O modelo estatístico por trás dos Modelos de Linguagem Modernos

Alexandre Galvão Patriota

Departamento de Estatística,
Instituto de Matemática e Estatística,
Universidade de São Paulo.

Apresentação para o V EPBEST

Conteúdo

Modelos de linguagem

Modelo de linguagem

É uma função que processa, relaciona e gera continuações coerentes com base em padrões aprendidos de grandes coleções, como:

- artigos, livros científicos, romances, notícias;

Modelo de linguagem

É uma função que processa, relaciona e gera continuações coerentes com base em padrões aprendidos de grandes coleções, como:

- artigos, livros científicos, romances, notícias;
- transcrições de diálogos (podcasts, entrevistas);

Modelo de linguagem

É uma função que processa, relaciona e gera continuações coerentes com base em padrões aprendidos de grandes coleções, como:

- artigos, livros científicos, romances, notícias;
- transcrições de diálogos (podcasts, entrevistas);
- mensagens instantâneas, postagens em redes sociais, etc.

Cronologia dos modelos de linguagem

Prof. Fábio Cozman:

- 1966: Chatbot Eliza (regras: "se X, então Y").

Cronologia dos modelos de linguagem

Prof. Fábio Cozman:

- 1966: Chatbot Eliza (regras: "se X, então Y").
- 1993: Modelos de linguagem com frequências simples.

Cronologia dos modelos de linguagem

Prof. Fábio Cozman:

- 1966: Chatbot Eliza (regras: "se X, então Y").
- 1993: Modelos de linguagem com frequências simples.
- 2013: *Embeddings* (representação vetorial de palavras).

Cronologia dos modelos de linguagem

Prof. Fábio Cozman:

- 1966: Chatbot Eliza (regras: "se X, então Y").
- 1993: Modelos de linguagem com frequências simples.
- 2013: *Embeddings* (representação vetorial de palavras).
- 2017: *Transformers* (mecanismo da atenção).

Cronologia dos modelos de linguagem

Prof. Fábio Cozman:

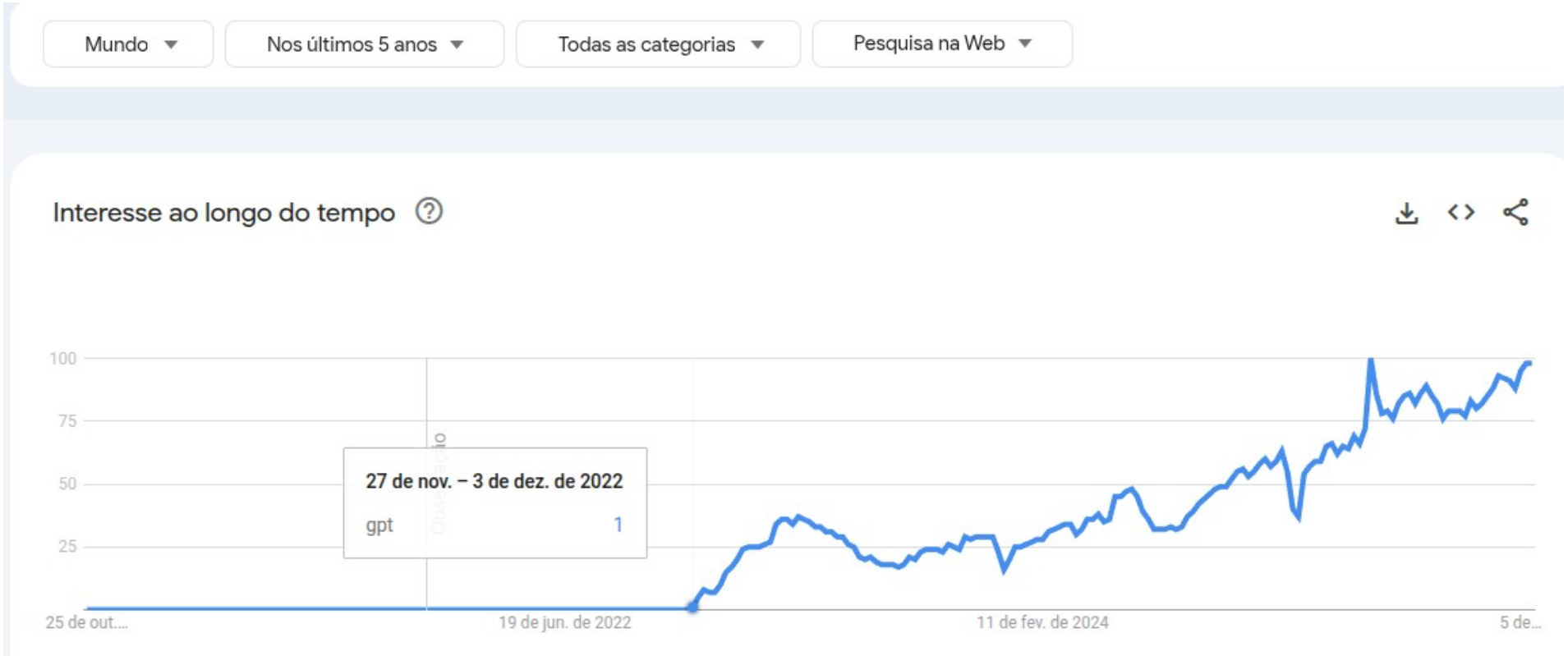
- 1966: Chatbot Eliza (regras: "se X, então Y").
- 1993: Modelos de linguagem com frequências simples.
- 2013: *Embeddings* (representação vetorial de palavras).
- 2017: *Transformers* (mecanismo da atenção).
- 2018-19: GPT-1 e GPT-2 (*Generative Pretrained models*)

Cronologia dos modelos de linguagem

Prof. Fábio Cozman:

- 1966: Chatbot Eliza (regras: "se X, então Y").
- 1993: Modelos de linguagem com frequências simples.
- 2013: *Embeddings* (representação vetorial de palavras).
- 2017: *Transformers* (mecanismo da atenção).
- 2018-19: GPT-1 e GPT-2 (*Generative Pretrained models*)
- 2020-Atual: GPT3, GPT-4, GPT-5, Llama, Gemini, Grok, modelos reflexivos, resolvedores de quebra-cabeças, geradores de áudios, imagens e vídeos, etc.

Google trends



Entrada e saída da função

A **entrada** de um modelo linguístico é também chamada de contexto (é o argumento da função).

A **saída** é um vetor/matriz de probabilidades para a próxima 'palavra', dado o contexto de entrada (é um ponto na imagem da função).

Entrada e saída da função

A **entrada** de um modelo linguístico é também chamada de contexto (é o argumento da função).

A **saída** é um vetor/matriz de probabilidades para a próxima 'palavra', dado o contexto de entrada (é um ponto na imagem da função).

A partir do contexto, calculam-se as probabilidades das próximas palavras em um vocabulário préfixado de tamanho V.

$$f(\text{contexto}) = \begin{pmatrix} P_{\hat{\theta}}(\ 1^{\underline{a}}\ \text{palavra} \mid \text{contexto}\ ) \\ \vdots \\ P_{\hat{\theta}}(\ V^{\underline{a}}\ \text{palavra} \mid \text{contexto}\ ) \end{pmatrix}$$

Exemplos

Por exemplo, para o contexto “Pedro é professor de”, as probabilidades condicionais poderiam ser:

Exemplos

Por exemplo, para o contexto “Pedro é professor de”, as probabilidades condicionais poderiam ser:

$$P_{\hat{\theta}}(\text{“matemática”}|\text{“Pedro é professor de”}) = 1.1\%$$

Exemplos

Por exemplo, para o contexto “Pedro é professor de”, as probabilidades condicionais poderiam ser:

$$P_{\hat{\theta}}(\text{“matemática”}|\text{“Pedro é professor de”}) = 1.1\%$$
$$P_{\hat{\theta}}(\text{“história”}|\text{“Pedro é professor de”}) = 1.0\%$$

Exemplos

Por exemplo, para o contexto “Pedro é professor de”, as probabilidades condicionais poderiam ser:

$$P_{\hat{\theta}}(\text{“matemática”}|\text{“Pedro é professor de”}) = 1.1\%$$
$$P_{\hat{\theta}}(\text{“história”}|\text{“Pedro é professor de”}) = 1.0\%$$
$$P_{\hat{\theta}}(\text{“casa”}|\text{“Pedro é professor de”}) = 0.00001\%$$

Exemplos

Por exemplo, para o contexto “Pedro é professor de”, as probabilidades condicionais poderiam ser:

$$P_{\hat{\theta}}(\text{“matemática”}|\text{“Pedro é professor de”}) = 1.1\%$$
$$P_{\hat{\theta}}(\text{“história”}|\text{“Pedro é professor de”}) = 1.0\%$$
$$P_{\hat{\theta}}(\text{“casa”}|\text{“Pedro é professor de”}) = 0.00001\%$$
$$P_{\hat{\theta}}(\text{“carro”}|\text{“Pedro é professor de”}) = 0.000001\%$$

Exemplos

Por exemplo, para o contexto “Pedro é professor de”, as probabilidades condicionais poderiam ser:

$$P_{\hat{\theta}}(\text{“matemática”}|\text{“Pedro é professor de”}) = 1.1\%$$
$$P_{\hat{\theta}}(\text{“história”}|\text{“Pedro é professor de”}) = 1.0\%$$
$$P_{\hat{\theta}}(\text{“casa”}|\text{“Pedro é professor de”}) = 0.00001\%$$
$$P_{\hat{\theta}}(\text{“carro”}|\text{“Pedro é professor de”}) = 0.000001\%$$

As probabilidades são estimadas por meio de algum método estatístico (e.g., máxima verossimilhança).

Tratamento inicial do texto

O texto pode ser representado por seus **caracteres** ou por **subpalavras** (tokens):

Tratamento inicial do texto

O texto pode ser representado por seus **caracteres** ou por **subpalavras** (tokens):

Caracteres:

“P”, “e”, “d”, “r”, “o”, “ ”, “é”, “ ”, “p”, “r”, “o”, “f”, “e”, “s”, “s”, “o”, “r”, “ ”, “d”, “e”

Tratamento inicial do texto

O texto pode ser representado por seus **caracteres** ou por **subpalavras** (tokens):

Caracteres:

“P”, “e”, “d”, “r”, “o”, “ ”, “é”, “ ”, “p”, “r”, “o”, “f”, “e”, “s”, “s”, “o”, “r”, “ ”, “d”, “e”

Subpalavras:

“Pe”, “dro”, “ ”, “é”, “ ”, “pro”, “fessor”, “ ”, “de”

Vocabulário de tokens

Com os tokens, podemos formar um vocabulário listando todos os tokens que serão usados na modelagem.

“<PAD>”	“<UNK>”	“<EOS>”	“ ”	...	“a”	...	“z”	“0”	...	“9”	“Aba”	...	“Zhu”
1	2	3	4		34		59	60		69	70		50257

Vocabulário fictício

Vocabulário de tokens

Com os tokens, podemos formar um vocabulário listando todos os tokens que serão usados na modelagem.

“<PAD>”	“<UNK>”	“<EOS>”	“ ”	...	“a”	...	“z”	“0”	...	“9”	“Aba”	...	“Zhu”
1	2	3	4		34		59	60		69	70		50257

Vocabulário fictício

Cada ‘frase’ pode ser escrita em termos das posições de seus tokens no vocabulário:

“Pe”	“dro”	“ ”	“é”	“ ”	“pro”	“fessor”	“ ”	“de”
226	136	4	25	4	1002	106	4	43

obs: numeração fictícia.

Representação one-hot

Podemos representar cada token por um vetor de zeros com 1 na posição do token.

Representação one-hot

Podemos representar cada token por um vetor de zeros com 1 na posição do token.

Considere o vocabulário $\{A, B, C, D\}$. A sequência A, A, B, B, C pode ser representada por

Representação one-hot

Podemos representar cada token por um vetor de zeros com 1 na posição do token.

Considere o vocabulário $\{A, B, C, D\}$. A sequência A, A, B, B, C pode ser representada por

$$\underbrace{\begin{pmatrix}1\\0\\0\\0\end{pmatrix}}_{A}, \underbrace{\begin{pmatrix}1\\0\\0\\0\end{pmatrix}}_{A}, \underbrace{\begin{pmatrix}0\\1\\0\\0\end{pmatrix}}_{B}, \underbrace{\begin{pmatrix}0\\1\\0\\0\end{pmatrix}}_{B}, \underbrace{\begin{pmatrix}0\\0\\1\\0\end{pmatrix}}_{C}$$

cujas posições no vocabulário são $1, 1, 2, 2, 3$.

Representação *one-hot*

A versão *one-hot* é a mais adequada para representar os vetores aleatórios.

Representação *one-hot*

A versão *one-hot* é a mais adequada para representar os vetores aleatórios.

Denotaremos o token em sua versão *one-hot* observada por x e em sua versão *one-hot* aleatória por X.

Representação *one-hot*

A versão *one-hot* é a mais adequada para representar os vetores aleatórios.

Denotaremos o token em sua versão *one-hot* observada por x e em sua versão *one-hot* aleatória por X.

Uma sequência máxima de k tokens será denotada por $x_{1:k} = (x_1, \ldots, x_k)$, em que $x_{1:1} = x_1$. As versões aleatórias serão representadas pela letra maiúscula.

Representação *one-hot*

A versão *one-hot* é a mais adequada para representar os vetores aleatórios.

Denotaremos o token em sua versão *one-hot* observada por x e em sua versão *one-hot* aleatória por X.

Uma sequência máxima de k tokens será denotada por $x_{1:k} = (x_1, \ldots, x_k)$, em que $x_{1:1} = x_1$. As versões aleatórias serão representadas pela letra maiúscula.

Se a sequência tiver menos do que k tokens, completa-se com o token <pad>.

Amostra (*batch*)

A amostra $x^{(1)}, \ldots, x^{(n)}$ é composta por pares $x^{(i)} = (x^{(i)}_{1:k}, x^{(i)}_{k+1})$.

([Lorem ipsum dolor sit], amet), consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. ([Nam arcu libero, nonummy eget, consectetuer], id), vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. ([Morbi dolor], nulla), malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum. Nam dui ligula, fringilla a, euismod sodales, sollicitudin vel, wisi. ([Morbi auctor lorem non justo. Nam lacus libero, pretium], at), lobortis vitae, ultricies et, tellus. Donec aliquet, tortor sed accumsan bibendum, erat ligula aliquet magna, vitae ornare odio metus a mi. Morbi ac orci et nisl hendrerit mollis. Suspendisse ut massa. Cras nec ante. Pellentesque a nulla. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Aliquam tincidunt urna. ([Nulla ullamcorper vestibulum], turpis). Pellentesque cursus luctus mauris. Nulla malesuada porttitor diam. Donec felis erat, congue non, volutpat at, tincidunt tristique, libero. Vivamus viverra fermentum felis. ([Donec nonummy pellentesque ante. Phasellus adipiscing semper], elit). Proin fermentum massa ac quam. Sed diam turpis, molestie vitae, placerat a, molestie nec, leo. Maecenas lacinia. Nam ipsum ligula, eleifend at, accumsan nec, suscipit a, ipsum. Morbi blandit ligula feugiat magna. Nunc eleifend consequat lorem. Sed lacinia nulla vitae enim. Pellentesque tincidunt purus vel magna. Integer non enim. Praesent euismod nunc eu purus. Donec bibendum quam in tellus. ([Nullam cursus pulvinar lectus. Donec et mi. Nam vulputate metus eu], enim.) Vestibulum pellentesque felis eu massa.

Modelo Estatístico

Modelo Estatístico

Seja V o tamanho do vocabulário e considere k o contexto máximo de entrada. A distribuição do próximo token X_{k+1} dado o contexto $x_{1:k} = (x_1, \ldots, x_k)$ é modelada abaixo:

Modelo Estatístico

Seja V o tamanho do vocabulário e considere k o contexto máximo de entrada. A distribuição do próximo token X_{k+1} dado o contexto $x_{1:k} = (x_1, \ldots, x_k)$ é modelada abaixo:

Para cada $s = 1, \ldots, k$,

$$X_{s+1} | X_{1:s} = x_{1:s} \sim Multinomial_V \left(1, \boldsymbol{p}_{\boldsymbol{\theta}}^{(s)}(x_{1:s}) \right)$$

em que

$\boldsymbol{p}_{\boldsymbol{\theta}}^{(s)}$ é o vetor de probabilidades condicionais dados os s tokens anteriores,

Função de verossimilhança

A função de probabilidade conjunta para $X = (X_{1:k}, X_{k+1})$ é

$$P_{\boldsymbol{\theta}}(X = x) = \prod_{s=1}^{k} \prod_{i=1}^{V} p_{\boldsymbol{\theta},i}^{(s)}(\boldsymbol{x}_{1:s})^{x_{i,s+1}},$$

em que $x = (x_{1:k}, x_{k+1})$, $x_{i,s+1}$ e $p_{\boldsymbol{\theta},i}^{(s)}$ são os i-ésimos elementos de x_{s+1} e $\boldsymbol{p}_{\theta}^{(s)}$; k é o tamanho máximo.

Função de verossimilhança

A função de probabilidade conjunta para $X = (X_{1:k}, X_{k+1})$ é

$$P_{\boldsymbol{\theta}}(X = x) = \prod_{s=1}^{k} \prod_{i=1}^{V} p_{\boldsymbol{\theta},i}^{(s)} (\boldsymbol{x}_{1:s})^{x_{i,s+1}},$$

em que $x = (x_{1:k}, x_{k+1})$, $x_{i,s+1}$ e $p_{\boldsymbol{\theta},i}^{(s)}$ são os i-ésimos elementos de x_{s+1} e $\boldsymbol{p}_{\theta}^{(s)}$; k é o tamanho máximo.

Para uma amostra aleatória (*batch*) $x^{(1)}, \ldots, x^{(n)}$, a função de verossimilhança é

$$L(\boldsymbol{\theta}) = \prod_{j=1}^{n} P_{\boldsymbol{\theta}}(X = x^{(j)})$$

Estimação

Os textos são divididos em ‘pedaços’ aleatórios em que a ordem das palavras dentro dos pedaços é mantida.

Estimação

Os textos são divididos em ‘pedaços’ aleatórios em que a ordem das palavras dentro dos pedaços é mantida.

A amostra $x^{(1)}, \ldots, x^{(n)}$ é composta por pares $x^{(i)} = (x^{(i)}_{1:k}, x^{(i)}_{k+1})$.

Estimação

Os textos são divididos em ‘pedaços’ aleatórios em que a ordem das palavras dentro dos pedaços é mantida.

A amostra $x^{(1)}, \ldots, x^{(n)}$ é composta por pares $x^{(i)} = (x^{(i)}_{1:k}, x^{(i)}_{k+1})$.

A função de verossimilhança considera todas as subsequências simultaneamente ($s = 1, \ldots, k$).

Estimação

O ‘aprendizado’ ocorre após maximizar a função de verossimilhança com respeito a θ.

Estimação

O ‘aprendizado’ ocorre após maximizar a função de verossimilhança com respeito a θ.

Os algoritmos de otimização (SGD, Adam, AdamW) utilizados consideram maximização/minimização por bateladas *batches*.

Estimação

O ‘aprendizado’ ocorre após maximizar a função de verossimilhança com respeito a θ.

Os algoritmos de otimização (SGD, Adam, AdamW) utilizados consideram maximização/minimização por bateladas *batches*.

As derivadas são obtidas por meio da regra da cadeia (*backpropagation*) e sua versão numérica pelo *AutoGrad*.

Estimação

O ‘aprendizado’ ocorre após maximizar a função de verossimilhança com respeito a θ.

Os algoritmos de otimização (SGD, Adam, AdamW) utilizados consideram maximização/minimização por bateladas *batches*.

As derivadas são obtidas por meio da regra da cadeia (*backpropagation*) e sua versão numérica pelo *AutoGrad*.

A função de verossimilhança é penalizada para controlar o excesso de parâmetros.

Geração de texto

Após estimar os parâmetros, utiliza-se uma técnica para escolher o próximo token: *greedy*, **top-k**, top-p, etc.

Geração de texto

Após estimar os parâmetros, utiliza-se uma técnica para escolher o próximo token: *greedy*, **top-k**, top-p, etc.

O texto é gerado de forma autoregressiva:

$$\boldsymbol{f}(x_1, \ldots, x_k) \to x_{k+1}, \quad \boldsymbol{f}(x_2, \ldots, x_k, x_{k+1}) \to x_{k+2}, \quad \ldots$$

Geração de texto

Após estimar os parâmetros, utiliza-se uma técnica para escolher o próximo token: *greedy*, **top-k**, top-p, etc.

O texto é gerado de forma autoregressiva:

$$\boldsymbol{f}(x_1, \ldots, x_k) \to x_{k+1}, \quad \boldsymbol{f}(x_2, \ldots, x_k, x_{k+1}) \to x_{k+2}, \quad \ldots$$

O token gerado é inserido no contexto e o processo é repetido até gerar o token EOS.

Geração de texto

Após estimar os parâmetros, utiliza-se uma técnica para escolher o próximo token: *greedy*, **top-k**, top-p, etc.

O texto é gerado de forma autoregressiva:

$$\boldsymbol{f}(x_1, \ldots, x_k) \to x_{k+1}, \quad \boldsymbol{f}(x_2, \ldots, x_k, x_{k+1}) \to x_{k+2}, \quad \ldots$$

O token gerado é inserido no contexto e o processo é repetido até gerar o token EOS.

A forma funcional de $\boldsymbol{p}_{\boldsymbol{\theta}}^{(s)}$ é definida pelo **Decoder do Transformer** ('Attention is all you need')

Função p_θ: Vaswani et al. (2017)

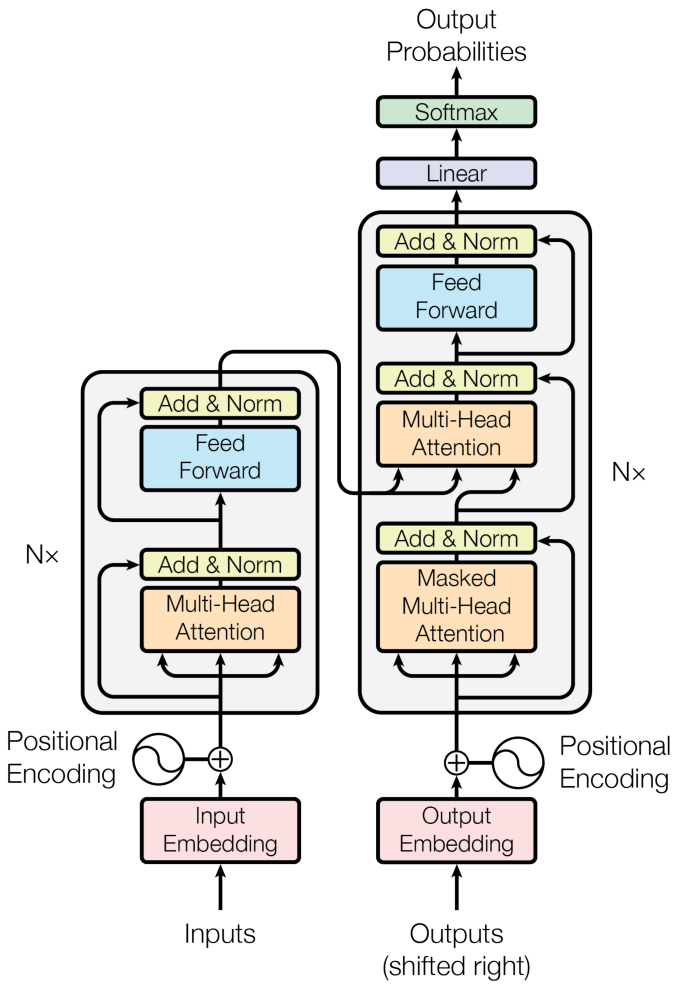


Figure 1: The Transformer - model architecture.

Parâmetros do vocabulário de tokens

Cada token no vocabulário é escrito como um vetor em um espaço de dimensão m (768 no GPT2 e $12\,288$ no GPT3).

Parâmetros do vocabulário de tokens

Cada token no vocabulário é escrito como um vetor em um espaço de dimensão m (768 no GPT2 e $12\,288$ no GPT3).

Embedding

$$\begin{array}{ccc} \text{“<PAD>”} & \longrightarrow & (a_{11} \ldots a_{1m}) \\ \text{“<UNK>”} & \longrightarrow & (a_{21} \ldots a_{2m}) \\ \vdots & \vdots & \vdots \\ \text{“Zhu”} & \longrightarrow & (a_{V1} \ldots a_{Vm}) \end{array}$$

Matriz de embeddings

$$A = \begin{bmatrix} a_{11} & \cdots & a_{1m} \\ a_{21} & \cdots & a_{2m} \\ \vdots & \ddots & \vdots \\ a_{V1} & \cdots & a_{Vm} \end{bmatrix}$$

Parâmetros do vocabulário de tokens

Cada token no vocabulário é escrito como um vetor em um espaço de dimensão m (768 no GPT2 e $12\,288$ no GPT3).

Embedding

$$\begin{array}{ccc} \text{“<PAD>”} & \longrightarrow & (a_{11} \ldots a_{1m}) \\ \text{“<UNK>”} & \longrightarrow & (a_{21} \ldots a_{2m}) \\ \vdots & \vdots & \vdots \\ \text{“Zhu”} & \longrightarrow & (a_{V1} \ldots a_{Vm}) \end{array}$$

Matriz de embeddings

$$A = \begin{bmatrix} a_{11} & \cdots & a_{1m} \\ a_{21} & \cdots & a_{2m} \\ \vdots & \ddots & \vdots \\ a_{V1} & \cdots & a_{Vm} \end{bmatrix}$$

Considerando $V = 50\,257$ e $m = 768$, temos $\approx$ 39 milhões de parâmetros (GPT3 ≈ 618 milhões)

Modelo simplificado

De forma simplificada, o Decoder do Transformer pode ser representado por:

$$\underbrace{\boldsymbol{x}}_{\text{input}} \to \underbrace{\boldsymbol{x}^{(1)}}_{A[\boldsymbol{x},]+P} \to \Big[\underbrace{\boldsymbol{x}^{(2l)}}_{\text{Att}} \to \underbrace{\boldsymbol{x}^{(3l)}}_{\text{FFN}} \Big]_{l=1}^{N} \to \underbrace{\boldsymbol{x}^{(4)}}_{\text{Final}} \to \underbrace{\boldsymbol{p}_{\boldsymbol{\theta}}(\boldsymbol{x})}_{\text{softmax}}$$

Cada camada introduz mais parâmetros para serem estimados. É uma espécie de regressão logística multivariada sofisticada.

O que esse modelo moderno pode fazer?

Capacidades

Um modelo de linguagem grande é capaz de

- continuar um texto a partir de uma entrada e manter a coerência do assunto em diversas línguas.

Capacidades

Um modelo de linguagem grande é capaz de

- continuar um texto a partir de uma entrada e manter a coerência do assunto em diversas línguas.
- ajudar a provar teoremas, desde que guiados e orientados por um especialista.

Capacidades

Um modelo de linguagem grande é capaz de

- continuar um texto a partir de uma entrada e manter a coerência do assunto em diversas línguas.
- ajudar a provar teoremas, desde que guiados e orientados por um especialista.
- fazer resumos, escrever códigos em R, Python, C, html, java, etc.

Capacidades

Um modelo de linguagem grande é capaz de

- continuar um texto a partir de uma entrada e manter a coerência do assunto em diversas línguas.
- ajudar a provar teoremas, desde que guiados e orientados por um especialista.
- fazer resumos, escrever códigos em R, Python, C, html, java, etc.
- explicar algumas teorias consolidadas, resolver alguns quebra-cabeças, corrigir a gramática, tornar um texto mais formal.

Capacidades

Um modelo de linguagem grande é capaz de

- continuar um texto a partir de uma entrada e manter a coerência do assunto em diversas línguas.
- ajudar a provar teoremas, desde que guiados e orientados por um especialista.
- fazer resumos, escrever códigos em R, Python, C, html, java, etc.
- explicar algumas teorias consolidadas, resolver alguns quebra-cabeças, corrigir a gramática, tornar um texto mais formal.

Problema: o modelo ainda erra de forma confiante.

Capacidades

Um modelo de linguagem grande é capaz de

- continuar um texto a partir de uma entrada e manter a coerência do assunto em diversas línguas.
- ajudar a provar teoremas, desde que guiados e orientados por um especialista.
- fazer resumos, escrever códigos em R, Python, C, html, java, etc.
- explicar algumas teorias consolidadas, resolver alguns quebra-cabeças, corrigir a gramática, tornar um texto mais formal.

Problema: o modelo ainda erra de forma confiante.

Não se sabe se inserir algum módulo ou modificar o treinamento resolveriam esses problemas.

Modelo de linguagem no R

- Modelo pequeno treinado em Shakespeare: https://github.com/AGPatriota/GPT4R

Modelo de linguagem no R

- Modelo pequeno treinado em Shakespeare: https://github.com/AGPatriota/GPT4R
- GPT-2 com pesos estimados pela da OpenAI: https://github.com/AGPatriota/GPT-2-for-R

Modelo de linguagem no R

- Modelo pequeno treinado em Shakespeare: https://github.com/AGPatriota/GPT4R
- GPT-2 com pesos estimados pela da OpenAI: https://github.com/AGPatriota/GPT-2-for-R
- Adição em um modelo de linguagem ultra pequeno: https://github.com/AGPatriota/ALGA-R

Pesquisas em andamento

- Resumo e classificação de prontuários médicos (em coautoria com Andrey Sarmento)

Pesquisas em andamento

- Resumo e classificação de prontuários médicos (em coautoria com Andrey Sarmento)
- Demonstrações de teoremas na teoria de conjuntos ZFC.

Pesquisas em andamento

- Resumo e classificação de prontuários médicos (em coautoria com Andrey Sarmento)
- Demonstrações de teoremas na teoria de conjuntos ZFC.
- Estudo dos modelos de linguagem a partir da teoria estatística.

Referências

- Vaswani, A., et al. (2017). Attention is all you need. *In Advances in Neural Information Processing Systems (NeurIPS)*.
- Brown, T.B., et al. (2020). Language Models are Few-Shot Learners, OpenAI, `arxiv.org/abs/2005.14165`.
- OpenAI (2023). GPT-4 Technical Report, `arxiv.org/abs/2303.08774`
- Yun, C., et al. (2020). Are transformers universal approximators of sequence-to-sequence functions?, *Published as a conference paper at ICLR 2020*

Obrigado!

Quer ingressar na Pós-graduação em Estatística no IME-USP?

Consulte:
ime.usp.br/pos-estatistica/ingresso