

[VER PLANOS](#)[PROGRAMAÇÃO](#)[FRONT-END](#)[DATA SCIENCE](#)[INTELIGÊNCIA ARTIFICIAL](#)[DEVOPS](#)[UX & DESIGN](#)[MOBILE](#)[INOVAÇÃO & GESTÃO](#)

Alura > Cursos de Inteligência Artificial > Cursos de IA para Negócios >
Conteúdos de IA para Negócios >
Primeiras aulas do curso Engenharia de Prompt: criando prompts eficazes para IA
Generativa

Engenharia de Prompt: criando prompts eficazes para IA Generativa

Conceitos iniciais - Apresentação

0:00 / 1:57

Boas-vindas a este curso de **Engenharia de Prompt** na Alura! Eu sou o **Fabrício Carraro**, e irei te acompanhar ao longo dessa jornada.

Audiodescrição: Fabrício se descreve como um homem de pele bronzeada, cabelos pretos lisos e curtos, e olhos pretos. Ele usa um piercing na sobrancelha esquerda, veste uma camiseta azul com o logotipo da Alura estampado, e está em frente a uma parede branca com uma porta também branca.

O que vamos aprender?

Neste curso, vamos explorar diversos tópicos juntos.

Começaremos aprendendo o que é um **modelo de linguagem**, conhecendo um pouco sobre como eles funcionam em segundo plano, além de alguns dos **conceitos mais relevantes** quando falamos sobre modelos de linguagens, ou modelos de linguagens grandes, os **LLMs (Large Language Models)**.

Em seguida, vamos aprofundar nos **princípios da Engenharia de Prompt**: o que é Engenharia de Prompt? Quais são as técnicas e os princípios mais importantes que utilizaremos para criar o prompt ideal, de modo a obter as respostas desejadas a partir desses modelos de linguagem?

Depois, vamos explorar as **técnicas mais conhecidas**, por meio de artigos científicos publicados pelas empresas que estão em alta, como *OpenAI*, *Google*, *Anthropic*, *Mistral AI*, e muitas outras que criam modelos de linguagem. Essas empresas publicaram artigos e iremos nos aprofundar neles para entender como funcionam, bem como utilizá-los na prática.

Por fim, vamos conhecer **outras técnicas** menos utilizadas na prática e mais utilizadas por pessoas que irão **aplicar os modelos de maneira programática**, utilizando API e código.

Essas técnicas também podem ser interessantes para você, mesmo que não seja da área de computação, para conhecê-las e saber como elas funcionam internamente.

Conclusão

Esperamos que você tenha um bom trajetória neste curso conosco!

Conceitos iniciais - A proximidade semântica

0:00 / 12:33

Neste primeiro vídeo, falaremos sobre o que é um **modelo de linguagem** e como ele funciona.

Modelos de linguagem

Quando observamos modelos de linguagem, como os famosos *Large Language Models*, vemos uma parte muito específica, que são modelos treinados com muitos dados**, praticamente, com a internet inteira. Porém, o conceito de modelos de linguagem é anterior a isso.

Imagine que, na língua **portuguesa**, temos a seguinte frase:

"Eu gosto de pizza".

Quando usamos o verbo "gostar", é evidente que, após esse verbo, conjugado ou não, precisamos da **preposição** "de". Assim, temos "Eu gosto de pizza", "Eu gosto de viajar", "Eu gosto de você", e assim por diante. Quando analisamos outros idiomas, começamos a perceber que nem sempre é assim.

Na língua **inglesa**, por exemplo, temos a frase "*I like pizza*". Nesse caso, o verbo "*to like*" equivale ao verbo "gostar" do português, mas **não usa a preposição** obrigatória do português.

Em inglês, isso não existe. Quando falamos sobre um verbo, como, por exemplo, na frase "Eu gosto de viajar", podemos dizer "*I like to travel*" ou "*I like traveling*". Porém,

não tem a preposição "de" obrigatória. O padrão da língua inglesa é diferente do padrão da língua portuguesa.

Quando observamos um idioma completamente diferente, como **turco**, podemos encontrar o verbo sempre no final da frase. Assim, em turco, "Eu gosto de pizza" seria "Pizza severim".

O substantivo "pizza" vem no começo, antes do verbo. Esse padrão do idioma turco é completamente diferente, tanto em relação ao português, quanto em relação ao inglês.

O que é um modelo de linguagem?

Analisados esses exemplos, **o que é um modelo de linguagem?**

Falamos sobre idiomas, pois a ideia é passar dados para que o modelo **aprenda padrões da linguagem humana e relações entre as palavras**.

Em português, o modelo que recebe os dados aprende, por exemplo, que depois do verbo "gostar" e suas conjugações, como "Eu gosto de" ou "Você gosta de", sempre teremos a preposição "de".

Já em inglês, não será assim. Da mesma forma, em turco, teremos primeiro o substantivo e depois o verbo. Portanto, o modelo irá aprender esses padrões e relações entre as palavras.

O que são *word embeddings*?

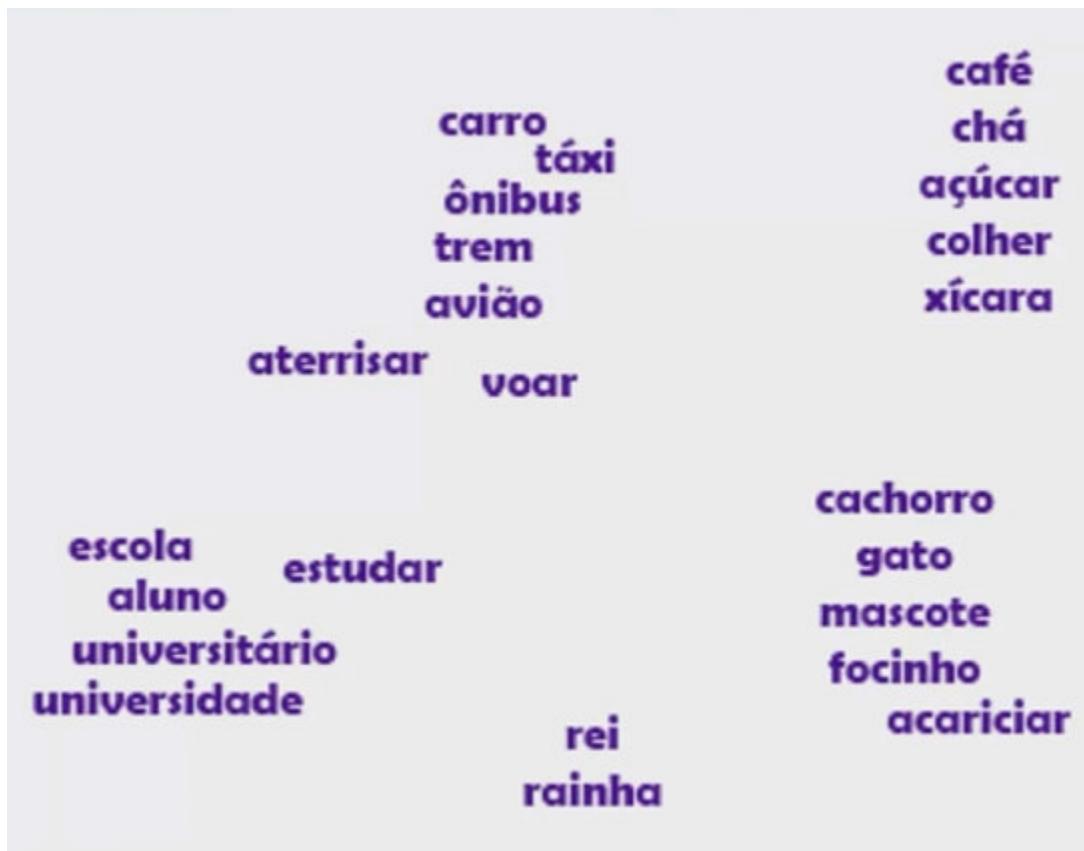
Como o modelo aprende padrões e relações?

Existe o conceito de **word embeddings**. Teoricamente, eles são **representações distribuídas de palavras ou sub palavras numéricas**; ou seja, é um número que representa cada palavra. Além disso, os embeddings conseguem representar a **proximidade semântica** entre uma palavra e outra.

Imagine a palavra "carro". Analisando um gráfico de embeddings, provavelmente, essa palavra estará muito próxima das palavras "táxi", "ônibus", "trem", pois são todos **meios de transporte**.

Da mesma forma, a palavra "carro" fica longe das palavras "cachorro", "focinho", "rei", "café", "escola", "universidade", entre outras, pois são palavras de categorias diferentes.

Observe o word embedding abaixo:



O word embedding consegue **entender o significado da palavra**. Quando observamos um gráfico desses vetores, notamos que a distância entre as palavras "carro" e "ônibus" é muito menor do que a distância entre as palavras "carro" e "escola", ou entre as palavras "carro" e "cachorro", por exemplo.

Essa é uma base mais teórica, mas também é possível verificar alguns números:

#	avião
voar	0,95
trem	0,78
ônibus	0,76
...	...
focinho	0,015

#	avião
xícara	0,09
abastecer	0,61

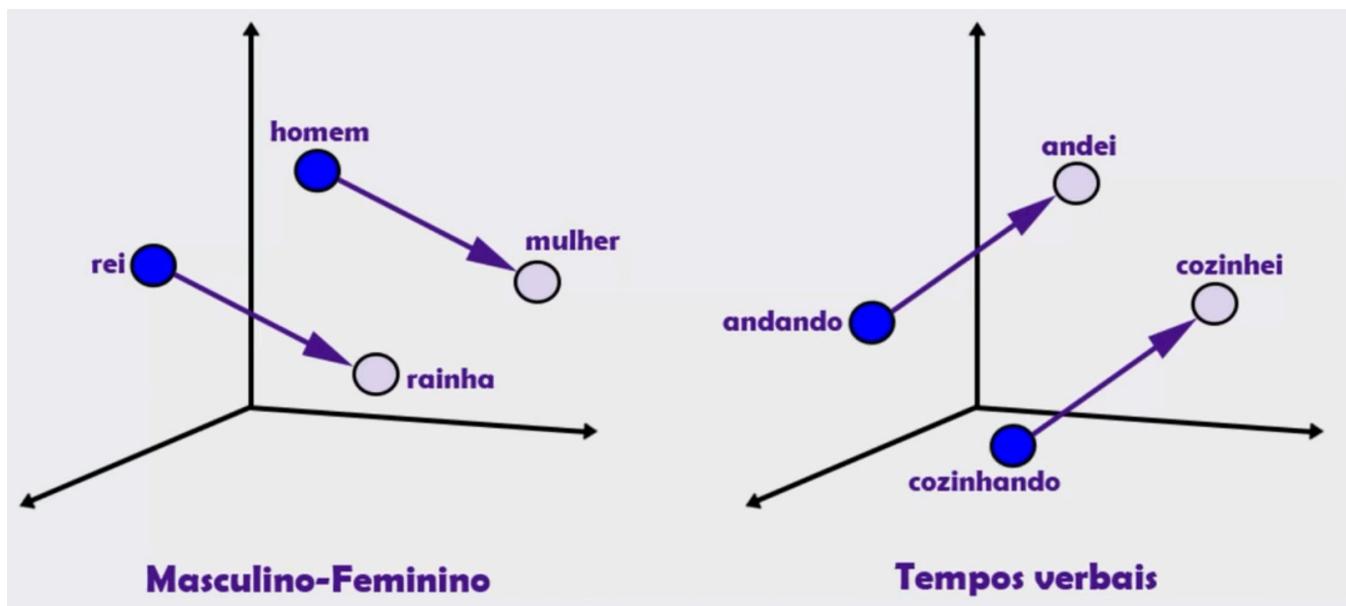
Observação! Os números da tabela acima são fictícios.

Por exemplo: entre 0 e 1, quão próxima a palavra "avião", que está no topo da tabela, é da palavra "voar"? As palavras "avião" e "voar" possuem uma relação muito forte, então entre 0 e 1, poderia ser **0,95** o valor da relação entre elas. Com a palavra "trem", temos uma relação de **0,78**, pois ambos são meios de transporte. Da mesma forma, a relação com "ônibus" é **0,76**.

Por outro lado, a relação de "avião" com "focinho" seria **0.0015**, pois não há proximidade. Embora sejam números fictícios, conseguimos ter uma ideia de como os modelos funcionam internamente.

Word embeddings e operações matemáticas

Outra coisa muito interessante que pessoas cientistas conseguiram resolver a partir desses embeddings é que conseguimos fazer **operações matemáticas**. Como mencionamos, no momento de calcular e treinar os modelos de linguagem, ele não terá, por exemplo, a palavra "avião", mas sim uma **representação numérica** da palavra "avião". Porém, como fazer cálculos com esses números?



Na imagem acima, temos como exemplo as palavras "homem" e "mulher". Da mesma forma, encontramos "rei" e "rainha". A palavra "homem", geralmente, é relacionada a "rei", e "rainha" a "mulher". Trata-se de uma questão de **gênero** das palavras.

A partir disso, foi identificado que quando temos, por exemplo, a palavra "rei" e a palavra "homem", ao fazer um cálculo matemático substituindo a palavra "rei" pela palavra "mulher", o word embedding muda para a palavra "rainha".

O mesmo acontece no exemplo à direita, mas para **tempos verbais**. Com o verbo "andando", se colocarmos uma soma, uma subtração, fizermos um cálculo com um valor específico, por exemplo, do passado, ele saberá que o resultado deve ser "andei" ou "andou". Também temos o exemplo do verbo "cozinhando", que está no gerúndio e se associa a "cozinhei" ou "cozinhou".

É muito interessante como os word embeddings funcionam internamente e como eles conseguem fazer associações entre muitas palavras. Evidentemente, trouxemos apenas o exemplo da palavra "avião" em relação a uma série de palavras, mas imagine que isso é uma tabela gigante, um gráfico tridimensional com todas as palavras relacionadasumas às outras. Assim funcionam os modelos de linguagem.

Prevendo as próximas palavras

Uma vez treinados os modelos de linguagem, eles tentam **prever a próxima palavra** em um contexto. Nesse caso, enviamos uma pergunta ou uma frase, e ele sempre tenta adivinhar a próxima palavra que faz mais sentido no contexto inserido. É exatamente isso que os modelos de linguagem, como o **ChatGPT**, o **Gemini** do Google, e todos os outros fazem: adivinhar a próxima palavra.

É como se fosse a funcionalidade de **preenchimento automático (autocomplete)** do celular, que consegue dar três opções de quais são as próximas palavras mais prováveis que estarão na frase. Os modelos de linguagem também fazem isso, mas de uma maneira muito mais complexa e com um **treinamento** muito maior, com **muito mais palavras e conhecimento** por trás dele.

Utilizando o **ChatGPT**

Com o site do **ChatGPT** aberto, vamos usá-lo no modo gratuito *GPT-4o* para fazer uma brincadeira de tentar adivinhar como funciona o ChatGPT. Para isso, preparamos um **prompt**, que é simplesmente um **pedido** que faremos para a ferramenta:

Vamos simular como funciona o ChatGPT. Para cada frase que eu escrever no pr

Começaremos dessa forma a primeira conversa com o ChatGPT. Como resposta, ele diz "Entendi.", então vamos enviar o primeiro exemplo utilizando o prompt abaixo:

Eu gosto muito de inteligência artificial e quero trabalhar

Como retorno, temos seguinte lista de palavras:

com (45%)

nessa (25%)

na (20%)

no (5%)

em (5%)

Temos a palavra "com" com **45%** de probabilidade de ser a próxima palavra. Também foi retornada a palavra "nessa"; poderia ser "nessa área", "quero trabalhar com isso", "quero trabalhar na área".

Agora, suponha que escolhemos a primeira palavra que ele sugeriu: "com". A partir disso, enviaremos um novo prompt:

Eu gosto muito de inteligência artificial e quero trabalhar com

Agora, recebemos novas respostas:

isso (40%)

ela (30%)

desenvolvimento (15%)

você (10%)

projetos (5%)

A cada palavra, o ChatGPT tenta adivinhar qual é a maior probabilidade da próxima.

Além disso, outra coisa muito interessante dos modelos de linguagem é que eles conseguem **manter o contexto da conversa**, que foi, na verdade, o grande motivo da disruptão que eles causaram.

Vamos abrir um chat novo do ChatGPT e fazer a seguinte pergunta:

Qual era o esporte do atleta Rogério Ceni?

Como resposta, o ChatGPT diz o seguinte:

Rogério Ceni foi um jogador de futebol brasileiro. Ele é amplamente conhecido por sua carreira como goleiro, especialmente no São Paulo Futebol Clube, onde se destacou não apenas por suas habilidades defensivas, mas também por ser um exímio cobrador de faltas e pênaltis, marcando mais de 130 gols ao longo de sua carreira.

Em seguida, vamos enviar uma nova pergunta:

Qual foi o título mais relevante da carreira dele?

Note que não escrevemos "da carreira do Rogério Ceni", mas sim "da carreira dele". Ele responde:

O título mais relevante da carreira de Rogério Ceni é, provavelmente, a conquista do Mundial de Clubes da FIFA em 2005 com o São Paulo Futebol Clube. Nesse torneio, Rogério Ceni teve uma atuação decisiva, especialmente na final contra o Liverpool, onde fez defesas cruciais e ainda foi eleito o melhor jogador da competição. Essa vitória solidificou sua reputação como um dos maiores goleiros da história do futebol.

Portanto, o ChatGPT consegue manter o contexto, tanto da pergunta anterior, quanto da própria resposta que ele forneceu. Cada vez que enviamos um prompt e recebemos uma resposta de um modelo de linguagem, ele mantém todo o contexto anterior. É isso que os torna tão poderosos: a capacidade de **não se perder em contextos longos**.

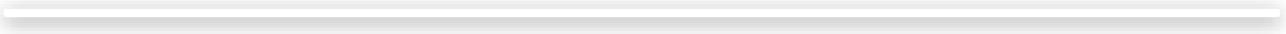
Conclusão

Pensando no exemplo anterior, quando testamos a previsão das palavras, será que o modelo de linguagem **sempre escolhe a palavra com maior probabilidade?**

Descobriremos no próximo vídeo!

Conceitos iniciais - O que são tokens?

0:00 / 10:16



No final do vídeo anterior, fizemos a seguinte pergunta:

*Será que o modelo **sempre tenta prever a próxima palavra?** Isto é, ele sempre escolhe a palavra que tem a **maior probabilidade?***

Na verdade, não exatamente. Existe um conceito que as pessoas desenvolvedoras dos modelos de linguagem inventaram, chamado **temperatura**. Vamos entender o que isso significa?

Temperatura

A **temperatura** é basicamente a probabilidade de um modelo escolher outra palavra que não seja a mais provável. Isso foi criado porque notaram que, quando o modelo escolhia sempre a palavra mais provável, a resposta ficava robótica demais, sem os truques que nós, humanos, usamos ao escrever um texto real. Mostraremos, na prática, como funciona o conceito de **temperatura**.

Entendendo o conceito de temperatura na prática

Podemos acessar diretamente o *Playground* da plataforma da *OpenAI*, onde, geralmente, as pessoas desenvolvedoras **exploram** mais, **criam** os modelos, e também **testam** modelos novos. Conseguimos experimentar bastante, pois a ferramenta oferece várias opções de testes.

Uma delas, no canto superior direito, é a "**Temperature**", que vai de 0 a 2. Por padrão, ela é definida como 1, exatamente a metade. Vamos fazer alguns **testes** para entender como ela funciona?

Realizando o teste com temperatura 0

Começaremos enviando o seguinte *prompt* no chat:

Complete a frase a seguir:

Eu gosto muito de comer...

Inicialmente, testaremos a **temperatura em 0**. Como retorno, recebemos:

Eu gosto muito de comer chocolate.

Feito o teste, podemos apagar a resposta clicando no botão de menos (-) à direita dela. Feito isso, conseguimos executar novamente o comando, clicando no botão "**Run**", no canto inferior direito.

Agora, recebemos como resultado a seguinte frase:

Eu gosto de comer pizza.

Note que ele mudou apenas um pouco a resposta. Vamos testar mais uma vez? Após executar novamente o prompt, recebemos a mesma resposta do primeiro envio:

Eu gosto muito de comer chocolate.

Após rodar algumas vezes seguidas, notamos que o Playground quase sempre retorna a resposta "Eu gosto muito de comer chocolate.". No nosso caso, aconteceu uma única vez a resposta "pizza".

Realizando o teste com temperatura 1

Agora, vamos testar a **temperatura no valor intermediário**, ou seja, 1. Feito isso, executaremos novamente o prompt abaixo:

Complete a frase a seguir:

Eu gosto muito de comer...

Após rodar algumas vezes, recebemos primeiro a resposta "Eu gosto muito de comer pizza.", depois "Eu gosto muito de comer chocolate.", em seguida "Eu gosto muito de comer chocolate" se repete duas vezes, e depois vem "Eu gosto muito de comer pizza." novamente.

Ele exibe a resposta com as palavras "pizza" e "chocolate" algumas vezes, então ainda existe uma preferência por essas duas comidas, mas as frequências estão variando mais.

Evidentemente, isso é um experimento anedótico, não um experimento científico, mas conseguimos entender melhor a temperatura na prática através desses testes.

Realizando o teste com temperatura 1.4

Agora vamos definir a **temperatura em 1.4**, por exemplo, e executar novamente o teste.

Complete a frase a seguir:

Eu gosto muito de comer...

pizza! É uma delícia, não acha?

Acima, temos a primeira resposta gerada pelo Playground. Além de usar a opção "pizza", ele completou a frase com "É uma delícia, não acha?". É a primeira vez que isso acontece nesse caso.

Ao executar novamente, recebemos a mesma resposta dos testes anteriores: "Eu gosto muito de comer pizza". Na sequência, recebemos "Eu gosto muito de comer chocolate quente".

Note que, quanto **mais alta** for a temperatura, **mais variadas** serão as respostas retornadas. Conforme dito anteriormente, quando colocamos uma temperatura **muito baixa**, ele tende a sempre, ou quase sempre, escolher a primeira resposta mais provável. Ou seja, da lista de palavras que ele pode escolher, ele irá optar quase sempre pela palavra com a **probabilidade mais alta**.

Conforme aumentamos a temperatura de 0 para 0.7, para 1, para 1.4, e assim em diante, talvez ele escolha outras palavras diferentes da primeira (a segunda, a terceira, a quarta, ou a quinta).

Realizando o teste com temperatura 2

Quando colocamos uma **temperatura muito alta**, talvez o Playground possa escolher uma palavra que sequer está entre as cinco listadas no *ChatGPT*.

Vamos observar o que acontece quando definimos a **temperatura em 2**?

Complete a frase a seguir:

Eu gosto muito de comer...

The server had an error processing your request. Sorry about that! You can retry your request, or contact us through our help center at help.openai.com if you keep seeing this error.

Note que houve um **erro**, ou seja, ele nem sabe como processar tudo isso. Podemos testar a temperatura um pouco mais baixa, como em **1.71**, por exemplo.

sorvete nos dias quentes e pipoca quando assisto a filmes.

É uma resposta diferente, mas quando executamos novamente, recebemos o mesmo erro de antes. Portanto, quando colocamos uma temperatura muito alta, **acima de 1.5**, para essa plataforma da OpenAI, em vez de usar palavras normais que têm uma probabilidade alta de acontecer, ele começa a inserir símbolos, códigos, coisas com probabilidade muito baixa de ser a próxima palavra.

No entanto, conforme alteramos isso, indicamos para a ferramenta que ela deve escolher uma próxima palavra com probabilidade muito baixa, provavelmente. Essa é a ideia da temperatura.

Tokens

Existe outro conceito que podemos abordar: o de **tokens**.

Falamos muito sobre palavras até o momento, mas modelos de linguagem não funcionam com palavras, e sim com tokens, que podemos definir como a **unidade básica em modelos de linguagem, uma representação de palavras ou sub palavras**.

Imagine, por exemplo, a palavra "**infeliz**". Na língua portuguesa, poderíamos ter dois tokens:

O token "in";

E o token "feliz".

O token "in" também pode ser usado com outras palavras, como "indivisível", "incrível", entre várias outras. Por exemplo: em "indispensável", um token seria "in", e o outro seria "dispensável".

Esse é apenas um conceito básico de tokens para explicar a questão dos modelos de linguagem, pois utilizamos exemplos com palavras anteriormente, mas, na verdade, são **partes de palavras**.

Às vezes, um token corresponde a uma palavra inteira, mas, às vezes, corresponde a uma parte de uma palavra, e isso depende do **algoritmo de tokenização** usado pela empresa.

Conhecendo o algoritmo de tokenização da *OpenAI*

Vamos conhecer o algoritmo de tokenização da própria **OpenAI**, disponibilizado na página [Tokenizer](#). Nela, conseguimos visualizar o tokenizador usado para os modelos GPT-3.5 e GPT-4. Em breve, teremos o do GPT-4o, modelo mais atual no momento da gravação deste curso.

Podemos testar uma mesma frase em três idiomas e conferir quantos tokens ela corresponde em casa. Começaremos com uma frase em **português**:

Eu estou fazendo um curso de Engenharia de Prompt na Alura porque eu quero (

Uma frase básica, de **152 caracteres e 37 tokens**. "Eu" é um token, o espaço seguido de "est" é outro token, "ou" é outro token, e assim em diante. Nesse caso, poderia ser "está", "estou", "estamos".

Da mesma forma, espaço seguido de "faz" um token. Poderíamos ter, por exemplo, "fazer" ou "fazendo". Espaço seguido de "curso" também é um token por si só.

O importante no momento é lembrar desse número: **37 tokens** para a frase em português. Agora, vamos testar a mesma frase, mas em **inglês**.

I'm taking a Prompt Engineering course at alura because I want to learn a lo



Nesse caso, temos **28 tokens**. Isso pode acontecer porque, na tradução, são usadas menos palavras, mas também há outros motivos. Observe, por exemplo, o token formado por espaço e "taking".

Da mesma forma, espaço seguido de "Prompt" é um token, e espaço "Engineering" também. Poderia ser "Engineer" e "ing", mas foi representado o espaço seguido de "Engineering" como uma unidade.

Quando esses modelos são treinados, nas línguas mais usadas no treinamento, **os tokens tendem a ser a própria palavra**. O português é um idioma com menos presença no treinamento dos modelos GPT-3.5 ou GPT-4 do que o inglês; por isso, é um pouco diferente e temos mais tokens em português.

Por último, vamos testar um idioma bem diferente: o **grego**.

Káνω ένα μάθημα Prompt Engineering στην Alura επειδή θέλω να μάθω πολλά περισσότερα.



Agora, temos **113 tokens**. O número de palavras é praticamente o mesmo, com **151 caracteres**, mas praticamente **cada letra é um token**. Observe, por exemplo, a palavra "περισσότερα". O grego é uma língua pouco representada no treinamento desses modelos, então quase toda letra vira um token.

Conclusão

Conhecemos a **base dos modelos de linguagem**: como eles funcionam; o conceito de temperatura; bem como o conceito de tokens. No próximo vídeo, continuaremos aprendendo sobre modelos de linguagem, mas entraremos na **engenharia de prompt** efetivamente!

Sobre o curso Engenharia de Prompt: criando prompts eficazes para IA Generativa

O [curso Engenharia de Prompt: criando prompts eficazes para IA Generativa](#) possui **86 minutos de vídeos**, em um total de **35 atividades**. Gostou? Conheça nossos outros [cursos de IA para Negócios](#) em [Inteligência Artificial](#), ou leia nossos [artigos de Inteligência Artificial](#).

Matricule-se e comece a estudar com a gente hoje! Conheça outros tópicos abordados durante o curso:

Conceitos iniciais

Engenharia de Prompt

Few-Shot Prompt

Chain-of-Thought Prompt

Mais técnicas de Prompt

Aprenda IA para Negócios acessando integralmente esse e outros cursos, comece hoje!

1 ano 2 anos OFERTA

[Compre e leve 2 meses grátis!](#)

PLUS

Impulsiona a sua carreira com os melhores cursos e faça parte da maior comunidade tech.

12x **R\$109**

1 ano de Alura

À vista R\$1.308

MATRICULE-SE

Matricule-se no plano PLUS e garanta:

22 Carreiras

2.025 Cursos

Acesso a TODOS os cursos por 14 meses

Certificado

Mentorias em grupo com especialistas

Comunidade exclusiva

Acesso ao conteúdo das Imersões

App Android e iOS para estudar onde quiser

PRO

Acelere o seu aprendizado com a IA da Alura e prepare-se para o mercado internacional.

12x **R\$149**

1 ano de Alura

À vista R\$1.788

MATRICULE-SE

Todos os benefícios do PLUS e mais vantagens exclusivas:

Luri, a inteligência artificial da Alura

Alura Língua - Inglês e Espanhol

A melhor opção para sua carreira 🎉

ULTRA LAB

Para estudantes ultra comprometidos atingirem seu objetivo mais rápido.

12x **R\$299**

1 ano de Alura

À vista R\$3.588

MATRICULE-SE

Todos os benefícios do PRO e mais vantagens exclusivas:

Luri, com mensagens ILIMITADAS

Luri Vision, a IA que enxerga suas dúvidas

Talent Lab, mentorias individuais e vagas exclusivas

6 Ebooks da Casa do Código



Pague com cartão de crédito em até 12x, PayPal, NuPay ou Pix

Garantimos cancelamento gratuito em até 7 dias

[Conheça os Planos para Empresas](#)

Nossas redes e apps



Institucional

[Sobre nós](#)

[Trabalhe na Alura](#)

[Para Empresas](#)

[Para Sua Escola](#)

[Política de Privacidade](#)

[Compromisso de Integridade](#)

[Termos de Uso](#)

[Documentos Institucionais](#)

[Status](#)

[Uma empresa do grupo Alun](#)

A Alura

[Como Funciona](#)

[Formações](#)

[Plataforma](#)

[Depoimentos](#)

[Instrutores\(as\)](#)

[Dev em <T>](#)

[Luri, a inteligência artificial da Alura](#)

[IA Conference 2025](#)

[Cursos imersivos](#)

[Certificações](#)

Conteúdos

Alura Cases

Imersões

Artigos

Podcasts

Artigos de educação corporativa

Imersão IA 4º edição

Fale Conosco

Email e telefone

Perguntas frequentes

Novidades e Lançamentos

bins.br@gmail.com

ENVIAR

CURSOS

Cursos de Programação

Lógica | Python | PHP | Java | .NET | Node JS | C | Computação | Jogos | IoT

Cursos de Front-end

HTML, CSS | React | Angular | JavaScript | jQuery

Cursos de Data Science

Ciência de dados | BI | SQL e Banco de Dados | Excel | Machine Learning | NoSQL | Estatística

Cursos de Inteligência Artificial

IA para Programação | IA para Dados

Cursos de DevOps

AWS | Azure | Docker | Segurança | IaC | Linux

Cursos de UX & Design

Usabilidade e UX | Vídeo e Motion | 3D

Cursos de Mobile

Flutter | iOS e Swift | Android, Kotlin | Jogos

Cursos de Inovação & Gestão

Métodos Ágeis | Softskills | Liderança e Gestão | Startups | Vendas

CURSOS UNIVERSITÁRIOS FIAP

Graduação | Pós-graduação | MBA