

Confira o [Cookbook](https://github.com/google-gemini/cookbook) (https://github.com/google-gemini/cookbook) da nova API Gemini e nosso [fórum da comunidade](https://discuss.ai.google.dev/?hl=pt-br) (https://discuss.ai.google.dev/?hl=pt-br).


translated by Google


Esta página foi traduzida pela API Cloud Translation

(//cloud.google.com/translate/?hl=pt-br).

[Switch to English](#)

Pesquisa de documentos com embeddings

 **Executar** no Google Colab [_](https://colab.research.google.com/github/google/generative-ai-docs/blob/main/site/en/gemini-api/tutorials/document_search.ipynb?hl=pt-br)(https://colab.research.google.com/github/google/generative-ai-docs/blob/main/site/en/gemini-api/tutorials/document_search.ipynb?hl=pt-br).

 **Consulte** o código-fonte no [GitHub](https://github.com/google-gemini/generative-ai-docs/blob/main/site/en/gemini-api/tutorials/document_search.ipynb) [_](https://github.com/google-gemini/generative-ai-docs/blob/main/site/en/gemini-api/tutorials/document_search.ipynb)(https://github.com/google-gemini/generative-ai-docs/blob/main/site/en/gemini-api/tutorials/document_search.ipynb).

Visão geral

Neste exemplo, demonstramos como usar a API Gemini para criar embeddings e realizar a pesquisa de documentos. Você usará a biblioteca de cliente Python para criar uma incorporação de palavras que permita comparar strings de pesquisa, ou perguntas, com o conteúdo do documento.

Neste tutorial, você vai usar embeddings para pesquisar documentos em um conjunto de documentos e fazer perguntas relacionadas ao Google Car.

Pré-requisitos

Este guia de início rápido pode ser executado no Google Colab.

Para concluir este guia de início rápido no seu próprio ambiente de desenvolvimento, verifique se ele atende aos seguintes requisitos:

- Python 3.9 ou superior

- Uma instalação de jupyter para executar o notebook.

Configuração

Primeiro, faça o download e instale a biblioteca Python da API Gemini.

```
$ pip install -U -q google.generativeai
```

```
import textwrap
import numpy as np
import pandas as pd

import google.generativeai as genai
import google.ai.generativeai as glm

# Used to securely store your API key
from google.colab import userdata

from IPython.display import Markdown
```

Obter uma chave de API

Antes de usar a API Gemini, é necessário ter uma chave de API. Se você ainda não tiver uma, crie uma chave com um clique no Google AI Studio.

Gerar uma chave de API (<https://makersuite.google.com/app/apikey?hl=pt-br>)

No Colab, adicione a chave ao gerenciador de secrets abaixo do "", no painel à esquerda. Dê o nome `API_KEY` a ela.

Quando você tiver a chave de API, transmita-a ao SDK. Faça isso de duas maneiras:

- Coloque a chave na variável de ambiente `GOOGLE_API_KEY`. O SDK vai selecioná-la automaticamente de lá.
- Transmita a chave para `genai.configure(api_key=...)`

```
# Or use `os.getenv('API_KEY')` to fetch an environment variable.
API_KEY=userdata.get('API_KEY')
```

```
genai.configure(api_key=API_KEY)
```

Importante :a seguir, escolha um modelo. Qualquer modelo de embedding funcionará para este tutorial, mas, para aplicativos reais, é importante escolher um modelo específico e insistir com ele. As saídas de modelos diferentes não são compatíveis entre si.

Observação: no momento, a API Gemini está disponível apenas em algumas regiões (https://developers.generativeai.google/available_regions?hl=pt-br).

```
for m in genai.list_models():
    if 'embedContent' in m.supported_generation_methods:
        print(m.name)
```

```
models/embedding-001
models/embedding-001
```

Geração de embeddings

Nesta seção, você vai aprender a gerar embeddings para um texto usando embeddings da API Gemini.

Mudanças de API em embeddings com embeddings-001 do modelo

Para o novo modelo de embeddings, embedding-001, há um novo parâmetro de tipo de tarefa e o título opcional (válido somente com `task_type=RETRIEVAL_DOCUMENT`).

Esses novos parâmetros se aplicam somente aos modelos de embeddings mais recentes. Os tipos de tarefa são:

Tipo de tarefa	Descrição
RETRIEVAL_QUERY	Especifica que o texto é uma consulta em uma configuração de pesquisa/recuperação.
RETRIEVAL_DOCUMENT	Especifica que o texto é um documento em uma configuração de pesquisa/recuperação.
SEMANTIC_SIMILARITY	Especifica o texto a ser usado para similaridade textual semântica (STS).

Tipo de tarefa	Descrição
CLASSIFICAÇÃO	Especifica que os embeddings serão usados para classificação.
CLUSTERING	Especifica que os embeddings serão usados para clustering.

Observação: a especificação de um **title** para **RETRIEVAL_DOCUMENT** proporciona embeddings de melhor qualidade para recuperação.

```
title = "The next generation of AI for developers and Google Workspace"
sample_text = ("Title: The next generation of AI for developers and Google Workspace"
               "\n"
               "Full article:\n"
               "\n"
               "Gemini API & Google AI Studio: An approachable way to explore and prototype with ")

model = 'models/embedding-001'
embedding = genai.embed_content(model=model,
                                content=sample_text,
                                task_type="retrieval_document",
                                title=title)

print(embedding)
```

```
{'embedding': [0.034585103, -0.044509504, -0.027291223, 0.0072681927, 0.061689284, 0.0...
```

Como criar um banco de dados de embeddings

Aqui estão três textos de amostra para criar o banco de dados de embeddings. Você vai usar a API Gemini para criar embeddings de cada um dos documentos. Transforme-os em um DataFrame para uma melhor visualização.

```
DOCUMENT1 = {
    "title": "Operating the Climate Control System",
    "content": "Your Googlecar has a climate control system that allows you to adjust"
DOCUMENT2 = {
    "title": "Touchscreen",
    "content": "Your Googlecar has a large touchscreen display that provides access to
```

```
DOCUMENT3 = {
    "title": "Shifting Gears",
    "content": "Your Googlecar has an automatic transmission. To shift gears, simply m"

documents = [DOCUMENT1, DOCUMENT2, DOCUMENT3]
```

Organizar o conteúdo do dicionário em um DataFrame para uma melhor visualização.

```
df = pd.DataFrame(documents)
df.columns = ['Title', 'Text']
df
```

Texto
perando o sistema de climatização...
eu Googlecar tem uma grande tela touchscreen...
eslocamento de engrenagem Seu Googlecar tem tampa automática...

Confira os embeddings de cada um desses corpos de texto. Adicione essas informações ao DataFrame.

```
# Get the embeddings of each text and add to an embeddings column in the dataframe
def embed_fn(title, text):
    return genai.embed_content(model=model,
                               content=text,
                               task_type="retrieval_document",
                               title=title)["embedding"]

df['Embeddings'] = df.apply(lambda row: embed_fn(row['Title'], row['Text']), axis=1)
df
```


Use a função `find_best_passage` para calcular os produtos escalares e, em seguida, classifique o DataFrame do maior para o menor valor do produto escalar para recuperar a passagem relevante do banco de dados.

```
def find_best_passage(query, dataframe):
    """
    Compute the distances between the query and each document in the dataframe
    using the dot product.
    """
    query_embedding = genai.embed_content(model=model,
                                          content=query,
                                          task_type="retrieval_query")
    dot_products = np.dot(np.stack(dataframe['Embeddings']), query_embedding["embedding"])
    idx = np.argmax(dot_products)
    return dataframe.iloc[idx]['Text'] # Return text from index with max value
```

Confira o documento mais relevante do banco de dados:

```
passage = find_best_passage(query, df)
passage
```

```
'Shifting Gears  Your Googlecar has an automatic transmission. To shift gears, simply r
```

Formulário de perguntas e respostas

Vamos tentar usar a API de geração de texto para criar um sistema de perguntas e respostas. Insira seus dados personalizados abaixo para criar um exemplo simples de perguntas e respostas. Você ainda usará o produto escalar como uma métrica de similaridade.

```
def make_prompt(query, relevant_passage):
    escaped = relevant_passage.replace("'", "").replace('"', "").replace("\n", " ")
    prompt = textwrap.dedent("""You are a helpful and informative bot that answers quest:
    Be sure to respond in a complete sentence, being comprehensive, including all releva
    However, you are talking to a non-technical audience, so be sure to break down compl
    strike a friendly and conversational tone. \
    If the passage is irrelevant to the answer, you may ignore it.
    QUESTION: '{query}'
    PASSAGE: '{relevant_passage}'
```

```
ANSWER:
""").format(query=query, relevant_passage=escaped)

return prompt
```

```
prompt = make_prompt(query, passage)
print(prompt)
```

You are a helpful and informative bot that answers questions using text from the refer
QUESTION: 'How do you shift gears in the Google car?'
PASSAGE: 'Shifting Gears Your Googlecar has an automatic transmission. To shift gea

ANSWER:

Escolha um dos modelos de geração de conteúdo do Gemini para encontrar a resposta à sua consulta.

```
for m in genai.list_models():
    if 'generateContent' in m.supported_generation_methods:
        print(m.name)
```

```
models/gemini-pro
models/gemini-pro-vision
models/gemini-ultra
```

```
model = genai.GenerativeModel('gemini-ultra')
answer = model.generate_content(prompt)
```

```
Markdown(answer.text)
```

O trecho fornecido não contém informações sobre como mudar de marcha em um carro do Google, então não é possível responder à sua pergunta nesta fonte.

Próximas etapas

Para saber mais sobre como usar os embeddings, confira os [exemplos](https://ai.google.dev/gemini-api/examples?keywords=embed&hl=pt-br) (https://ai.google.dev/gemini-api/examples?keywords=embed&hl=pt-br) disponíveis. Para saber como usar outros serviços na API Gemini, acesse o [guia de início rápido do Python](https://ai.google.dev/gemini-api/tutorials/python_quickstart?hl=pt-br) (https://ai.google.dev/gemini-api/tutorials/python_quickstart?hl=pt-br).

Exceto em caso de indicação contrária, o conteúdo desta página é licenciado de acordo com a [Licença de atribuição 4.0 do Creative Commons](https://creativecommons.org/licenses/by/4.0/) (https://creativecommons.org/licenses/by/4.0/), e as amostras de código são licenciadas de acordo com a [Licença Apache 2.0](https://www.apache.org/licenses/LICENSE-2.0) (https://www.apache.org/licenses/LICENSE-2.0). Para mais detalhes, consulte as [políticas do site do Google Developers](https://developers.google.com/site-policies?hl=pt-br) (https://developers.google.com/site-policies?hl=pt-br). Java é uma marca registrada da Oracle e/ou afiliadas.

Última atualização 2024-04-26 UTC.