Documentation API reference Forum Help

Embeddings

Learn how to turn text into numbers, unlocking use cases like search.

New embedding models

`text-embedding-3-small` and `text-embedding-3-large`, our newest and most performant embedding models are now available, with lower costs, higher multilingual performance, and new parameters to control the overall size.

What are embeddings?

OpenAI's text embeddings measure the relatedness of text strings. Embeddings are commonly used for:

- **Search** (where results are ranked by relevance to a query string)
- **Clustering** (where text strings are grouped by similarity)
- **Recommendations** (where items with related text strings are recommended)
- **Anomaly detection** (where outliers with little relatedness are identified)
- **Diversity measurement** (where similarity distributions are analyzed)
- **Classification** (where text strings are classified by their most similar label)

An embedding is a vector (list) of floating point numbers. The distance between two vectors measures their relatedness. Small distances suggest high relatedness and large distances suggest low relatedness.

Visit our pricing page to learn about Embeddings pricing. Requests are billed based on the number of tokens in the input.

How to get embeddings

To get an embedding, send your text string to the embeddings API endpoint along with the embedding model name (e.g. `text-embedding-3-small`). The response will contain an embedding (list of floating point numbers), which you can extract, save in a vector database, and use for many different use cases:

Example: Getting embeddings node

```
import OpenAI from "openai";

const openai = new OpenAI();

async function main() {
  const embedding = await openai.embeddings.create({
    model: "text-embedding-3-small",
    input: "Your text string goes here",
    encoding_format: "float",
  });

  console.log(embedding);
}

main();
```

The response will contain the embedding vector along with some additional metadata.

Example embedding response json

```json
{
  "object": "list",
  "data": [
    {
      "object": "embedding",
      "index": 0,
      "embedding": [
        -0.006929283495992422,
        -0.005336422007530928,
        ... (omitted for spacing)
        -4.547132266452536e-05,
        -0.024047505110502243
      ],
    }
  ],
  "model": "text-embedding-3-small",
  "usage": {
    "prompt_tokens": 5,
    "total_tokens": 5
  }
}
```

By default, the length of the embedding vector will be 1536 for `text-embedding-3-small` or 3072 for `text-embedding-3-large`. You can reduce the dimensions of the embedding by passing in the dimensions parameter without the embedding losing its concept-representing properties. We go into more detail on embedding dimensions in the embedding use case section.

Embedding models

OpenAI offers two powerful third-generation embedding model (denoted by `-3` in the model ID). You can read the embedding v3 announcement blog post for more details.

Usage is priced per input token, below is an example of pricing pages of text per US dollar (assuming ~800 tokens per page):

MODEL	~ PAGES PER DOLLAR	PERFORMANCE ON MTEB EVAL	MAX INPUT
text-embedding-3-small	62,500	62.3%	8191
text-embedding-3-large	9,615	64.6%	8191
text-embedding-ada-002	12,500	61.0%	8191

Use cases

Here we show some representative use cases. We will use the Amazon fine-food reviews dataset for the following examples.

Obtaining the embeddings

The dataset contains a total of 568,454 food reviews Amazon users left up to October 2012. We will use a subset of 1,000 most recent reviews for illustration purposes. The reviews are in English and tend to be positive or negative. Each review has a ProductId, UserId, Score, review title (Summary) and review body (Text). For example:

PRODUCT ID	USER ID	SCORE	SUMMARY	TEXT
B001E4KFG0	A3SGXH7AUHU8GW	5	Good Quality Dog Food	I have bought several of the Vitality canned...
B00813GRG4	A1D87F6ZCVE5NK	1	Not as Advertised	Product arrived labeled as Jumbo Salted Peanut...

We will combine the review summary and review text into a single combined text. The model will encode this combined text and output a single vector embedding.

Get_embeddings_from_dataset.ipynb ↗

```python
from openai import OpenAI
client = OpenAI()

def get_embedding(text, model="text-embedding-3-small"):
   text = text.replace("\n", " ")
   return client.embeddings.create(input = [text], model=model).data[0].em

df['ada_embedding'] = df.combined.apply(lambda x: get_embedding(x, model='
df.to_csv('output/embedded_1k_reviews.csv', index=False)
```

To load the data from a saved file, you can run the following:

```python
import pandas as pd

df = pd.read_csv('output/embedded_1k_reviews.csv')
df['ada_embedding'] = df.ada_embedding.apply(eval).apply(np.array)
```

Reducing embedding dimensions

Question answering using embeddings-based search

Text search using embeddings

Code search using embeddings

Recommendations using embeddings

Data visualization in 2D

Embedding as a text feature encoder for ML algorithms

Classification using the embedding features

Zero-shot classification

Obtaining user and product embeddings for cold-start recommendation

Clustering

Frequently asked questions

How can I tell how many tokens a string has before I embed it?

In Python, you can split a string into tokens with OpenAI's tokenizer `tiktoken`.

Example code:

```python
import tiktoken

def num_tokens_from_string(string: str, encoding_name: str) -> int:
    """Returns the number of tokens in a text string."""
    encoding = tiktoken.get_encoding(encoding_name)
    num_tokens = len(encoding.encode(string))
    return num_tokens

num_tokens_from_string("tiktoken is great!", "cl100k_base")
```

For third-generation embedding models like `text-embedding-3-small`, use the `cl100k_base` encoding.

More details and example code are in the OpenAI Cookbook guide how to count tokens with tiktoken.

How can I retrieve K nearest embedding vectors quickly?

For searching over many vectors quickly, we recommend using a vector database. You can find examples of working with vector databases and the OpenAI API in our

Cookbook on GitHub.

Which distance function should I use?

We recommend cosine similarity. The choice of distance function typically doesn't matter much.

OpenAI embeddings are normalized to length 1, which means that:

- Cosine similarity can be computed slightly faster using just a dot product
- Cosine similarity and Euclidean distance will result in the identical rankings

Can I share my embeddings online?

Yes, customers own their input and output from our models, including in the case of embeddings. You are responsible for ensuring that the content you input to our API does not violate any applicable law or our Terms of Use.

Do V3 embedding models know about recent events?

No, the `text-embedding-3-large` and `text-embedding-3-small` models lack knowledge of events that occurred after September 2021. This is generally not as much of a limitation as it would be for text generation models but in certain edge cases it can reduce performance.