Nhiều người dùng GPT-4 mà không kiểm chứng độ chính xác của đánh giá
- Chất lượng đánh giá có thể dao động từ rất tốt đến rất kém tùy theo tác vụ

2. Các trường hợp cần đặc biệt cẩn trọng:

- Khi so sánh GPT-4 với chính nó: GPT-4 có xu hướng thiên vị, đánh giá cao hơn cho các kết quả do chính nó tạo ra
- Khi tối ưu hóa đầu ra bằng RLHF: Theo định luật Goodhart, khi ta bắt đầu tối ưu hóa theo một tiêu chí đánh giá, tiêu chí đó sẽ trở nên kém hiệu quả

Một cách tiếp cận mới trong đánh giá QA-based là tập trung vào việc xác định các lỗi cụ thể thay vì cho điểm. Nghiên cứu của Patrick Fernandez chỉ ra rằng:

- Yêu cầu mô hình chỉ ra các lỗi cụ thể trong đầu ra thay vì đưa ra điểm số
- Phương pháp này cho kết quả nhất quán hơn
- Tương tự như con người, khi được yêu cầu xác định từng lỗi cụ thể, GPT-4 cũng cho kết quả đánh giá nhất quán hơn

Có một hiện tượng thú vị là các mô hình thường có xu hướng thiên vị tới kết quả do chính nó tạo ra. Điều này có thể được giải thích bởi 3 nguyên nhân chính:

- Mô hình luôn tạo ra output từ không gian xác suất cao của chính nó.
- Các output có xác suất cao thường liên quan đến chất lượng tốt, vì điều này cho thấy chúng gần với dữ liệu huấn luyện.
- Khi mô hình nhận biết nó đang ở trong vùng xác suất cao của không gian embedding, nó có xu hướng đưa ra đánh giá tích cực về chất lượng output.

Ba yếu tố này kết hợp lại khiến mô hình có thiên hướng đánh giá cao hơn các output do chính nó tạo ra so với output từ các mô hình khác. Tuy nhiên, đây mới chỉ là giả thuyết ban đầu và cần thêm nghiên cứu thực nghiệm để kiểm chứng, đặc biệt là về mối liên hệ giữa vùng xác suất cao và xu hướng đánh giá tích cực của mô hình.

Meta-evaluation of Metrics

Khi chúng ta nói về việc một tiêu chí đánh giá có tốt hay không, điều này thường được xác định thông qua một quá trình gọi là "meta evaluation". Meta evaluation là quá trình đánh giá các tiêu chí đánh giá, và cách thực hiện thường là so sánh giữa các điểm số do con người đánh giá và các điểm số tự động. Chúng ta thường tính toán một số dạng tương quan giữa các điểm số

này, ví dụ như "rank correlations" (tương quan thứ hạng) như Pearson's correlation hoặc Kendall's Tau.

Mức độ tương quan cao giữa điểm số tự động và điểm số của con người cho thấy tiêu chí đánh giá tự động có độ tin cậy cao. Ngoài ra, nếu bạn muốn kiểm tra xem một mô hình có phù hợp với các ưu tiên cặp đôi của con người hay không, bạn có thể tính toán độ chính xác của các ưu tiên cặp đôi. Một cách khác là tính toán sai số tuyệt đối giữa các đánh giá nếu bạn muốn biết sai số tuyệt đối có phù hợp hay không.

Đây là những phương pháp hữu ích nếu bạn muốn sử dụng một tiêu chí đánh giá nhưng không chắc chắn về độ tin cậy của nó. Bạn nên kiểm tra xem các tác giả đã thực hiện loại meta evaluation này chưa. Nếu chưa, bạn nên cẩn trọng; nếu đã thực hiện, bạn có thể yên tâm hơn.

Thông thường, để thực hiện meta evaluation một cách đáng tin cậy, cần có một tập dữ liệu đủ lớn. Các tập dữ liệu thường được sử dụng bao gồm các nhiệm vụ chia sẻ của WMT hoặc các tập dữ liệu khác như SumEval.

Use in a DownStream System

Ngoài việc đánh giá dựa trên phản hồi của con người, chúng ta còn có hai phương pháp đánh giá quan trọng khác: đánh giá nội tại (intrinsic evaluation) và đánh giá ngoại tại (extrinsic evaluation):

- Đánh giá nội tại: Tập trung vào việc đánh giá chất lượng của đầu ra một cách trực tiếp. Thường được thực hiện thông qua đánh giá của con người về mức độ tốt của kết quả.
- Đánh giá ngoại tại: Đánh giá chất lượng đầu ra dựa trên tính hữu ích của nó. Ví dụ: Đánh giá bản tóm tắt của LLMs thông qua độ chính xác của việc trả lời câu hỏi.

Ví dụ thực tế về đánh giá ngoại tại trong lĩnh vực y tế:

- Sử dụng LLM để tạo bản tóm tắt tiền sử bệnh của bệnh nhân.
- Bản tóm tắt này được bác sĩ sử dụng để đưa ra chẩn đoán.
- Việc đánh giá có thể dựa trên độ chính xác của chẩn đoán cuối cùng sau khi có kết quả xét nghiệm

Thách thức trong đánh giá ngoại tại:

- Khó tách biệt được ảnh hưởng của các thành phần khác nhau trong hệ thống
- Ví dụ: Nếu mô hình QA kém, khó xác định được liệu vấn đề nằm ở bản tóm tắt hay ở mô hình QA

Xử lý sự đa dạng trong đánh giá của con người:

1. Chuẩn hóa điểm số:
- Sử dụng Z-score để chuẩn hóa điểm của từng người đánh giá
- Mục đích: loại bỏ sự khác biệt về độ nghiêm khắc giữa các người đánh giá
- Chuẩn hóa về mean = 0 và variance = 1

2. Xử lý bất đồng:
- Có thể lấy trung bình các đánh giá
- Hoặc loại bỏ các ví dụ có sự bất đồng cao giữa người đánh giá

Ví dụ đặc biệt trong Code Generation:

- Nhiều người chỉ tập trung vào đánh giá dựa trên việc thực thi code
- Tuy nhiên, cần quan tâm đến các khía cạnh khác như khả năng đọc hiểu
- Cần kết hợp cả đánh giá nội tại và ngoại tại để có đánh giá toàn diện

Error and Risk

Error

Trong phần này, chúng ta sẽ thảo luận về lỗi và rủi ro. Đầu tiên, cách chúng ta tính toán lỗi là tạo ra một đầu ra và đánh giá mức độ "tệ" (badness) của nó. Việc tạo ra đầu ra có thể thông qua các phương pháp như argmax, sampling, hoặc các phương pháp khác.

$$\widehat{Y} = argmax_{\tilde{Y}} P\left(\tilde{Y}|X\right)$$

Sau đó, chúng ta tính toán mức độ "tệ" của đầu ra, có thể là một phép đo trực tiếp hoặc thông qua công thức như 1 trừ đi giá trị đánh giá. Đây được định nghĩa là lỗi.

$$error(Y, \widehat{Y}) = 1 - eval(Y, \widehat{Y})$$

Mục tiêu chính là giảm thiểu lỗi, vì cuối cùng, chúng ta sẽ triển khai một hệ thống chỉ tạo ra một đầu ra duy nhất và mong muốn đầu ra đó đạt chất lượng tốt nhất có thể.

Problem: Argmax is Non-differentiable

Tuy nhiên, việc tối thiểu hóa Error gặp phải một vấn đề lớn: không có cách đơn giản để tối ưu hóa giá trị này, đặc biệt trong các bài toán sinh văn bản và ngay cả với các bài toán phân loại. Lý do chính là khi xét đến bề mặt của Error, tại một số điểm sẽ xuất hiện các phần không khả vi (non-differentiable) - điều này xảy ra khi thực hiện các phép toán như argmax hoặc sampling. Do đó, chúng ta không thể áp dụng các phương pháp tối ưu dựa trên gradient để giải quyết vấn đề này.

Risk

Để giải quyết vấn đề trên, chúng ta thường sử dụng một phương pháp khác gọi là Risk (Rủi ro). Risk được định nghĩa là giá trị kỳ vọng của Error cho output. Điểm khác biệt quan trọng là trong hàm mục tiêu của Risk có chứa một xác suất, và xác suất này là khả vi (differentiable). Nhờ vậy, chúng ta có thể áp dụng các phương pháp tối ưu dựa trên gradient.

$$risk(X, Y, \theta) = \sum_{\tilde{Y}} P\left(\tilde{Y}|X; \theta\right) error\left(Y, \tilde{Y}\right)$$

Tuy nhiên, với bài toán sinh văn bản, phương pháp này vẫn gặp khó khăn do tổng trong công thức là không khả thi để tính toán. Lý do là số lượng output tiềm năng quá lớn. Ví dụ, với độ dài văn bản là 50 và kích thước từ điển là 30,000, chúng ta sẽ có 30,000^50 khả năng.

Nhiều phương pháp học đã được phát triển để tối thiểu hóa Risk, như Minimum Risk Training và Reinforcement Learning (đặc biệt là các mô hình Policy Gradient). Risk là một khái niệm cơ bản và quan trọng, giúp ta dễ dàng tiếp cận các phương pháp học phức tạp hơn sau này.

Sampling for Tractability

Để tối ưu hóa rủi ro, thay vì tính tổng trên tất cả các đầu ra có thể, chúng ta chỉ tính tổng trên một số lượng nhỏ các đầu ra có thể và sau đó chuẩn hóa để tổng này bằng một. Bộ chuẩn hóa này thực chất là tổng của tất cả các xác suất mà chúng ta có. Các mẫu này có thể được tạo ra bằng cách sử dụng phương pháp lấy mẫu hoặc tìm kiếm n-best.

$$risk(X,\ Y,\ \theta) = \sum_{\tilde{Y} \in S} \frac{P\left(\tilde{Y}|X;\theta\right)}{Z}\ error\left(Y,\ \tilde{Y}\right)$$

Để thực hiện huấn luyện tối thiểu rủi ro một cách chính xác, phương pháp “ancestral sampling” được đề xuất. Tuy nhiên, vấn đề phát sinh khi lấy mẫu từ mô hình ngôn ngữ với temperature bằng 1, vì nó có thể tạo ra nhiều đầu ra không tốt. Do đó, có thể sử dụng các phương pháp thay thế như tìm kiếm n-best hoặc lấy mẫu với temperature khác 1 để tạo ra danh sách các giả thuyết có thể.

Khi lấy mẫu với temperature khác 1 và có khả năng nhận được nhiều đầu ra, nên loại bỏ các mẫu trùng lặp hoặc lấy mẫu không thay thế để tránh làm sai lệch các phương trình. Đây là một ví dụ đơn giản về cách thực hiện huấn luyện tối thiểu rủi ro.

Reinforcement Learning Basics: Policy Gradient

(Review of Karpathy 2016)

What is Reinforcement Learning?

Hiện nay, học tăng cường đang trở thành phương pháp được ưa chuộng nhất để học từ phản hồi của con người. Mặc dù có các phương pháp thay thế như học dựa trên biên (margin-based losses).

Học tăng cường là một phương pháp học mà trong đó chúng ta có một môi trường (X), khả năng thực hiện các hành động (A), và nhận được phần thưởng trễ (R). Một ví dụ điển hình là trò chơi Pong: X là hình ảnh quan sát, A là hành động lên hoặc xuống, và R là kết quả thắng/thua cuối cùng.

Trong bối cảnh xử lý ngôn ngữ tự nhiên, chúng ta có thể áp dụng học tăng cường như thế nào? Ví dụ, trong một tác vụ hội thoại, X có thể là lịch sử hội thoại, A là từ tiếp theo được tạo ra, và R là phần thưởng nhận được tại một thời điểm nào đó, không nhất thiết ngay sau khi tạo từ.

Đối với các tác vụ khác như sinh mã, X có thể là trình biên dịch và ngữ cảnh mã xung quanh, A là từng token trong mã, và R là phần thưởng sau một chuỗi hành động dài. Phần thưởng này có thể đến từ trình biên dịch, mô hình đánh giá độ dễ đọc của mã, hoặc tốc độ thực thi.

Một điểm thú vị của học tăng cường là khả năng sáng tạo trong việc thiết kế phần thưởng R, điều này không dễ thực hiện nếu chỉ dùng phương pháp maximum likelihood. Nó cho phép chúng ta tạo ra phần thưởng phù hợp với mục tiêu đầu ra mong muốn.

Why Reinforcement Learning in NLP?

Học tăng cường có thể được áp dụng trong nhiều tình huống trong NLP, và dưới đây là ba lý do chính:

- Kịch bản đối thoại: Trong các hệ thống đối thoại, người dùng thường đưa ra phản hồi dưới dạng "thích" hoặc "không thích". Đây là một ví dụ điển hình của học tăng cường, nơi mà phần thưởng được nhận muộn, phụ thuộc vào phản hồi của người dùng trong suốt cuộc đối thoại.

- Trung tâm cuộc gọi: Học tăng cường đã được sử dụng trong các hệ thống đối thoại tại trung tâm cuộc gọi. Nếu một cuộc gọi được xử lý thành công mà không cần sự can thiệp của nhân viên, hệ thống sẽ nhận được phần thưởng lớn. Ngược lại, nếu cuộc gọi phải chuyển cho nhân viên hoặc người gọi tức giận và cúp máy, phần thưởng sẽ giảm hoặc trở thành tiêu cực.

- Biến ẩn và Chain of Thought: Trong một số trường hợp, các biến ẩn có thể ảnh hưởng đến kết quả đầu ra. Một Chain of Thought không tốt vẫn có thể dẫn đến câu trả lời đúng, điều này khiến việc đánh giá chất lượng Chain of Thought trở thành một bài toán học tăng cường. Hơn nữa, một số chỉ số đánh giá ở cấp độ chuỗi yêu cầu toàn bộ chuỗi được tạo ra trước khi có thể tối ưu hóa, điều này cũng tạo ra cơ hội cho việc áp dụng học tăng cường.

Supervised MLE

Trong học có giám sát, hàm mất mát MLE (Maximum Likelihood Estimation) thực chất là logarit của xác suất:

$$l_{super}(Y,\ X) = -\ log\ P(Y|\ X)$$

Trong bối cảnh học tăng cường, phương pháp này còn được gọi là học bắt chước (imitation learning), bởi về bản chất, mô hình đang học cách thực hiện các hành động bằng cách bắt chước từ một "người thầy". Tuy nhiên, cần lưu ý rằng học bắt chước không chỉ giới hạn ở phương pháp học có giám sát MLE - còn nhiều biến thể khác của học bắt chước, nhưng đây là một trong những cách tiếp cận phổ biến.

Self Training

Tiếp theo, chúng ta sẽ nói về kỹ thuật tự huấn luyện (self-training). Ý tưởng cốt lõi của phương pháp này là lấy mẫu hoặc tìm giá trị lớn nhất từ mô hình hiện tại.

$$\widehat{Y} \sim P(Y|X) \quad \text{or} \quad \widehat{Y} \sim argmax_Y P(Y|X)$$

Sau đó sử dụng các mẫu này để maximize likelihood. Khác với maximize likelihood thông thường - vốn dựa trên đầu ra chuẩn, phương pháp này lại sử dụng chính đầu ra của mô hình làm chuẩn.

$$l_{self}(X) = -\ log\ P\left(\widehat{Y}|\ X\right)$$

Tuy nhiên, cách tiếp cận này có một nhược điểm đáng kể: khi không có cơ chế kiểm soát chất lượng đầu ra, mô hình sẽ tối ưu hóa cả đầu ra tốt lẫn xấu. Điều này có nghĩa là nếu mô hình tạo ra đầu ra kém chất lượng, nó sẽ càng củng cố những lỗi sai đã có. Mặc dù vậy, trong một số trường hợp, self-training vẫn có thể cải thiện độ chính xác, đặc biệt khi mô hình ban đầu có độ chính xác trên 50%. Điều này có thể đạt được nhờ vào các cơ chế điều chuẩn ngầm và kỹ thuật early stopping.

Có một số phương pháp thay thế hiệu quả hơn self-training:

- Co-training (Blum and Mitchell 1998): sử dụng nhiều mô hình và chỉ chọn những câu mà các mô hình đều đồng thuận.
- Một phương pháp khác là thêm nhiễu vào đầu vào để khớp với nhiễu trong đầu ra (He et al. 2020), chẳng hạn như sử dụng word-based dropout.

Tóm lại, mặc dù self-training có thể hữu ích, nhưng sẽ có những lựa chọn tốt hơn nếu bạn có thể xây dựng được hàm reward.

Policy Gradient/ REINFORCE

Việc tối ưu hoá policy là một phần quan trọng trong học tăng cường. Một trong những phương pháp đơn giản nhất là Policy Gradient, hay REINFORCE. Cốt lõi của phương pháp này là thêm một thành phần để điều chỉnh hàm mất mát theo giá trị phần thưởng. Thay vì chỉ áp dụng self-training đơn thuần, ta nhân nó với một giá trị phần thưởng. Cách tiếp cận này cho phép tăng khả năng xuất hiện của những đầu ra có phần thưởng cao và giảm khả năng của những đầu ra có phần thưởng thấp.

$$l_{REINFORCE}\left(X,\ \widehat{Y}\right) = -\ R\left(Y,\ \widehat{Y}\right)\ log\ P\left(\widehat{Y}|\ X\right)$$

Một điểm thú vị là phương pháp này có thể tương đương với Maximum Likelihood Estimation trong một số điều kiện đặc biệt. Cụ thể, khi sử dụng hàm đánh giá nhị phân (zero-one loss), tức là gán giá trị phần thưởng bằng 1 khi đầu ra dự đoán (Y hat) hoàn toàn khớp với đầu ra thật (Y), và bằng 0 trong các trường hợp còn lại, phương pháp này sẽ tương đương với MLE.

Tuy nhiên, điều này cũng cho thấy ưu điểm của Policy Gradient/REINFORCE: nó linh hoạt hơn nhiều so với MLE. Thay vì chỉ giới hạn ở giá trị 0 và 1 cho việc khớp hoàn toàn, ta có thể sử

dụng các hàm phần thưởng đa dạng hơn, ví dụ như cho điểm một phần (partial credit) hoặc chấp nhận nhiều đầu ra đúng tiềm năng khác nhau. Điều này mở ra nhiều khả năng ứng dụng phong phú hơn trong thực tế.

Credit Assignment for Rewards

Một thách thức lớn trong các phương pháp học tăng cường là làm sao xác định được chính xác hành động nào dẫn đến phần thưởng. Kịch bản lý tưởng nhất là sau mỗi hành động (ví dụ như sinh ra một token), chúng ta nhận được phản hồi trực tiếp từ người dùng (như thumbs up hoặc thumbs down) để biết họ thích hay không thích token đó. Tuy nhiên, điều này không khả thi trong thực tế - chẳng hạn, người dùng sẽ không thể đánh giá từng token khi sử dụng ChatGPT.

Thực tế, chúng ta thường chỉ nhận được phần thưởng sau khi đã thực hiện một chuỗi nhiều hành động, và khó xác định được hành động cụ thể nào đóng góp vào phần thưởng đó. Hiện nay có một số cách để giải quyết vấn đề này:

- Cách phổ biến nhất là "không giải quyết" - tức là kỳ vọng thuật toán tối ưu hóa sẽ tự động phân bổ credit. Điều này thường được thực hiện bằng cách gán phần thưởng bằng nhau cho mỗi token trong đầu ra.

- Sử dụng phần thưởng giảm dần từ các sự kiện tương lai. Ví dụ với chatbot có 20 lượt tương tác: nếu người dùng cho thumbs up ở lượt thứ 20, ta có thể gán phần thưởng 1.0 cho lượt thứ 19, 0.5 cho lượt thứ 18, 0.25 cho lượt thứ 17, và cứ thế. Logic ở đây là cảm xúc tích cực của người dùng có khả năng bị ảnh hưởng nhiều hơn bởi những tương tác gần đây.

Về nguồn của phần thưởng, có thể là:
- Phản hồi trực tiếp từ người dùng (thumbs up/down)
- Từ một mô hình phần thưởng được huấn luyện trước
- Lý thuyết thì có thể học đồng thời mô hình phần thưởng cùng với mô hình chính

Mặc dù việc triển khai cơ bản của phương pháp này - lấy mẫu và tối ưu hóa hàm mục tiêu - không quá phức tạp nếu đã có nguồn phần thưởng, học tăng cường vẫn có thể rất bất ổn định. Để đạt hiệu quả tốt, chúng ta cần áp dụng thêm một số kỹ thuật đặc biệt.

Stabilizing Reinforcement Learning

Problems with Reinforcement Learning

Học tăng cường thường gặp nhiều vấn đề về tính ổn định. Có hai nguyên nhân chính dẫn đến tình trạng này:

Thứ nhất, khi huấn luyện, mô hình chỉ lấy mẫu và tính toán dựa trên một đầu ra cụ thể, trong khi tồn tại vô số đầu ra khác có thể được tối ưu hóa. Điều này khác biệt so với phương pháp

Maximum Likelihood Estimation (MLE). Trong MLE, mô hình luôn so sánh đầu ra chuẩn với tất cả các đầu ra khả dĩ trong không gian, từ đó tăng trọng số cho đầu ra chuẩn và giảm trọng số cho các đầu ra còn lại. Ngược lại, trong học tăng cường, việc chỉ dựa vào một đầu ra được lấy mẫu - vốn có thể sai - tạo ra sự bất ổn định. Vấn đề này đặc biệt nghiêm trọng khi làm việc với không gian đầu ra lớn, chẳng hạn như toàn bộ từ vựng của mô hình.

Thứ hai, việc sử dụng phần thưởng âm (negative rewards) cũng gây ra bất ổn định. Khi áp dụng phần thưởng âm, trọng số của một chuỗi đầu ra cụ thể sẽ bị giảm xuống - điều này có thể hữu ích trong một số trường hợp, ví dụ như khi mô hình tạo ra nội dung độc hại. Tuy nhiên, bên cạnh nội dung độc hại đó còn có số lượng lớn các đầu ra vô nghĩa không phải là câu tiếng Anh hợp lệ. Điều này có thể khiến mô hình dần đi chệch khỏi phân phối mô hình ngôn ngữ tự nhiên ban đầu của nó.

Đây là những thách thức lớn trong học tăng cường, và các nhà nghiên cứu đã phát triển nhiều chiến lược khác nhau để ổn định quá trình học này.

Pre-training with MLE (Ranzato et al. 2016)

Đầu tiên, một chiến lược để ổn định quá trình học tăng cường là pre-training với Maximum Likelihood Estimation (MLE). Quy trình thực hiện khá đơn giản: Đầu tiên, ta bắt đầu với một mô hình đã được pre-train, sau đó mới chuyển sang giai đoạn học tăng cường.

Hiện nay, đây được xem là phương pháp hiển nhiên và không ai làm việc với mô hình ngôn ngữ lại bỏ qua bước này. Tuy nhiên, chiến lược này cũng có những hạn chế đáng chú ý:

- Chỉ áp dụng được trong những tình huống có thể thực hiện MLE. Ví dụ, nó không hoạt động hiệu quả khi bạn cần dự đoán các biến ẩn không có trong không gian dữ liệu gốc.

- Gặp khó khăn trong một số trường hợp thực tế. Ví dụ, khi bạn muốn xây dựng một chatbot phục vụ khách hàng cho một công ty với danh mục sản phẩm mà mô hình ngôn ngữ chưa từng "nhìn thấy" trước đó. Nếu mô hình không được cung cấp thông tin về danh mục sản phẩm (chẳng hạn thông qua RAG), nó sẽ phải tìm kiếm trong một không gian quá rộng và khó có thể hội tụ với các mục tiêu mô hình hóa ngôn ngữ đề ra.

Để khắc phục những hạn chế này, bạn cần tạo ra ít nhất một số dữ liệu huấn luyện có giám sát để training với MLE.

Regularization to an Existing Model (e.g. Schulman et al. 2017)

Sau pre-training, bước tiếp theo để ổn định mô hình trong học tăng cường là áp dụng điều chuẩn (regularization) với một mô hình có sẵn. Mục đích chính là ngăn mô hình mới đi quá xa so với mô hình gốc. Ví dụ, khi bạn gán phần thưởng âm cho các phát ngôn độc hại, bạn vẫn muốn mô hình duy trì khả năng sinh ngôn ngữ tự nhiên của nó. Có hai phương pháp điều chuẩn phổ biến:

1. **Điều chuẩn KL:** Phương pháp này bao gồm hai thành phần chính:

$$\ell_{regularized} = \underbrace{\frac{P(\hat{Y}|X;\theta)}{P(\hat{Y}|X;\theta_{old})}R(Y,\hat{Y})}_{\text{improve reward}} - \underbrace{\beta \mathrm{KL}\left[P(\cdot|X;\theta_{\mathrm{old}}), P(\cdot|X;\theta)\right]}_{\text{keep model similar}}$$

- Thành phần thứ nhất tập trung vào cải thiện phần thưởng: So sánh xác suất của mô hình cũ và mô hình mới kết hợp với tín hiệu phần thưởng để tăng khả năng sinh ra các chuỗi có phần thưởng cao.
- Thành phần thứ hai là điều chuẩn KL: Giữ cho phân phối xác suất của mô hình mới tương đồng với mô hình cũ.
- Tham số beta trong điều chuẩn KL có thể được điều chỉnh để kiểm soát mức độ tương đồng mong muốn giữa hai mô hình.

2. **PPO (Proximal Policy Optimization):**

- PPO sử dụng phương pháp clipping (cắt giới hạn) cho các đầu ra.
- Phương pháp định nghĩa một tỷ lệ thể hiện mức độ mô hình mới tăng trọng số cho các chuỗi có phần thưởng cao.

$$rat(Y,\ X)\ =\ P(Y|X;\ \theta)/P\left(Y|X;\ \theta_{old}\right)$$

- Bằng cách lấy giá trị nhỏ nhất giữa tỷ lệ gốc và phiên bản bị cắt giới hạn, PPO khuyến khích mô hình:

$$\ell_{PPO} = \min(\mathrm{rat}(\hat{Y},X)R(\hat{Y}), \underbrace{\mathrm{clip}(\mathrm{rat}(\hat{Y},X),1+\epsilon,1-\epsilon)R(\hat{Y})}_{\text{don't reward large jumps}})$$

- Tránh những thay đổi quá lớn trong không gian tìm kiếm
- Quay trở lại mô hình gốc nếu không gian mới cho kết quả kém hơn
- Ngăn chặn việc sinh ra các đầu ra vô nghĩa có phần thưởng thấp

PPO từng rất phổ biến, nhưng gần đây nhiều người đã chuyển sang sử dụng các chiến lược thay thế dựa trên điều chuẩn KL. Cả hai phương pháp đều có ưu điểm riêng, trong đó KL regularization được đánh giá là đơn giản hơn về mặt khái niệm.

Các phương pháp này đã được tích hợp sẵn trong nhiều thư viện phổ biến như Hugging Face TRL (Transformer Reinforcement Learning).

Adding a Baseline

Trong quá trình dịch câu, việc thiết lập một Baseline (mức chuẩn) cho phần thưởng là rất quan trọng. Ý tưởng cơ bản là chúng ta có những kỳ vọng về phần thưởng cho từng câu. Ví dụ, khi

dịch câu "This is an easy sentence" và câu "buffalo buffalo buffalo", câu đầu tiên có thể nhận được phần thưởng 0.8, trong khi câu thứ hai chỉ nhận được 0.3.

Tuy nhiên, nếu câu đầu tiên thực sự dễ và câu thứ hai lại khó, thì việc nhận phần thưởng 0.8 cho câu dễ là không hợp lý, trong khi phần thưởng 0.3 cho câu khó lại cho thấy chúng ta đang đi đúng hướng.

	Reward	Baseline	B-R
"This is an easy sentence"	0.8	0.95	-0.15
"Buffalo Buffalo Buffalo"	0.3	0.1	0.2

Để điều chỉnh phần thưởng một cách hợp lý, chúng ta sử dụng công thức: phần thưởng - Baseline. Kết quả sẽ cho ra giá trị âm cho câu đầu tiên và giá trị dương cho câu thứ hai. Mục tiêu là dự đoán trước độ khó của từng ví dụ và điều chỉnh phần thưởng dựa trên đó. Như vậy, việc có một mô hình Baseline để dự đoán độ khó là rất cần thiết để cải thiện quá trình đánh giá.

$$l_{baseline}(X) = -\left(R(\widehat{Y}, Y) - B(\widehat{Y})\right) log\, P(\widehat{Y}|\, X)$$

Calculating Baselines

Khi triển khai Baseline, mục tiêu chính là giảm phương sai của phần thưởng và giúp quá trình học ổn định hơn. Có hai phương pháp phổ biến:

1. **Dự đoán phần thưởng cuối cùng:** Phương pháp này sử dụng một mô hình để dự đoán phần thưởng cuối cùng với một số đặc điểm quan trọng:

- Mô hình chỉ xem xét dữ liệu đầu vào hoặc các trạng thái trung gian
- Không được sử dụng kết quả dự đoán thực tế của mô hình chính
- Có thể áp dụng ở nhiều cấp độ:
  - Cấp độ câu: một baseline cho mỗi câu
  - Cấp độ decoder: baseline cho từng trạng thái của decoder
- Việc triển khai được thực hiện bằng cách huấn luyện một mô hình hồi quy dựa trên các phần thưởng nhận được. Điểm quan trọng là baseline không được phép sử dụng bất kỳ dự đoán thực tế nào từ mô hình chính - nếu không nó sẽ không còn đóng vai trò là một baseline nữa.

2. **Tính trung bình phần thưởng trong batch:** Đây là phương pháp đơn giản nhưng vẫn hiệu quả:

- Tính giá trị trung bình của phần thưởng trong một batch dữ liệu
- Trừ giá trị trung bình này khỏi phần thưởng của mỗi mẫu
- Ví dụ: Nếu phần thưởng trung bình trong batch là 0.4, ta sẽ trừ 0.4 từ phần thưởng của mỗi mẫu để tính toán phần thưởng điều chỉnh

- Phương pháp này dễ triển khai và đặc biệt hiệu quả khi làm việc với các batch dữ liệu lớn.

Constrasting Pairwise Examples (e.g. Rafailov et al. 2023)

Một cách tiếp cận đặc biệt trong việc tạo baseline là so sánh các cặp ví dụ hoặc các đầu ra khác nhau cho cùng một đầu vào. Phương pháp này cho phép học trực tiếp từ sự ưu tiên của con người và mang lại sự ổn định cao hơn, vì ta có thể chắc chắn về việc đầu ra nào tốt hơn đầu ra nào.

DPO (Direct Preference Optimization) là một phương pháp gần đây trở nên phổ biến dựa trên ý tưởng này. Cách thức hoạt động của DPO:

$$\ell_{DPO} = \log \sigma \left(\underbrace{\beta \frac{P(Y_w|X;\theta)}{P(Y_w|X;\theta_{\text{old}})}}_{\text{better outputs}} - \underbrace{\beta \frac{P(Y_l|X;\theta)}{P(Y_l|X;\theta_{\text{old}})}}_{\text{worse outputs}} \right)$$

1. **Cơ chế cốt lõi:**

- Tính toán tỷ lệ xác suất giữa mô hình mới và mô hình cũ
- Tăng xác suất cho đầu ra tốt, giảm xác suất cho đầu ra kém

2. **Tham số β trong DPO:**

- Đóng vai trò như một hyperparameter
- Ảnh hưởng đến kích thước bước gradient
- Tác động đến hàm sigmoid trong công thức: β lớn hơn sẽ "kéo giãn" sigmoid. Điều này ảnh hưởng đến cách mô hình phản ứng với những khác biệt nhỏ trong giá trị

3. **So sánh với PPO:**

- DPO có cấu trúc tương tự PPO ở việc sử dụng các tỷ lệ xác suất
- Tuy nhiên, DPO có một hạn chế quan trọng: yêu cầu các đánh giá theo cặp. Không thể huấn luyện mô hình nếu không có các đánh giá theo cặp này

Increasing Batch Size

Trong quá trình huấn luyện học tăng cường, một vấn đề thường gặp là phương sai cao hơn so với Maximum Likelihood Estimation. Điều này chủ yếu do việc lấy mẫu và các yếu tố khác.

Một giải pháp đơn giản để cải thiện sự ổn định là tăng kích thước batch. Cụ thể, bạn có thể tăng số lượng ví dụ hoặc số lần chạy trước khi thực hiện cập nhật để ổn định quá trình học. Nếu bạn gặp vấn đề về độ ổn định sau khi đã thử các kỹ thuật khác, việc tăng kích thước batch có thể giúp giải quyết vấn đề này.

Ngoài ra, một phương pháp khác mà nhiều người thường áp dụng là lưu trữ nhiều lần chạy trước đó. Việc thực hiện các lần chạy và thu thập phần thưởng thường tốn kém, vì vậy việc lưu

trữ các lần chạy đã thực hiện sẽ giúp bạn cập nhật tham số với các lô dữ liệu lớn một cách hiệu quả hơn.

Resources

1. https://phontron.com/class/anlp2024/lectures/#reinforcement-learning-feb-22
2. Recommended Reading: Deep Reinforcement Learning Tutorial (Karpathy 2016)
3. Recommended Reading: Human Feedback Survey (Fernandes et al. 2023)
4. Reference: Course in Machine Learning Chapter 17 (Daume)
5. Reference: Reinforcement Learning Textbook (Sutton and Barto 2016)
6. Reference: TrueSkill (Sakaguchi et al. 2014)
7. Reference: Multi-dimensional Quality Metrics
8. Reference: Large-scale MQM Annotation (Freitag et al. 2021)
9. Reference: BERTScore (Zhang et al. 2019)
10. Reference: COMET (Rei et al. 2020)
11. Reference: GEMBA (Kocmi and Federmann 2023)
12. Reference: AutoMQM (Fernandes et al. 2023)
13. Reference: WMT Metrics Shared Task (Freitag et al. 2023)
14. Reference: SummEval (Fabbri et al. 2020)
15. Reference: Summarization Evaluation through QA (Eyal et al. 2019)
16. Reference: Minimum Risk Training for NMT (Shen et al. 2015)
17. Reference: REINFORCE (Williams 1992)
18. Reference: Co-training (Blum and Mitchell 1998)
19. Reference: Revisiting Self-training (He et al. 2020)
20. Reference: Adding Baselines (Dayan 1990)
21. Reference: Sequence-level Training for RNNs (Ranzato et al. 2016)
22. Reference: PPO (Schulman et al. 2017)
23. Reference: DPO (Rafailov et al. 2023)