Text Embeddings

Model Choices

Voyage currently provides the following text embedding models:

Model	Context Length (tokens)	Embedding Dimension	Description
`voyage-3-large`	32,000	1024 (default), 256, 512, 2048	The best general-purpose and multilingual retrieval quality. See blog post for details.
`voyage-3`	32,000	1024	Optimized for general-purpose and multilingual retrieval quality. See blog post for details.
`voyage-3-lite`	32,000	512	Optimized for latency and cost. See blog post for details.
`voyage-code-3`	32,000	1024 (default), 256, 512, 2048	Optimized for **code** retrieval. See blog post for details.
`voyage-finance-2`	32,000	1024	Optimized for **finance** retrieval and RAG. See blog post for details.
`voyage-law-2`	16,000	1024	Optimized for **legal** retrieval and RAG. Also improved performance across all domains. See blog post for details.

Model	Context Length (tokens)	Embedding Dimension	Description
`voyage-code-2`	16,000	1536	Optimized for code retrieval (17% better than alternatives) / Previous generation of code embeddings. See blog post for details.

Need help deciding which text embedding model to use? Check out our FAQ.

▶ *Older models*

Python API

Voyage text embeddings are accessible in Python through the `voyageai` package. Please install the `voyageai` package, set up the API key, and use the `voyageai.Client.embed()` function to vectorize your inputs.

```
voyageai.Client.embed (texts : List[str], model : str, input_type :
Optional[str] = None, truncation : Optional[bool] = None, output_dimension:
Optional[int] = None, output_dtype: Optional[str] = "float")
```

Parameters

- **texts** (List[str]) - A list of texts as a list of strings, such as `["I like cats", "I also like dogs"]`. Currently, we have two constraints on the list:
  - The maximum length of the list is 128.
  - The total number of tokens in the list is at most 1M for `voyage-3-lite`; 320K for `voyage-3` and `voyage-2`; and 120K for `voyage-3-large`, `voyage-code-3`, `voyage-large-2-instruct`, `voyage-finance-2`, `voyage-multilingual-2`, `voyage-law-2`, and `voyage-large-2`.
- **model** (str) - Name of the model. Recommended options: `voyage-3-large`, `voyage-3`, `voyage-3-lite`, `voyage-code-3`, `voyage-finance-2`, `voyage-law-2`.
- **input_type** (str, optional, defaults to `None`) - Type of the input text. Options: `None`, `query`, `document`.
  - When `input_type` is `None`, the embedding model directly converts the inputs (`texts`) into numerical vectors. For retrieval/search purposes, where a "query" is used to search for relevant information among a collection of data, referred to as "documents", we recommend specifying whether your inputs (`texts`) are intended as queries or documents by setting `input_type` to `query` or `document`, respectively. In these cases, Voyage automatically prepends a prompt to your inputs (`texts`) before vectorizing them, creating vectors more tailored for retrieval/search tasks. Embeddings generated with and without the `input_type` argument are compatible.

    - For transparency, the following prompts are prepended to your input.
        - For query, the prompt is "*Represent the query for retrieving supporting documents:* ".
        - For document, the prompt is "*Represent the document for retrieval:* ".
- **truncation** (bool, optional, defaults to `True`) - Whether to truncate the input texts to fit within the context length.
    - If `True`, an over-length input texts will be truncated to fit within the context length, before vectorized by the embedding model.
    - If `False`, an error will be raised if any given text exceeds the context length.
- **output_dimension** (int, optional, defaults to `None`) - The number of dimensions for resulting output embeddings.
    - Most models only support a single default dimension, used when `output_dimension` is set to `None` (see model embedding dimensions above).
    - `voyage-3-large` and `voyage-code-3` support the following `output_dimension` values: 2048, 1024 (default), 512, and 256.
- **output_dtype** (str, optional, defaults to `float`) - The data type for the embeddings to be returned. Options: `float`, `int8`, `uint8`, `binary`, `ubinary`. `float` is supported for all models. `int8`, `uint8`, `binary`, and `ubinary` are supported by `voyage-3-large` and `voyage-code-3`. Please see our FAQ for more details about output data types.
    - `float`: Each returned embedding is a list of 32-bit (4-byte) single-precision floating-point numbers. This is the default and provides the highest precision / retrieval accuracy.
    - `int8` and `uint8`: Each returned embedding is a list of 8-bit (1-byte) integers ranging from -128 to 127 and 0 to 255, respectively.
    - `binary` and `ubinary`: Each returned embedding is a list of 8-bit integers that represent bit-packed, quantized single-bit embedding values: `int8` for `binary` and `uint8` for `ubinary`. The length of the returned list of integers is 1/8 of `output_dimension` (which is the actual dimension of the embedding). The `binary` type uses the offset binary method. Please refer to our FAQ for details on offset binary and binary embeddings.

Returns

- A `EmbeddingsObject`, containing the following attributes:
    - **embeddings** (List[List[float]] or List[List[int]]) - A list of embeddings for the corresponding list of input texts. Each embedding is a vector represented as a list of floats when `output_dtype` is set to `float` and as a list of integers for all other values of `output_dtype` (`int8`, `uint8`, `binary`, `ubinary`).
    - **total_tokens** (int) - The total number of tokens in the input texts.

Example

Python Output

```
import voyageai
```

```python
vo = voyageai.Client()
# This will automatically use the environment variable VOYAGE_API_KEY.
# Alternatively, you can use vo = voyageai.Client(api_key="<your secret key>")

texts = [
    "The Mediterranean diet emphasizes fish, olive oil, and vegetables, believe
    "Photosynthesis in plants converts light energy into glucose and produces e
    "20th-century innovations, from radios to smartphones, centered on electron
    "Rivers provide water, irrigation, and habitat for aquatic species, vital f
    "Apple's conference call to discuss fourth fiscal quarter results and busin
    "Shakespeare's works, like 'Hamlet' and 'A Midsummer Night's Dream,' endure
]

# Embed the documents
result = vo.embed(texts, model="voyage-3", input_type="document")
print(result.embeddings)
```

▶ *Deprecated Functions*

REST API

Voyage text embeddings can be accessed by calling the endpoint `POST https://api.voyageai.com/v1/embeddings` . Please refer to the Text Embeddings API Reference for the specification.

Example

Embed a single string Embed a list of strings

```bash
curl https://api.voyageai.com/v1/embeddings \
  -H "Content-Type: application/json" \
  -H "Authorization: Bearer $VOYAGE_API_KEY" \
  -d '{
    "input": "Sample text",
    "model": "voyage-3",
    "input_type": "document"
  }'
```

TypeScript Library

Voyage text embeddings are accessible in TypeScript through the Voyage TypeScript Library, which exposes all the functionality of our text embeddings endpoint (see Text Embeddings API Reference).

Updated 5 days ago