Embeddings

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 sent.

To see embeddings in action, check out our code samples

- Classification
- Topic clustering
- Search
- Recommendations

Browse Samples

How to get embeddings

To get an embedding, send your text string to the embeddings API endpoint along with a choice of embedding model ID (e.g., `text-embedding-ada-002`). The response will contain an embedding, which you can extract, save, and use.

Example requests:

python Copy

Example: Getting embeddings

```python
response = openai.Embedding.create(
    input="Your text string goes here",
    model="text-embedding-ada-002"
)
embeddings = response['data'][0]['embedding']
```

Example response:

```
{
  "data": [
    {
      "embedding": [
        -0.006929283495992422,
        -0.005336422007530928,
        ...
        -4.547132266452536e-05,
        -0.024047505110502243
      ],
      "index": 0,
      "object": "embedding"
    }
  ],
  "model": "text-embedding-ada-002",
  "object": "list",
  "usage": {
    "prompt_tokens": 5,
    "total_tokens": 5
  }
}
```

See more Python code examples in the OpenAI Cookbook.

When using OpenAI embeddings, please keep in mind their limitations and risks.

Embedding models

OpenAI offers one second-generation embedding model (denoted by `-002` in the model ID) and 16 first-generation models (denoted by `-001` in the model ID).

We recommend using text-embedding-ada-002 for nearly all use cases. It’s better, cheaper, and simpler to use. Read the blog post announcement.

MODEL GENERATION	TOKENIZER	MAX INPUT TOKENS	KNOWLEDGE CUTOFF
V2	cl100k_base	8191	Sep 2021
V1	GPT-2/GPT-3	2046	Aug 2020

Usage is priced per input token, at a rate of $0.0004 per 1000 tokens, or about ~3,000 pages per US dollar (assuming ~800 tokens per page):

MODEL	ROUGH PAGES PER DOLLAR	EXAMPLE PERFORMANCE ON BEIR SEARCH EVAL
text-embedding-ada-002	3000	53.9
-davinci--001	6	52.8
-curie--001	60	50.9
-babbage--001	240	50.4
-ada--001	300	49.0

Second-generation models

MODEL NAME	TOKENIZER	MAX INPUT TOKENS	OUTPUT DIMENSIONS
text-embedding-ada-002	cl100k_base	8191	1536

First-generation models (not recommended)

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.

Obtain_dataset.ipynb

```
def get_embedding(text, model="text-embedding-ada-002"):
   text = text.replace("\n", " ")
   return openai.Embedding.create(input = [text], model=model)['data'][0

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:

```
import pandas as pd

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

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

Text search using embeddings

Code search using embeddings

Recommendations using embeddings

Limitations & risks

Our embedding models may be unreliable or pose social risks in certain cases, and may cause harm in the absence of mitigations.

Social bias

Limitation: The models encode social biases, e.g. via stereotypes or negative sentiment towards certain groups.

We found evidence of bias in our models via running the SEAT (May et al, 2019) and the Winogender (Rudinger et al, 2018) benchmarks. Together, these benchmarks consist of 7 tests that measure whether models contain implicit biases when applied to gendered names, regional names, and some stereotypes.

For example, we found that our models more strongly associate (a) European American names with positive sentiment, when compared to African American names, and (b) negative stereotypes with black women.

These benchmarks are limited in several ways: (a) they may not generalize to your particular use case, and (b) they only test for a very small slice of possible social bias.

These tests are preliminary, and we recommend running tests for your specific use cases. These results should be taken as evidence of the existence of the phenomenon, not a definitive characterization of it for your use case. Please see our usage policies for more details and guidance.

Please contact our support team via chat if you have any questions; we are happy to advise on this.

Blindness to recent events

Limitation: Models lack knowledge of events that occurred after August 2020.

Our models are trained on datasets that contain some information about real world events up until 8/2020. If you rely on the models representing recent events, then they may not perform well.

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:

```
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 second-generation embedding models like `text-embedding-ada-002`, 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.

Vector database options include:

- Pinecone, a fully managed vector database
- Weaviate, an open-source vector search engine
- Redis as a vector database
- Qdrant, a vector search engine
- Milvus, a vector database built for scalable similarity search
- Chroma, an open-source embeddings store
- Typesense, fast open source vector search
- Zilliz, data infrastructure, powered by Milvus

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?

Customers own their input and output from our models, including in the case of embeddings.