Lecture 6: Thuật toán Tạo Sinh

Giới thiệu

Hôm nay, chúng ta sẽ cùng khám phá một số thuật toán sinh (generation algorithms) phổ biến được sử dụng trong Mô hình ngôn ngữ lớn. Bài viết này sẽ cung cấp cái nhìn tổng quan về các phương pháp này cũng như lý thuyết đằng sau chúng!

Probability Distributions

Good news!

Chúng ta bắt đầu với một tin vui: giả sử bạn có một người bạn sở hữu một công ty công nghệ lớn và họ đã tặng bạn một mô hình khổng lồ mang tên M. Đây là một mô hình tuyệt vời, được huấn luyện trước với kiến trúc mới nhất và đã được đào tạo trên hàng triệu token văn bản. Mô hình này có tới bảy tỷ tham số và đứng đầu trong nhiều bảng xếp hạng.

So what's in the box?

Tuy nhiên, nếu bạn mở hộp mô hình M và xem xét kỹ lưỡng, bạn sẽ nhận ra rằng thực chất M chỉ là một phân phối xác suất có điều kiện. Khi bạn đưa một đầu vào X vào mô hình, nó sẽ cho ra xác suất cho bất kỳ chuỗi nào mà bạn muốn đánh giá. Cụ thể, M cung cấp một phân phối xác suất cho tất cả các token trong từ vựng của nó để dự đoán token tiếp theo mà bạn sẽ xuất ra.

$$P(Y|X) = \prod_{j=1}^{J} P\left(y_j|X, y_1, \ldots, y_{j-1}\right)$$

Nếu bạn nhân tất cả các xác suất trong chuỗi lại với nhau, bạn có thể tính xác suất của bất kỳ đầu ra y nào dựa trên đầu vào X. Vậy nên, mô hình đắt tiền mà bạn đã đầu tư thực chất chỉ là một phân phối xác suất có điều kiện.

A model defines a conditional probability distribution

Input *X*	Output *Y*	Task
English text	Japanese	Translation
Question	Answer	Question-answering
Document	Short description	Summarization
Utterance	Response	Response generation
Chess game state	Next chess move	Game-playing
Math problem	Answer	Math reasoning

Việc sử dụng phân phối xác suất điều kiện cho phép chúng ta thực hiện hầu hết các tác vụ mà chúng ta quan tâm. Bằng cách thay đổi cách xác định đầu vào (X) và đầu ra (y), chúng ta có thể áp dụng mô hình này cho nhiều nhiệm vụ khác nhau như dịch thuật, tóm tắt và các nhiệm vụ suy luận, …. Điều này cho thấy tính linh hoạt của mô hình trong việc xử lý và tạo ra các kết quả mong muốn chỉ bằng cách điều chỉnh các cài đặt cho đầu vào và đầu ra.

Probability distributions: confidence

Việc mô hình hóa dữ liệu dưới dạng phân phối xác suất thay vì chỉ cung cấp một câu trả lời duy nhất cho mỗi đầu vào mang lại cả lợi ích và hạn chế. Một trong những điểm tích cực của phân phối xác suất là nó cho phép chúng ta đánh giá mức độ tự tin của mô hình. Ví dụ, khi đưa vào đầu vào "2 plus 2 equals", nếu hầu hết xác suất tập trung vào token "four", chúng ta có thể khẳng định rằng mô hình dự đoán với độ tin cậy cao rằng "2 plus 2 equals four". Ngược lại, nếu đầu vào là một câu hỏi mở như "Graham's favorite color", và phân phối xác suất khá phẳng, với "green" là đầu ra có xác suất cao nhất nhưng không có nhiều tự tin vào câu trả lời đó, điều này cho thấy sự không chắc chắn trong dự đoán.

Khái niệm này liên quan chặt chẽ đến việc hiệu chỉnh mô hình, một chủ đề đã được thảo luận trong các buổi học trước. Tuy nhiên, cần lưu ý rằng trong trường hợp "2 plus 2 equals 4", không phải toàn bộ xác suất đều tập trung vào "four", điều này cho thấy rằng vẫn có những yếu tố không chắc chắn trong dự đoán của mô hình.

Probability distributions: hallucination

Các mô hình xác suất có điều kiện có thể tạo ra những kết quả không chính xác hoặc không mong muốn, được gọi là "hallucination". Dù bạn có làm gì đi chăng nữa, luôn tồn tại một xác suất không bằng 0 cho những đầu ra không đúng hoặc thậm chí có thể gây phản cảm, điều mà bạn không muốn mô hình xuất ra. Đây là một hiện tượng do cách mà các mô hình này được huấn luyện.

Có những nghiên cứu lý thuyết cho thấy rằng ngay cả khi toàn bộ dữ liệu huấn luyện của bạn đều chính xác và không có sai sót, vẫn có khả năng xuất hiện những đầu ra không mong muốn. Điều này xảy ra với hầu hết các đầu vào mà bạn quan tâm. Vậy, nếu chúng ta gặp phải những vấn đề này, làm thế nào để có được đầu ra tốt từ mô hình?

Sampling from LMs

Ancestral Sampling

Khi làm việc với mô hình được xem như một phân phối có điều kiện, tại mỗi bước, chúng ta cần giải mã phân phối xác suất cho tất cả các token trong từ vựng. Mục tiêu của chúng ta là tạo ra một đầu ra tốt, theo một định nghĩa nào đó mà chúng ta sẽ phát triển dần dần.

Một trong những phương pháp đầu tiên để thử nghiệm là lấy mẫu từ phân phối xác suất $y_j \sim P(y_j|X, y_1, ..., y_{j-1})$, được gọi là "ancestral sampling". Tại mỗi bước thời gian, chúng ta sẽ rút một token từ phân phối này theo xác suất tương đối của nó. Điều này có nghĩa là nếu một token có xác suất gấp đôi so với token khác, nó sẽ được chọn thường xuyên gấp đôi.

Việc lấy mẫu từ phân phối này cho phép chúng ta có được các mẫu chính xác từ mô hình. Nếu chúng ta có thể lấy một số lượng mẫu gần như vô hạn, chúng ta sẽ nhận được một tập hợp mẫu phản ánh chính xác xác suất mà mô hình đã cung cấp. Tuy nhiên, mặc dù phương pháp này có vẻ lý tưởng, nhưng thực tế vẫn còn một số vấn đề.

Issues with ancestral sampling: long tail

Mô hình Llama có 32.000 token từ vựng. Trong số đó, có thể chỉ có khoảng 1.000 đến 2.000 token là hợp lý để dự đoán cho một nhiệm vụ mở. Tuy nhiên, trong phân phối này có rất nhiều token khác, như dấu câu, hoặc những token không dẫn đến câu trả lời chính xác, và chúng thường có xác suất rất thấp.

Vấn đề là nếu chúng ta phân bổ một lượng nhỏ xác suất cho 30.000 token khác nhau, tổng xác suất này sẽ nhanh chóng tăng lên. Để minh họa, chúng ta có thể nhìn vào một biểu đồ phân phối dài.

Phần màu xanh lá cây trong biểu đồ, đại diện cho nhiều token có xác suất cao, chiếm 50% tổng xác suất. Trong khi đó, phần màu vàng, bao gồm nhiều token có xác suất thấp, cũng chiếm 50% xác suất còn lại. Điều này có nghĩa là khi thực hiện phương pháp "ancestral sampling", có đến 50% khả năng chúng ta sẽ chọn một token không hợp lý từ phần "long tail".

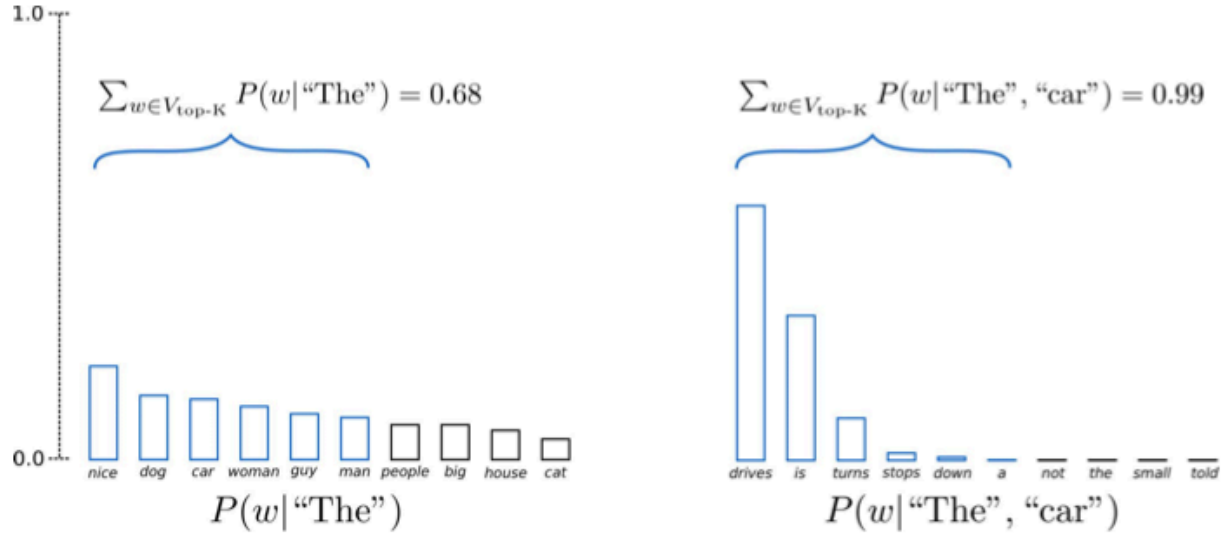
Vậy có giải pháp nào cho vấn đề này không? Một giải pháp rõ ràng là liệu chúng ta có thể cắt bỏ phần đuôi này hay không, tức là nếu biết rằng những token này không có xác suất cao, chúng ta có thể bỏ qua chúng.

What if we just ignore the long tail?

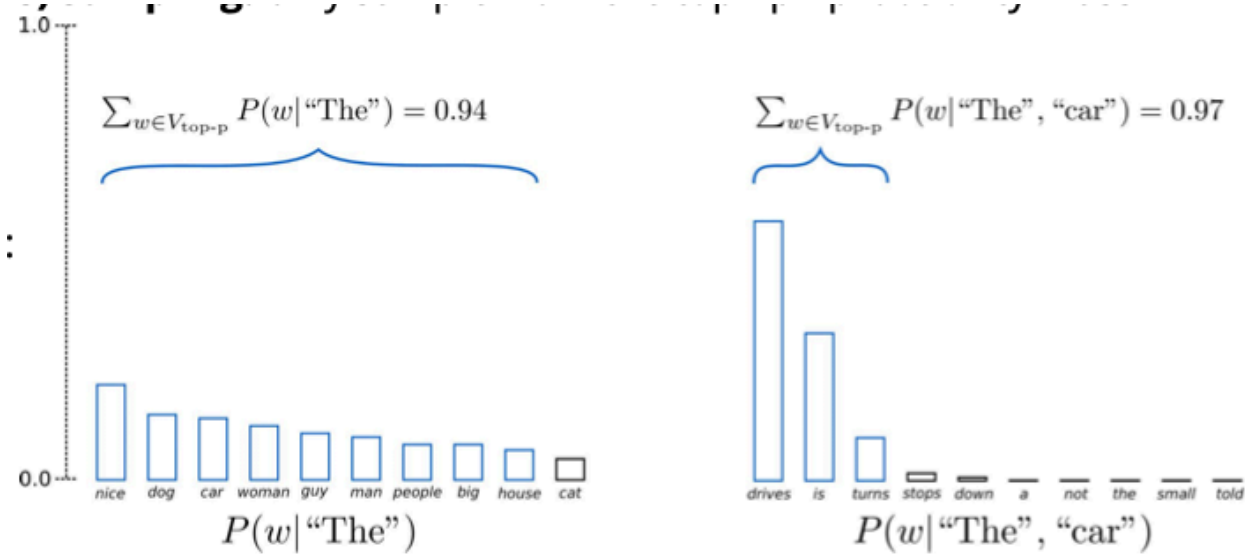
Làm thế nào để chọn ra các token (đơn vị ngôn ngữ) có khả năng cao nhất mà không bị ảnh hưởng bởi những token ít khả năng hơn? Một giải pháp cho vấn đề này là cắt bỏ những token có xác suất thấp, và có một số phương pháp để thực hiện điều này.

Phương pháp đầu tiên là top-k sampling. Trong phương pháp này, chúng ta sẽ chọn ra k token có xác suất cao nhất để lấy mẫu. Ví dụ, nếu chúng ta quyết định chọn top 6 sampling, chúng ta sẽ chỉ lấy mẫu từ 6 token có xác suất cao nhất. Tuy nhiên, nếu phân phối xác suất khá đồng

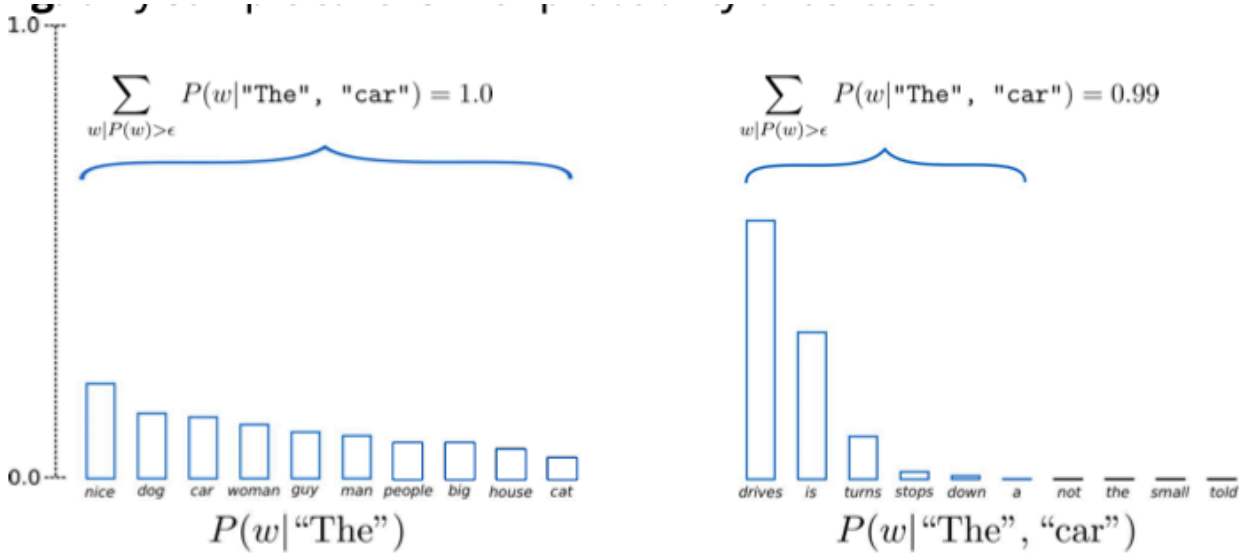
đều, 6 token này chỉ chiếm khoảng 68% tổng xác suất. Ngược lại, nếu phân phối xác suất có xu hướng rõ ràng hơn, 6 token có thể chiếm đến 99% tổng xác suất.



Phương pháp thứ hai là top-p sampling (nucleus sampling). Thay vì chọn một số lượng token cụ thể, phương pháp này cho phép chúng ta chọn một lượng xác suất nhất định từ phân phối. Ví dụ, nếu chúng ta quyết định rằng p là 94%, chúng ta sẽ tiếp tục thêm token vào cho đến khi tổng xác suất đạt khoảng 0.94. Trong trường hợp có nhiều token có xác suất cao, chúng ta có thể chỉ cần một vài token để đạt được xác suất này.



Cuối cùng, phương pháp Epsilon sampling cho phép chúng ta chỉ lấy mẫu những token có xác suất tối thiểu nhất định. Ví dụ, chúng ta có thể chỉ muốn lấy mẫu những token có xác suất ít nhất là 0.05. Trong trường hợp đầu tiên, khi tất cả các token đều có xác suất hợp lý, chúng ta sẽ lấy mẫu từ toàn bộ phân phối. Tuy nhiên, trong trường hợp thứ hai, khi có nhiều token có xác suất rất thấp, chúng ta sẽ chỉ lấy mẫu từ phần có xác suất cao hơn.

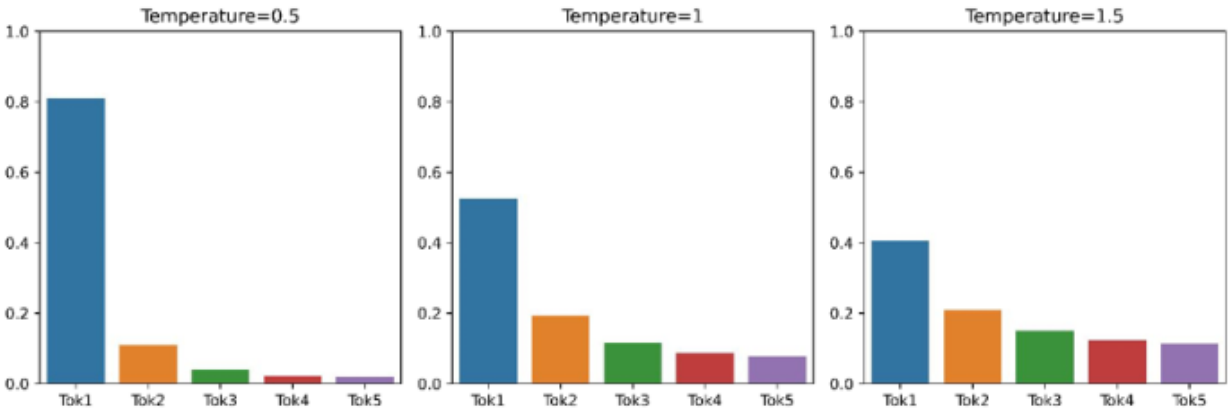


Distribution Temperature

Chúng ta có thể điều chỉnh các đặc điểm của phân phối xác suất cho các token tiếp theo. Một trong những yếu tố quan trọng là độ "nhọn" hay "phẳng" của phân phối.

Khi bạn muốn tạo ra một đầu ra đa dạng và thú vị, chẳng hạn như trong việc viết truyện, bạn có thể muốn phân phối xác suất rộng hơn, tức là làm phẳng phân phối. Điều này có nghĩa là bạn sẽ phân bổ xác suất nhiều hơn cho một tập hợp lớn các token, giúp đầu ra trở nên bất ngờ và phong phú hơn.

Ngược lại, trong các trường hợp như toán học, nơi chỉ có một hoặc một vài câu trả lời hợp lý, bạn có thể muốn phân phối trở nên nhọn hay sắc nét hơn. Điều này có nghĩa là bạn sẽ tập trung xác suất vào những câu trả lời chính xác nhất.



Để thực hiện điều này, bạn sẽ điều chỉnh các giá trị đầu ra của lớp cuối cùng của mô hình trước khi áp dụng hàm softmax. Khi bạn nhận được đầu ra từ lớp cuối cùng, bạn có thể chia các giá trị này cho một số trước khi áp dụng softmax. Nếu bạn sử dụng một số lớn hơn 1, phân phối sẽ

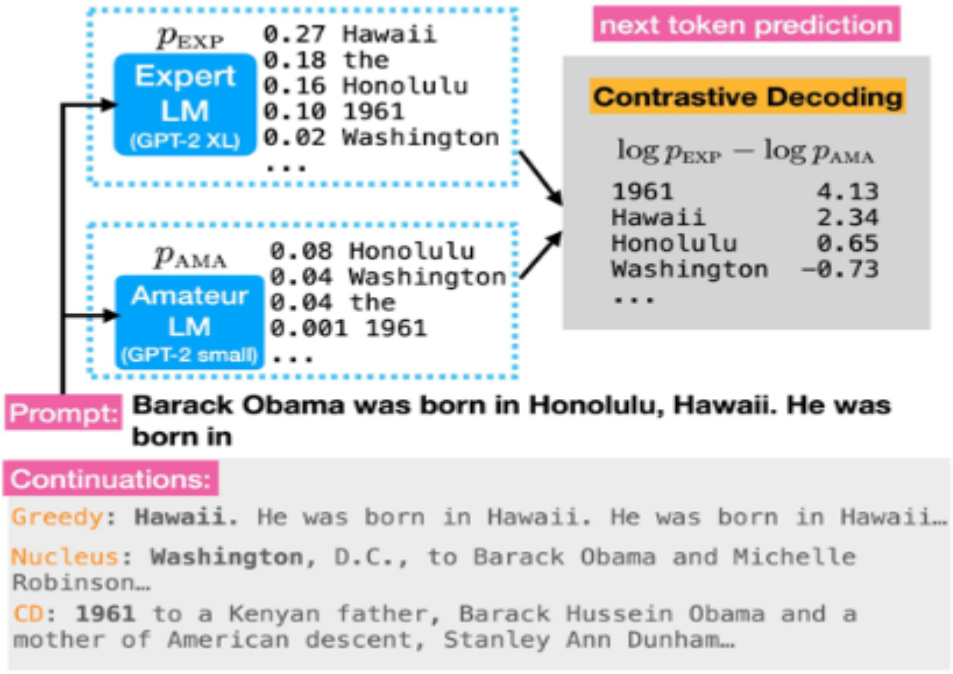
trở nên phẳng hơn; nếu sử dụng số nhỏ hơn 1, phân phối sẽ trở nên sắc nét hơn. Tham số này được gọi là "temperature".

Contrastive Decoding

Trong phần này, chúng ta sẽ khám phá một ý tưởng mới được gọi là "contrastive decoding". Ý tưởng chính là chúng ta có thể kết hợp thông tin bổ sung trong quá trình giải mã bằng cách sử dụng một mô hình khác hoặc một phân phối khác.

Khi làm việc với các mô hình ngôn ngữ nhỏ, chẳng hạn như "gpt2 small", bạn có thể nhận thấy rằng chúng thường lặp lại các chuỗi đầu ra hoặc đưa ra các câu trả lời sai lệch cho các câu hỏi thực tế. Ngược lại, các mô hình lớn hơn, được đào tạo trên nhiều dữ liệu hơn, thường không gặp phải những vấn đề này. Vậy liệu chúng ta có thể sử dụng những sai sót của mô hình nhỏ để cải thiện mô hình lớn hơn không?

Cách thực hiện là điều chỉnh phân phối xác suất của mô hình lớn để chọn những đầu ra mà mô hình lớn cho là có khả năng cao, nhưng mô hình nhỏ lại cho là không khả thi. Ví dụ, nếu đầu vào là "Barack Obama was born in Hawaii he was born in L", mô hình nhỏ có thể bắt đầu lặp lại câu này, trong khi mô hình lớn có thể cung cấp thông tin chính xác hơn, như năm sinh của Barack Obama.



Phương pháp "contrastive decoding" cho phép chúng ta trừ đi xác suất từ mô hình yếu để giữ lại những đầu ra tốt mà chỉ mô hình mạnh mới biết. Điều này giúp loại bỏ những hành vi lặp lại không mong muốn và cải thiện chất lượng đầu ra.

Khi áp dụng phương pháp này cho các chuỗi dài, chúng ta sẽ sử dụng mô hình chuyên gia để giải mã và mô hình yếu để điều chỉnh xác suất cho từng bước dự đoán token tiếp theo. Mặc dù có thể có sự khác biệt nhỏ giữa hai mô hình, nhưng điều quan trọng là mô hình yếu không nên quá gần với mô hình mạnh để không loại bỏ những thông tin hữu ích.

Cuối cùng, phương pháp này không yêu cầu phải huấn luyện lại mô hình, mà chỉ cần sử dụng các mô hình đã được đào tạo sẵn để cải thiện quá trình suy diễn.

Mode-seeking search

Mode-seeking decoding methods

Có thể chúng ta không chỉ muốn lấy một mẫu từ phân phối của mô hình hoặc từ một phân phối đã được điều chỉnh, mà thực sự chúng ta chỉ muốn tìm ra "điều tốt nhất", tức là đầu ra có xác suất cao nhất dựa trên đầu vào mà chúng ta đã cung cấp. Đầu ra này được ký hiệu là $\widehat{Y}$, là chuỗi duy nhất thỏa mãn điều kiện có điểm số cao nhất $P(Y|X)$ cho đầu vào X mà chúng ta đã đưa cho mô hình.

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

Phương pháp này được gọi là "mode seeking search", vì nó tìm kiếm mode của phân phối đầu ra. Nếu bạn lấy mẫu một số lượng lớn từ phân phối này và xem chuỗi có xác suất cao nhất mà bạn nhận được, đó chính là $\widehat{Y}$. Vậy làm thế nào để chúng ta tìm ra được $\widehat{Y}$ này?

Greedy decoding

Trong quá trình tìm kiếm đầu ra cho mô hình ngôn ngữ, một trong những phương pháp phổ biến là Greedy decoding. Ý tưởng cơ bản là tại mỗi bước, chúng ta sẽ chọn từ có xác suất cao nhất từ phân phối xác suất và tiếp tục sinh ra các từ cho đến khi gặp token dừng, hay còn gọi là end of sequence token.

$$y_j = argmax\, P\left(y_j|X,\, y_1,\, \ldots,\, y_{j-1}\right)$$

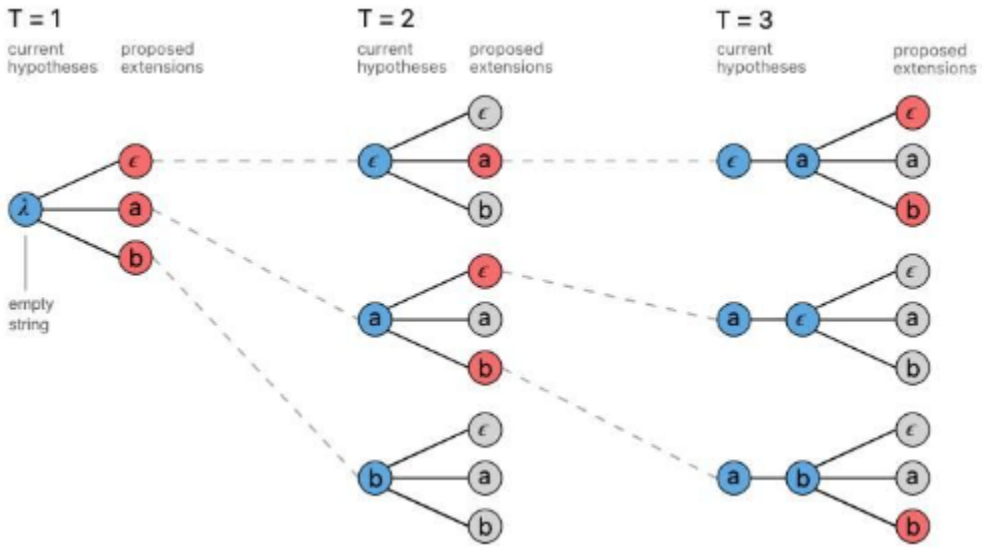
Đối với một đầu ra chỉ gồm một token, phương pháp này hoạt động rất tốt, vì nó cho ra đầu ra có xác suất cao nhất. Tuy nhiên, nếu chúng ta cần một đầu ra dài hơn, có thể gồm nhiều token, thì Greedy decoding có thể gặp vấn đề. Cụ thể, chuỗi có xác suất cao nhất có thể không bắt đầu bằng token có xác suất cao nhất ở bước đầu tiên. Ví dụ, trong trường hợp sinh ngẫu nhiên không có điều kiện, token có xác suất cao nhất ở bước đầu tiên có thể là "the", nhưng có thể có một câu rất có xác suất cao mà không bắt đầu bằng từ "the". Do đó, Greedy decoding không đảm bảo tìm ra đầu ra có xác suất cao nhất cho một chuỗi dài hơn một token.

Vậy có cách nào tốt hơn để tìm ra đầu ra này không? Một trong những phương pháp phổ biến đó là beam search.

Beam Search

Ý tưởng chính của Beam Search là chúng ta không muốn bỏ lỡ một token có xác suất cao, có thể bị ẩn sau một tiền tố có xác suất thấp. Do đó, chúng ta cần khám phá nhiều lựa chọn khác nhau để không loại bỏ quá sớm những khả năng có xác suất cao trong quá trình sinh.

Beam search hoạt động như một loại tìm kiếm theo chiều rộng. Tại mỗi bước thời gian, chúng ta sẽ xem xét một loạt các lựa chọn và chọn một tập hợp để tiếp tục. Thay vì chỉ tạo ra một chuỗi duy nhất và sau đó lại tạo ra một chuỗi khác, chúng sẽ chọn một số ứng viên để khám phá, ví dụ như chọn 3 tuỳ chọn cho bước thời gian đầu tiên. Từ mỗi lựa chọn này, chúng ta sẽ xác định ba khả năng có xác suất cao nhất cho bước thời gian tiếp theo.



Quá trình này sẽ tạo ra nhiều lựa chọn, nhưng để tránh sự bùng nổ tổ hợp, chúng ta cần chọn một tập con để tiếp tục. Chúng ta sẽ xem xét xác suất của từng chuỗi hai token và chọn ba chuỗi có xác suất cao nhất. Cuối cùng, chúng ta sẽ tiếp tục quá trình này cho đến khi hoàn thành, và chỉ chọn chuỗi có xác suất cao nhất từ ba chuỗi cuối cùng.

Mặc dù beam search không đảm bảo sẽ tìm ra lựa chọn có xác suất cao nhất, nhưng nó thường hoạt động tốt hơn so với greedy decoding. Trong các mô hình ngôn ngữ hiện nay, phương pháp giải mã phổ biến thường là beam search hoặc temperature sampling.

Tuy nhiên, beam search cũng có một số vấn đề. Một trong những vấn đề lớn nhất là khi thực hiện sampling tối đa, chúng ta có thể hy sinh sự đa dạng trong các đầu ra. Kết quả là, cuối cùng bạn có thể nhận được ba đầu ra khác nhau nhưng lại rất giống nhau, chỉ khác nhau ở một vài token.

Improving diversity: diverse and stochastics beam search

Có hai phương pháp để tạo ra các tập hợp đầu ra đa dạng hơn trong quá trình sinh văn bản đó là diverse beam search và stochastic beam search.

Diverse beam search nhằm mục đích cải thiện bước đánh giá khi chọn lựa các beam tiếp theo. Thay vì chỉ chọn những beam có xác suất cao nhất, phương pháp này còn xem xét sự tương đồng giữa các beam. Ví dụ, nếu beam có xác suất cao nhất là một chuỗi "A", và có một chuỗi khác cũng có xác suất cao nhưng rất giống với "A", thì phương pháp này sẽ ưu tiên chọn một chuỗi có xác suất thấp hơn nhưng khác biệt hơn để tăng cường sự đa dạng trong tập hợp đầu ra.

Ngược lại, stochastic beam search giữ nguyên cách đánh giá nhưng thay vì chỉ chọn ba token có xác suất cao nhất, nó sẽ lấy mẫu từ một phân phối xác suất. Điều này cho phép khám phá nhiều khả năng hơn trong không gian đầu ra của mô hình, giúp tránh việc tạo ra các đầu ra quá giống nhau.

Cả hai phương pháp này đều có thể được áp dụng cho các tác vụ sinh văn bản mở, nhưng hiệu quả có thể khác nhau tùy thuộc vào chất lượng của mô hình. Để đo lường sự tương đồng giữa các beam, có thể sử dụng các hàm tương đồng khác nhau, chẳng hạn như so sánh số lượng token giống nhau giữa các beam.

Một điểm quan trọng trong stochastic beam search là việc lấy mẫu không thay thế, nghĩa là các token được chọn sẽ luôn khác nhau, đảm bảo rằng mỗi beam đều có sự độc đáo nhất định. Điều này khác với các phương pháp lấy mẫu thông thường, nơi có thể chọn cùng một token nhiều lần.

Minimum Bayes Risk

Wait: is the highest-probability output best?

Liệu việc tìm kiếm chuỗi có xác suất cao nhất từ mô hình có thực sự là điều chúng ta muốn hay không? Chúng ta biết rằng các đầu ra có xác suất rất thấp thường kém hơn so với các đầu ra có xác suất cao. Ví dụ, khi dự đoán câu tiếp theo sau "the cat saw the dog", câu "the cat sat down" có xác suất cao hơn nhiều so với "the cat grew wings". Rõ ràng, câu đầu tiên là một đầu ra hợp lý hơn.

Tuy nhiên, khi chỉ xem xét các đầu ra có xác suất tương đối cao, việc xác định thứ hạng giữa chúng trở nên khó khăn hơn. Chẳng hạn, câu "the cat sat down" có thể có xác suất cao hơn một chút so với "the cat ran away", nhưng cả hai đều là những đầu ra hợp lý.

Nếu chúng ta xem xét một loạt các đầu ra, giả sử có sáu đầu ra mà mô hình có thể tạo ra, theo thứ tự xác suất là: "the cat sat down", "it ran away", "it sprinted off", "it got out of there", "it's very small", và "it grew wings". Chúng ta có thể chắc chắn rằng "the cat sat down" là đầu ra tốt hơn so với "it grew wings". Tuy nhiên, nếu nhìn vào các đầu ra khác, chúng ta thấy rằng có một nhóm đầu ra khác cũng diễn đạt ý nghĩa "the cat left the area". Tổng xác suất của ba đầu ra này thậm chí còn gấp đôi xác suất của "the cat sat down". Nếu chỉ thực hiện tìm kiếm theo chế độ xác suất cao nhất, chúng ta sẽ không nhận ra được điều này. Vậy, liệu chúng ta có nên trả về "the cat sat down" hay nên chọn một đầu ra khác có nghĩa là "the cat left the area"?

Then what makes a good output?

Trong quá trình đánh giá đầu ra của một mô hình, câu hỏi đặt ra là: nếu không phải xác suất là yếu tố quyết định chất lượng đầu ra, vậy điều gì sẽ là yếu tố đó? Chúng ta có thể có một đầu ra có xác suất rất cao nhưng lại khác biệt so với các đầu ra khác trong tập dữ liệu, trong khi đó, có những đầu ra khác cũng có xác suất cao nhưng lại tương đồng với nhiều đầu ra khác. Có thể, việc chọn một đầu ra có xác suất cao nhưng ít rủi ro hơn sẽ là lựa chọn an toàn hơn.

Một ví dụ minh họa cho điều này là: giả sử bạn và bạn bè đang gian lận trong một bài kiểm tra (mặc dù điều này không nên làm). Nếu tất cả bạn bè của bạn gửi cho bạn câu trả lời, có thể một người bạn có điểm số cao hơn một chút so với những người khác nhưng lại cho rằng câu trả lời là "answer A", trong khi mọi người khác đều chọn "answer B". Trong trường hợp này, bạn có thể vẫn chọn câu trả lời mà mọi người khác đã chọn, vì nó có vẻ ít rủi ro hơn.

Vậy làm thế nào để tìm ra một đầu ra có xác suất cao và rủi ro thấp? Có hai câu hỏi cần giải quyết: đầu tiên là làm thế nào để ước lượng xác suất, và thứ hai là làm thế nào để ước lượng rủi ro. Để ước lượng xác suất, chúng ta có thể lấy mẫu từ mô hình và đếm tần suất xuất hiện của các đầu ra. Phương pháp này được gọi là Monte Carlo sampling. Nếu thực hiện đủ lần, nó sẽ cho chúng ta phân phối chính xác của mô hình.

Để ước lượng rủi ro, chúng ta có thể xem xét các đầu ra khác trong tập dữ liệu như là các tham chiếu giả. Chúng ta sẽ đánh giá sự đồng thuận giữa đầu ra đang xem xét và các tham chiếu này bằng cách sử dụng một số chỉ số đo lường độ tương đồng, chẳng hạn như các chỉ số n-gram như Rouge hoặc BLEU, hoặc các chỉ số ngữ nghĩa như BT score hoặc Bart score.

Minimum Bayes Risk (MBR)

Để tìm ra một đầu ra có xác suất cao và rủi ro thấp, chúng ta sẽ xem xét khái niệm "minimum based risk decoding" (MBR). MBR là một phương pháp giải mã nhằm chọn lựa các đầu ra có rủi ro thấp và xác suất cao. Điều này có nghĩa là chúng ta sẽ chọn những đầu ra tương tự với nhiều đầu ra khác trong tập hợp đã được lấy mẫu, và những đầu ra có xác suất cao sẽ có ảnh hưởng lớn hơn đến quyết định cuối cùng.

$$\hat{y} = \underset{y' \in \mathcal{Y}_h}{\operatorname{argmax}} \sum_{y \in \mathcal{Y}_e} G(y, y')$$

Để minh họa, chúng ta có thể xem xét ví dụ trong tóm tắt văn bản, nơi chúng ta sử dụng một chỉ số như "Rouge" - một chỉ số đo lường sự trùng lặp n-gram. Khi chúng ta lấy mẫu 100 đầu ra và áp dụng MBR, kết quả cho thấy phương pháp này vượt trội hơn so với các phương pháp khác như greedy decoding, beam search và diverse beam search. Mặc dù MBR cho kết quả tốt hơn, nó yêu cầu nhiều tài nguyên hơn do cần phải lấy mẫu nhiều đầu ra.

Một câu hỏi thú vị được đặt ra là tại sao beam search với nhiều beam lại có hiệu suất kém hơn. Hiện tượng này được gọi là "cursive beam search", trong đó việc tìm kiếm đầu ra có xác suất

cao nhất không nhất thiết mang lại kết quả tốt nhất cho các chỉ số đánh giá sau này. Điều này cho thấy rằng việc tối đa hóa xác suất không đồng nghĩa với việc tối ưu hóa cho các mục tiêu cụ thể trong ứng dụng.

Ngoài ra, cách mà các mô hình được huấn luyện cũng ảnh hưởng đến kết quả. Hầu hết các mô hình hiện nay được huấn luyện bằng phương pháp "maximum likelihood", điều này có thể dẫn đến việc chúng trả lời những câu ngắn gọn hoặc thậm chí là không trả lời, vì những câu này có xác suất cao hơn so với việc đưa ra một câu trả lời cụ thể.

Cuối cùng, chúng ta cũng cần lưu ý rằng việc đánh giá độ tin cậy của các đầu ra trong các chuỗi dài là một thách thức lớn, vì độ tin cậy thường chỉ phản ánh xác suất của một chuỗi cụ thể, không phải là không gian ngữ nghĩa tổng thể mà chúng ta thực sự quan tâm.

MBR variants: output ensembling

Một biến thể của MBR là "output ensembling", tức là khi bạn có nhiều mô hình khác nhau và nhận được đầu ra từ tất cả chúng, bạn cần chọn ra đầu ra tốt nhất trong tập hợp đó. Một cách thông dụng để thực hiện điều này là so sánh độ tương đồng embedding giữa các mô hình. Ví dụ, bạn có thể hỏi: "Does model one think these two things are really similar?" và cố gắng chọn một đầu ra có độ tương đồng cao với nhiều đầu ra khác.

Gần đây, chúng ta đã thảo luận về MBR (Minimum Bayes Risk), và bạn có thể nhận thấy rằng phương pháp này có cùng một công thức tổng quát. Tuy nhiên, thay vì tổng hợp các đầu ra từ một mô hình duy nhất, giờ đây bạn đang xem xét các đầu ra từ một tập hợp các mô hình.

MBR variants: self-consistency (Wang et al., 2023)

Một trong các biến thể MBR gần đây đó là self-consistency. Ý tưởng ở đây là bạn muốn thực hiện một dạng lý luận toán học và rất quan tâm đến việc có được câu trả lời cuối cùng chính xác, nhưng không nhất thiết phải đảm bảo rằng tất cả các bước lý luận giữa các bước đó đều đúng.

Bạn sẽ prompt cho mô hình đưa ra câu trả lời bằng cách sử dụng phương pháp Chain of Thought, tức là yêu cầu nó trình bày các bước mà nó sẽ thực hiện trước khi đưa ra câu trả lời cuối cùng. Sau đó, bạn sẽ lấy mẫu nhiều lần và hoàn toàn bỏ qua các chuỗi lý luận, chỉ giữ lại câu trả lời từ mỗi lần sinh. Bạn có thể có khoảng 20, 30 hoặc 100 câu trả lời và chỉ cần trả về câu trả lời được tạo ra nhiều nhất.

Phương pháp này thực hiện một dạng MBR, trong đó tiêu chí mà bạn thực sự quan tâm là sự khớp chính xác của câu trả lời, bỏ qua phần còn lại của quá trình sinh. Mục tiêu là có được một đầu ra có xác suất cao, tức là được tạo ra nhiều lần, nhưng cũng có rủi ro thấp, nghĩa là không có nhiều đầu ra khác trong tập hợp của bạn mâu thuẫn với câu trả lời này.

Constrained Generation

Imposing global constraints

Trong phần này, chúng ta sẽ xem xét ba phương pháp suy diễn chính bao gồm: lấy mẫu từ một phân phối nào đó, tìm kiếm trong một không gian phân phối và thực hiện một số phân tích trên một tập hợp mẫu để chọn ra những mẫu mà chúng ta muốn trả về.

Một câu hỏi được đặt ra là tại sao việc trung bình hóa trọng số của các mô hình không được coi là MBR. Câu trả lời là vì một trong những yếu tố quan trọng làm cho một phương pháp trở thành MBR là khái niệm so sánh giữa nhiều "pseudo references" (tham chiếu giả). Khi bạn trung bình hóa trọng số của các mô hình, bạn có thể kết thúc với một đầu ra duy nhất, có thể sử dụng thông tin từ hai phân phối mô hình mà bạn đã kết hợp lại, nhưng điều này không giống như việc so sánh với các tham chiếu giả hoặc xếp hạng trong một tập hợp.

Mục tiêu của chúng ta là tìm ra một đầu ra tốt từ mô hình. Tuy nhiên, chúng ta cũng muốn thêm một số ràng buộc bổ sung. Ví dụ, nếu tôi yêu cầu mô hình gợi ý một số sở thích mà tôi có thể làm để giữ dáng, tôi có thể nói rằng tôi không muốn mô hình gợi ý việc leo núi. Tôi đã thử và không thích nó. Vậy làm thế nào để tôi có thể yêu cầu mô hình ngừng gợi ý việc leo núi cho tôi? Nếu bạn đã thử nghiệm với một số mô hình ngôn ngữ lớn gần đây, bạn có thể nghĩ rằng điều này khá dễ dàng. Bạn chỉ cần nói với mô hình rằng bạn không muốn nói về việc leo núi.

Putting instructions in the input isn’t enough

Gần đây, tôi đã trò chuyện với Bard và nhận thấy rằng việc điều khiển mô hình không hề đơn giản. Khi tôi yêu cầu mô hình không nói về việc leo núi, nó có thể tuân theo một thời gian, nhưng sau đó lại gợi ý: "maybe you want to try rap climbing". Điều này cho thấy rằng, mặc dù chúng ta có thể cố gắng hướng dẫn mô hình, nhưng có thể có cách khác để áp đặt ràng buộc này trong quá trình giải mã. Vì vậy, tôi quyết định sẽ thực hiện một điều gì đó táo bạo.

Constrained decoding: logit manipulation

Tôi có thể thao túng phân phối xác suất cho đầu ra của mô hình. Ví dụ, nếu tôi không muốn thấy từ "climbing", tôi có thể thiết lập xác suất của nó bằng 0. Điều này có thể thực hiện dễ dàng bằng cách thêm một số âm lớn vào giá trị logic của từ "climbing" trước khi áp dụng hàm softmax, từ đó xác suất của nó sẽ gần như bằng 0 và chúng ta sẽ không thấy nó trong kết quả.

Tuy nhiên, giải pháp này không hoàn hảo. Nếu mô hình gợi ý cho tôi về "bouldering", liệu tôi có cần phải liệt kê tất cả các từ đồng nghĩa với "climbing" hay không? Hơn nữa, có những cách sử dụng hợp lệ cho từ "climbing". Chẳng hạn, tôi muốn mô hình gợi ý "hiking" vì "climbing up a mountain to see a good view is relaxing", nhưng chúng ta đã quyết định không sử dụng từ "climbing".

Ngoài ra, mô hình có thể gợi ý các thuật ngữ liên quan trước khi đến từ bị hạn chế. Ví dụ, nó có thể đề xuất "maybe you can work out by going to an indoor rock blank", và lúc này chúng ta lại

không thể nói "rock climbing". Thậm chí, mô hình có thể gợi ý rằng "rock collecting is a hobby to stay in shape", nhưng điều này cũng không hợp lý. Do đó, chúng ta có thể phải tiếp tục xây dựng các quy tắc phức tạp hơn để xử lý những tình huống này.

Constrained decoding: sample-then-rank (or reject)

Một cách tiếp cận đơn giản hơn để đảm bảo đầu ra mô hình không liên quan đến "climbing" là lấy mẫu nhiều đầu ra từ mô hình và kiểm tra xem chúng có liên quan đến chủ đề này hay không. Nếu có, chúng ta sẽ loại bỏ chúng.

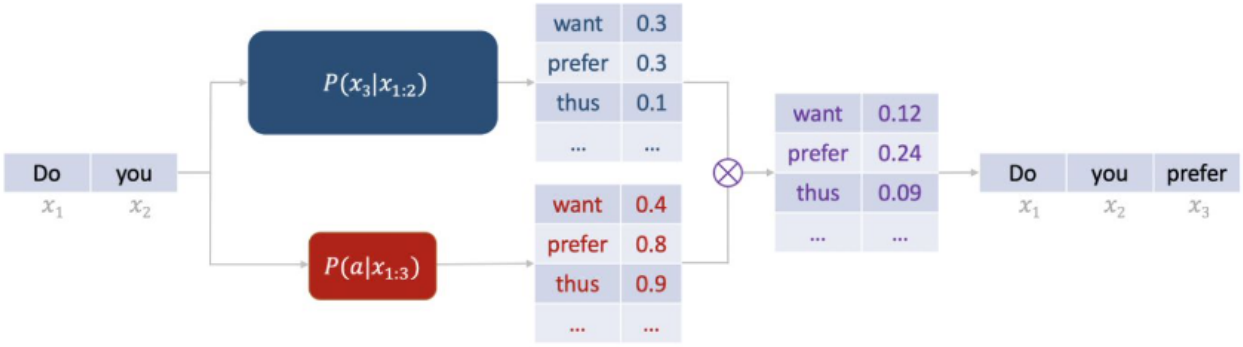
```
for si in S:
    if si.is_about_climbing():
        discard(si)
```

Một lợi thế của phương pháp này là dễ dàng kiểm tra sau khi tạo ra đầu ra để xác định xem nó có thỏa mãn yêu cầu hay không. Chúng ta có thể huấn luyện một mô hình nhỏ hơn để dự đoán xem chủ đề của câu có phải là "climbing" hay không, hoặc thậm chí nhờ một người bạn xem qua nội dung và loại bỏ những gì không liên quan.

Tuy nhiên, nếu mô hình gán xác suất cao cho những đầu ra liên quan đến "climbing", chúng ta có thể phải tạo ra hàng trăm hoặc hàng nghìn chuỗi để tìm ra một cái không liên quan, điều này có vẻ không hiệu quả. Do đó, một giải pháp tốt hơn là cố gắng dự đoán trong quá trình tạo ra liệu chuỗi có khả năng liên quan đến "climbing" hay không, từ đó điều chỉnh hoặc thậm chí khởi động lại quá trình tạo ra để tránh kết quả không mong muốn ngay từ đầu.

Constrained Decoding: FUDGE (Yang & Klein, 2021)

Một phương pháp hữu ích trong trường hợp này là "fudge". Phương pháp này nhằm tạo ra các câu có tính trang trọng hơn trong quá trình sinh văn bản. Cụ thể, tại mỗi bước dự đoán, mô hình sẽ đưa ra các dự đoán cho từ tiếp theo, tạo thành một phân phối xác suất. Đồng thời, chúng ta cũng có một phân phối thứ hai để đánh giá khả năng câu sẽ trở nên trang trọng khi hoàn thành.



Ví dụ, một câu bắt đầu bằng "Do you want" có thể có xác suất thấp hơn để trở thành một câu trang trọng so với câu bắt đầu bằng "Do you prefer". Chúng ta sẽ kết hợp hai phân phối này bằng cách nhân chúng với nhau, sau đó lấy mẫu từ phân phối đã được điều chỉnh. Kết quả là, chúng ta có thể tạo ra các câu có khả năng cao hơn để đạt được tính trang trọng mà không cần phải thử nghiệm nhiều câu và loại bỏ những câu không phù hợp.

Tên "fudge" xuất phát từ việc sử dụng một mô hình gọi là "future discriminator" (người phân biệt tương lai). Mô hình này được huấn luyện trên các tiền tố của câu để dự đoán xem câu đó có khả năng trở nên trang trọng hay không. Nếu có một tập dữ liệu được phân loại rõ ràng thành câu trang trọng và không trang trọng, chúng ta có thể tạo ra các ví dụ huấn luyện từ các tiền tố của những câu này.

Constrained decoding via… RLHF?

Một trong những ràng buộc mà chúng ta quan tâm có thể là phù hợp với sở thích của con người. Cách mà chúng ta thường thực hiện ràng buộc này là thông qua "reinforcement learning through human feedback".

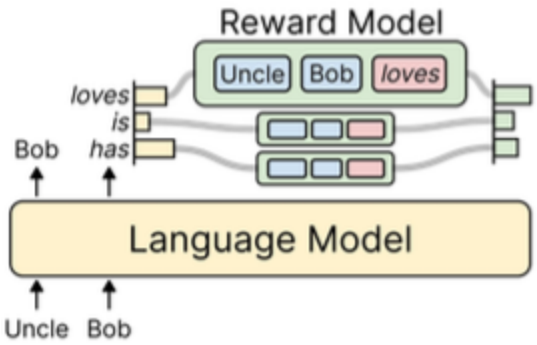
Chúng ta bắt đầu với phân phối mô hình ban đầu và lấy một phân phối rất chặt chẽ của các bằng chứng, cho thấy rằng mô hình này đánh giá một chuỗi nào đó là có phần thưởng cao, trong khi một chuỗi khác lại có phần thưởng thấp. Mục tiêu của chúng ta là kết hợp chúng thông qua quá trình huấn luyện để có được một mô hình mới, mà chúng ta gọi là "aligned" (được căn chỉnh), có khả năng cao hơn trong việc cung cấp những kết quả có phần thưởng cao theo phân phối phần thưởng của chúng ta.

Có thể xem quá trình này như một loại "Bayesian inference" (suy diễn Bayes). Điều này có nghĩa là phân phối mà chúng ta thực sự muốn đạt được cuối cùng là một phân phối kết hợp giữa phân phối của mô hình ban đầu và một ý tưởng về khả năng chúng ta thỏa mãn phần thưởng. Chúng ta thực hiện điều này thông qua học tăng cường, nhưng nếu chúng ta biết rõ hai phân phối này trông như thế nào, thì có vẻ như chúng ta có thể chỉ cần thêm thông tin bên ngoài vào thời điểm giải mã để đạt được một số hiệu ứng tương tự.

Reward-augmented decoding

Một ví dụ điển hình là phương pháp "Reward Augmented Decoding" được giới thiệu trong một bài báo năm ngoái. Phương pháp này có cùng ý tưởng với kỹ thuật FUDGE, nhưng thay vì dự đoán khả năng thỏa mãn ràng buộc, nó dự đoán mức độ phần thưởng (reward) mà chuỗi có thể nhận được khi kết thúc quá trình sinh.

Cụ thể, quá trình này diễn ra như sau:
1. Sử dụng mô hình gốc (chưa qua RLHF) để dự đoán token tiếp theo.
2. Dùng một mô hình được huấn luyện để dự đoán phần thưởng có thể nhận được, dựa trên prefix hiện tại (tương tự như một "future discriminator").
3. Tính toán phần thưởng có thể nhận được nếu chọn mỗi token.
4. Kết hợp hai phân phối này để quyết định token nào sẽ được chọn tiếp theo trong quá trình giải mã.



Phương pháp này mang lại một số lợi ích của RLHF mà không cần thực hiện quá trình học củng cố phức tạp. Nó cho phép xem việc điều chỉnh mô hình theo phản hồi của con người như một ràng buộc có thể áp dụng trong quá trình giải mã.

Tuy nhiên, phương pháp này cũng có một số hạn chế:

- Cần phải huấn luyện một mô hình phân biệt riêng cho mỗi ràng buộc mới.
- Đòi hỏi một tập dữ liệu bao gồm các mẫu thỏa mãn và không thỏa mãn ràng buộc để huấn luyện mô hình phân biệt.

Một cách tiếp cận thay thế là huấn luyện một mô hình phân biệt duy nhất để xác định xem có bất kỳ ràng buộc nào bị vi phạm hay không. Điều này có thể giảm bớt gánh nặng huấn luyện khi có nhiều ràng buộc.

Về mặt hiệu suất, việc dự đoán khả năng vi phạm ràng buộc thường là một tác vụ nhẹ hơn so với việc dự đoán token tiếp theo. Ta có thể sử dụng một mô hình nhỏ hơn cho tác vụ này, và thậm chí có thể tận dụng cùng một biểu diễn (representation) để dự đoán với một bộ phân loại nhị phân.

Các ứng dụng thực tế của kỹ thuật này bao gồm:

- Tăng cường các thuộc tính mong muốn trong văn bản được tạo ra, như sự đồng cảm.
- Ngăn chặn mô hình tạo ra nội dung không phù hợp hoặc xúc phạm.
- Áp dụng các ràng buộc phức tạp, như "không đề cập đến một tập hợp các chủ đề cụ thể".

Để thu thập dữ liệu huấn luyện cho mô hình phân biệt, có thể:

- Sử dụng các bộ dữ liệu có sẵn đã ghi nhận thuộc tính cần quan tâm.
- Viết các quy tắc heuristic để tạo ra các mẫu dữ liệu.
- Tận dụng dữ liệu từ các diễn đàn hoặc nguồn chuyên biệt.

Cuối cùng, việc lựa chọn giữa fine-tuning toàn bộ mô hình hay sử dụng các kỹ thuật giải mã có ràng buộc phụ thuộc vào cân nhắc giữa chi phí huấn luyện một lần và chi phí tính toán trong quá trình suy luận.

Human-in-the-loop decoding

Trong phần này, chúng ta sẽ thảo luận về các phương pháp kết hợp tương tác của con người vào quá trình giải mã. Những gì chúng ta đã xem xét cho đến nay đều mang tính chất "hộp đen", tức là bạn cung cấp cho mô hình M một đầu vào, có thể thực hiện một số thao tác trên phía giải mã và nhận lại một đầu ra. Tuy nhiên, trong nhiều tình huống, đặc biệt là các ứng dụng có rủi ro cao, cần có sự giám sát và can thiệp liên tục từ con người, hoặc trong các tình huống mà bạn muốn hợp tác giữa con người và AI, việc tương tác với mô hình sẽ diễn ra qua một loạt các lần giải mã có sự can thiệp của con người.

Human-in-the-loop decoding: interleaved text

Chúng ta sẽ xem xét một số chiến lược này, và nếu bạn đã sử dụng các mô hình ngôn ngữ hiện đại, bạn có thể đã quen thuộc với một vài trong số chúng. Một trong những cách đơn giản nhất để bắt đầu là "interleaving text" (đan xen văn bản). Bạn có thể chọn thời điểm mà mô hình bắt đầu và dừng giải mã, cũng như thời điểm mà con người viết văn bản xen kẽ. Điều này cho phép bạn điều kiện hóa mô hình dựa trên sự kết hợp giữa văn bản do con người và mô hình tạo ra.

Ngoài ra, bạn cũng có thể yêu cầu mô hình tạo ra một tập hợp văn bản và chỉnh sửa nó theo cách nào đó. Ví dụ, con người có thể áp đặt một ràng buộc tinh tế như "I want it to sound like my writing style". Mặc dù chúng ta không có một bộ phân loại cho điều này, nhưng con người có thể sửa đổi văn bản và tiếp tục tạo ra từ điểm đó, ảnh hưởng đến phong cách của văn bản tiếp theo.

Một ví dụ cụ thể là khi bạn đang viết một câu chuyện cùng nhau, bạn sẽ quay đi quay lại và chỉnh sửa văn bản. Tuy nhiên, bạn cũng có thể nghĩ về bất kỳ cuộc trò chuyện nào với mô hình như một dạng "interleaving of text", nơi mô hình cung cấp một số văn bản, bạn cung cấp một số văn bản khác và cả hai cùng tương tác để điều kiện hóa mô hình.

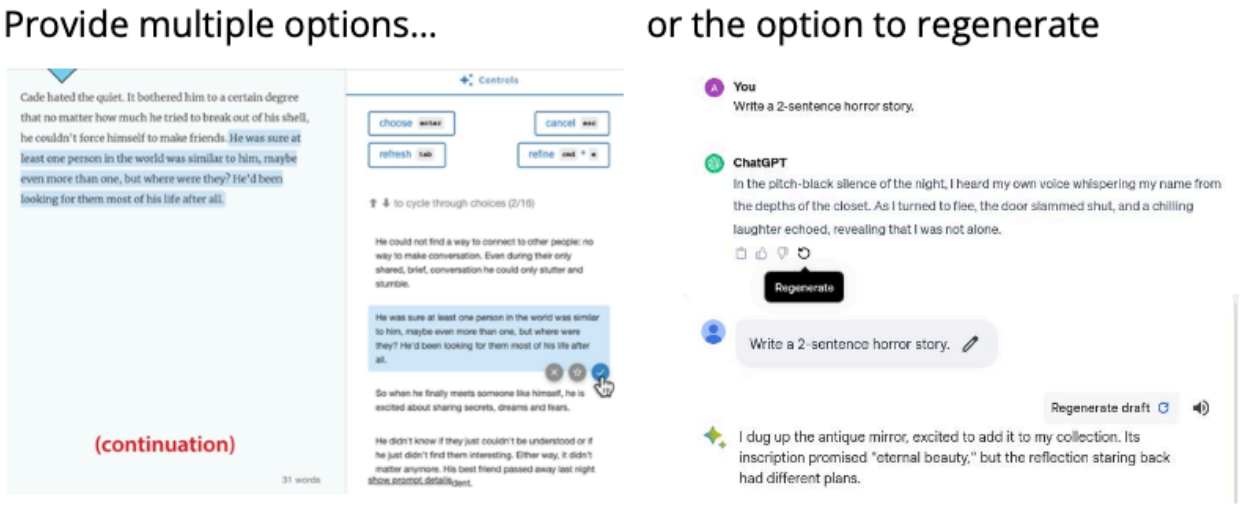
Human-in-the-loop decoding: fined-grained replacement

Trong quá trình tạo văn bản, đôi khi cần thực hiện các điều chỉnh tinh vi hơn dựa trên yêu cầu của người dùng. Ví dụ, người dùng có thể muốn làm cho đoạn văn mô tả chi tiết hơn, rút ngắn thành hai từ, hoặc thậm chí yêu cầu làm cho nội dung vui vẻ hơn hay thay đổi phần kết. Có ba cách chính để thực hiện những điều chỉnh này:

- Thao tác đầu vào: Điều chỉnh prompt để đưa ra các ràng buộc khác nhau cho mô hình.
- Thay đổi mô hình: Sử dụng các mô hình đặc biệt, như mô hình điền khuyết (infilling model) cho phép xem xét ngữ cảnh từ cả hai phía, đặc biệt hữu ích trong việc tạo mã.
- Thay đổi quá trình giải mã: Áp dụng các kỹ thuật đã đề cập trước đó, như sử dụng một bộ phân biệt (discriminator) để đánh giá tính mô tả của văn bản, hoặc sử dụng phương pháp lấy mẫu và xếp hạng để tạo ra kết quả phù hợp hơn với yêu cầu.

Human-in-the-loop decoding: choosing outputs

Một phương pháp phổ biến là sử dụng các kỹ thuật sampling và reranking, trong đó con người sẽ là người quyết định lựa chọn kết quả trả về. Ví dụ, trong Wordcraft, người dùng có thể thấy một tập hợp các lựa chọn và chọn văn bản để chèn vào. Tương tự, trong các ứng dụng như ChatGPT hoặc Bard, người dùng có tùy chọn "regenerate text" (tạo lại văn bản). Điều này cho phép người dùng từ chối văn bản không phù hợp và yêu cầu một đầu ra khác. Đây cũng là một cách để kiểm soát quá trình giải mã, nhưng thực hiện ở cấp độ con người thay vì thuật toán. Tuy nhiên, không nhất thiết phải có sự can thiệp của con người trong quá trình này.



Human-in-the-loop decoding: Tree of Thought

Nghiên cứu gần đây đã tập trung vào việc sử dụng mô hình để đưa ra quyết định trong quá trình tạo văn bản. Một ví dụ nổi bật là phương pháp "Tree of Thought". Ý tưởng chính của phương pháp này là:

- Tạo ra nhiều chuỗi ngắn (vài câu hoặc một bước lập luận).
- Sử dụng một mô hình để chọn chuỗi nào sẽ được tiếp tục phát triển.
- Áp dụng các ràng buộc khác nhau, như xếp hạng một tập hợp ba chuỗi hoặc dự đoán xem bước lập luận có hợp lý không.

Phương pháp này, thông qua việc sử dụng prompt, tạo ra một quá trình tương tự như tìm kiếm chùm (beam search). Thay vì đưa ra quyết định dựa trên xác suất của từng token, nó tạo ra nhiều câu cùng lúc và đưa ra quyết định dựa trên phản hồi của mô hình.

Nguồn phản hồi này có thể là một mô hình khác hoặc thậm chí là con người (nếu chấp nhận thời gian chờ đợi). Điều này cho phép đưa ra phản hồi ở cấp độ rộng hơn so với từng token riêng lẻ, hướng dẫn quá trình giải mã để tạo ra kết quả cuối cùng tốt hơn.

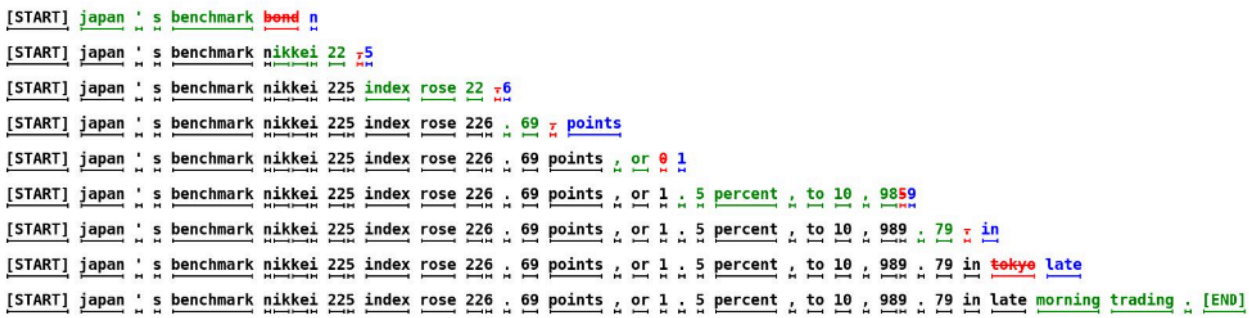
Practical Considerations

Practical considerations: speed (speculative decoding)

Trong quá trình tạo văn bản bằng mô hình ngôn ngữ lớn, phần lớn thời gian thường được dành cho quá trình giải mã (decoding). Điều này có thể gây trở ngại đáng kể cho các ứng dụng yêu cầu phản hồi nhanh như chat hay streaming.

Để giải quyết vấn đề này, một phương pháp được gọi là "Speculative decoding" đã được phát triển. Ý tưởng chính của phương pháp này là:

- Sử dụng một mô hình nhỏ hơn để dự đoán nhanh các token tiếp theo.
- Mô hình lớn hơn đóng vai trò như một "verifier", kiểm tra xem chuỗi được tạo ra có phù hợp với phân phối của nó hay không.



Trong hình minh họa, các token màu xanh lá cây được tạo bởi mô hình nhỏ hơn (amateur model). Mô hình lớn sẽ kiểm tra xem đầu ra có đi đúng hướng không. Nếu gặp một token không phù hợp, mô hình lớn sẽ từ chối token đó và tạo ra một token khác (thể hiện bằng các phần màu đỏ và xanh dương).

Phương pháp này giúp giảm đáng kể số lượng bước giải mã cần thực hiện bởi mô hình lớn. Ví dụ, thay vì phải thực hiện khoảng 20 bước giải mã, mô hình lớn chỉ cần thực hiện khoảng 8 bước, trong khi các bước còn lại được xử lý nhanh chóng bởi mô hình nhỏ hơn.

Khái niệm sử dụng mô hình nhỏ hơn như một xấp xỉ đã được chứng minh là rất hiệu quả. Có thể đây là một trong những kỹ thuật được sử dụng bởi các mô hình như ChatGPT hay Bard để tạo văn bản nhanh chóng.

Ngoài ra, cấu trúc mô hình thưa (sparse model architecture) cũng đóng vai trò quan trọng trong việc tăng tốc quá trình này, mặc dù điều này có thể liên quan đến khái niệm "mixture of experts" sẽ được thảo luận sau.

Libraries for decoding (and fast inference)

Trong lĩnh vực suy luận nhanh (fast inference), các thư viện như BLM đã triển khai nhiều kỹ thuật hữu ích. Chẳng hạn, chúng sử dụng phương pháp "speculative decoding" và các thủ thuật ở cấp phần cứng, như lựa chọn trọng số attention để tăng tốc độ suy diễn.

Ngoài ra, còn có những thư viện tuyệt vời cho việc "constraint decoding", cho phép bạn đặt ra các ràng buộc cho đầu ra. Ví dụ, bạn có thể yêu cầu tất cả đầu ra phải ở định dạng JSON, và thư viện sẽ áp dụng các ràng buộc bổ sung trong quá trình giải mã để đảm bảo điều đó.

Hầu hết các kỹ thuật mà chúng ta đã thảo luận, như "sampling", "mode seeking", "search" và đôi khi "MBR", đều được triển khai trong hầu hết các thư viện mà bạn sử dụng cho các mô hình như Hugging Face, Fairseq hoặc JAX.

Summary: two levels of decoding

Trong bài học, chúng ta đã thảo luận về hai phương pháp chính trong quá trình giải mã dựa trên phân phối ban đầu từ mô hình cho một token tiếp theo $P(y|X)$.

Phương pháp đầu tiên là trong từng bước giải mã, chúng ta có thể chọn một hàm nào đó $f(P(y|X))$ để điều chỉnh phân phối của token tiếp theo. Ví dụ, chúng ta có thể cắt đuôi của phân phối, điều chỉnh temperature, hoặc thêm thông tin từ một mô hình khác hoặc từ một mô hình phân biệt (discriminator model).

Phương pháp thứ hai là trong một phần lớn hơn của quá trình giải mã, chúng ta có thể chọn một hàm $g(s)$ để lựa chọn giữa các chuỗi. Điều này có thể bao gồm việc chọn giữa các token tiếp theo trong "beam search" khi chúng ta cắt tỉa các beam, hoặc lựa chọn từ các chuỗi đầy đủ khi thực hiện các phương pháp như "MBR" hay "sample and rerank".

Chúng ta cũng có thể thực hiện hai phương pháp này song song, tức là chọn một hàm khác để điều chỉnh phân phối token tiếp theo và một hàm rộng hơn để quyết định cách xử lý các chuỗi đầy đủ mà chúng ta nhận được từ phân phối đó.

Takeaways: decoding methods

Phương pháp giải mã có thể rất mạnh mẽ trong việc kiểm soát các đặc điểm đầu ra, đặc biệt khi bạn muốn áp đặt các ràng buộc cụ thể hoặc tích hợp các hàm phần thưởng mà không có trong quá trình đào tạo.

Một số phương pháp giải mã đắt đỏ có thể bù đắp cho mô hình kém hoặc chưa được đào tạo để thực hiện chính xác nhiệm vụ mong muốn. Tuy nhiên, không thể biến GPT-2 nhỏ thành GPT-4 chỉ bằng cách thay đổi phương pháp giải mã, nhưng có thể cải thiện hiệu suất ở mức độ nào đó bằng cách tăng thời gian suy luận.

Các phương pháp giải mã có sự đánh đổi giữa chất lượng, độ đa dạng và tốc độ suy luận. Ví dụ, sampling từ mô hình trực tiếp nhanh nhưng cho đầu ra kém chất lượng hơn, trong khi các phương pháp tìm kiếm như beam search cho đầu ra chất lượng cao hơn nhưng ít đa dạng hơn. Phương pháp như MBR (Minimum Bayes Risk) cho đầu ra chất lượng cao nhưng tốn nhiều thời gian suy luận.

Điều quan trọng là lựa chọn phương pháp giải mã có thể ảnh hưởng mạnh mẽ đến hiệu suất của mô hình. Việc để các thư viện chọn mặc định có thể bỏ lỡ nhiều tiềm năng cải thiện. Nếu mô hình của bạn có hành vi không mong muốn, hãy xem xét liệu phương pháp giải mã có đang áp đặt một số trực giác hoặc thiên kiến nào đó không, và liệu có thể thay đổi để cải thiện mà không cần đào tạo thêm.

Cuối cùng, việc dự đoán ảnh hưởng của các thao tác trong thời gian suy luận dễ hơn so với việc dự đoán ảnh hưởng của việc tinh chỉnh mô hình. Ví dụ, beam search với mô hình được huấn luyện theo maximum likelihood thường tạo ra các đầu ra ngắn hơn, trong khi greedy decoding có xu hướng tạo ra các đầu ra dài hơn và lặp lại nhiều hơn. Hiểu rõ các đặc điểm này giúp bạn biết cách khắc phục khi gặp vấn đề.

Resources

https://phontron.com/class/anlp2024/lectures/#generation-algorithms-feb-1