Basilica Documentation

Release 0.2.7

Michael Lucy, Jorge Silva

Dec 05, 2019

Contents

CHAPTER 1

Quickstart

1.1 Install the python client

First, install the Python client.

```
$ pip install basilica
```

1.2 Embed some sentences

Let's embed some sentences to make sure the client is working.

```
import basilica
sentences = [
    "This is a sentence!",
    "This is a similar sentence!",
    "I don't think this sentence is very similar at all...",
]
with basilica.Connection('SLOW_DEMO_KEY') as c:
    embeddings = list(c.embed_sentences(sentences))
print(embeddings)
```

```
[[0.8556405305862427, ...], ...]
```

Let's also make sure these embeddings make sense, by checking that the cosine distance between the two similar sentences is smaller:

```
from scipy import spatial
print(spatial.distance.cosine(embeddings[0], embeddings[1]))
print(spatial.distance.cosine(embeddings[0], embeddings[2]))
```

```
0.024854343247535327
0.25084750542635814
```

Great!

1.3 Get an API key

The example above uses the slow demo key. You can get an API key of your own by signing up at https://www.basilica.ai/accounts/register . (If you already have an account, you can view your API keys at https://www.basilica.ai/api-keys .)

1.4 What next?

- Read the documentation for the python client: *Basilica Python Client*
- See an in-depth tutorial on training an image model: How To Train An Image Model With Basilica

CHAPTER 2

Basilica Python Client

class basilica.**Connection**(*auth_key*, *server='https://api.basilica.ai'*, *retries=2*, *backoff_factor=0.1*, *status_forcelist=500*)

A connection to basilica.ai that can be used to generate embeddings.

Parameters

- **auth_key** (*str*) – Your auth key. You can view your auth keys at https://basilica.ai/api-keys/.
- **server** (*str*) – What URL to use to connect to the server.
- **retries** (*int*) – Number of times to retry failed connections and requests.
- **backoff_factor** (*float*) – See urllib3.util.retry.Retry.backoff_factor .
- **status_forcelist** (*Tuple[int]*) – What HTTP response codes trigger a retry.

```
>>> with basilica.Connection('SLOW_DEMO_KEY') as c:
...   print(c.embed_sentence('A sentence.'))
[0.6246702671051025, ..., -0.03025037609040737]
```

embed_image(*image*, *model='generic'*, *version='default'*, *opts={}*, *timeout=10*)

Generate the embedding for a JPEG image. The image should be passed as a byte string.

Parameters

- **image** (*str*) – The image to embed.
- **model** (*str*) – What model to use (i.e. the kind of image being embedded).
- **version** (*str*) – What version of that model to use.
- **opts** (*Dict[str, Any]*) – Options specific to the model/version you chose.
- **opts["dimensions"]** (*int*) – Number of dimensions to return. PCA will be used to reduce the number of dimensions with minimal information loss.

- **opts["normalize_l2"]** (*bool*) – Whether or not each instance should be scaled to have unit L2 norm. (This is sometimes useful for instance retrieval tasks.) Defaults to False.
- **opts["normalize_mean"]** (*bool*) – Whether or not to normalize each feature in the embedding to have mean 0 across our sample dataset. Defaults to True when *dimensions* is set, or False otherwise.
- **opts["normalize_variance"]** (*bool*) – Whether or not to normalize each feature in the embedding to have unit variance across our sample dataset. Defaults to True when *dimensions* is set, or False otherwise.
- **timeout** (*int*) – HTTP timeout for request.

Returns An embedding.

Return type List[float]

```
>>> with basilica.Connection('SLOW_DEMO_KEY') as c:
...   with open('img.jpg', 'rb') as f:
...     print(c.embed_image(f.read()))
[0.6246702671051025, ...]
```

embed_image_file(*image_file*, *model='generic'*, *version='default'*, *opts={}*, *timeout=10*)

Generate the embedding for a JPEG image file. The file name should be passed as a path that can be understood by *open*.

Parameters

- **image_file** (*str*) – Path to the image to embed.
- **model** (*str*) – What model to use (i.e. the kind of image being embedded).
- **version** (*str*) – What version of that model to use.
- **opts** (*Dict[str, Any]*) – Options specific to the model/version you chose.
- **opts["dimensions"]** (*int*) – Number of dimensions to return. PCA will be used to reduce the number of dimensions with minimal information loss.
- **opts["normalize_l2"]** (*bool*) – Whether or not each instance should be scaled to have unit L2 norm. (This is sometimes useful for instance retrieval tasks.) Defaults to False.
- **opts["normalize_mean"]** (*bool*) – Whether or not to normalize each feature in the embedding to have mean 0 across our sample dataset. Defaults to True when *dimensions* is set, or False otherwise.
- **opts["normalize_variance"]** (*bool*) – Whether or not to normalize each feature in the embedding to have unit variance across our sample dataset. Defaults to True when *dimensions* is set, or False otherwise.
- **timeout** (*int*) – HTTP timeout for request.

Returns An embedding.

Return type List[float]

```
>>> with basilica.Connection('SLOW_DEMO_KEY') as c:
...   print(c.embed_image_file('img.jpg')
[0.6246702671051025, ...]
```

embed_image_files(*image_files*, *model='generic'*, *version='default'*, *batch_size=32*, *opts={}*, *timeout=30*)

Generate embeddings for JPEG image files. The file names should be passed as paths that can be understood by *open*.

Parameters

- **image_files** (*Iterable[str]*) – An iterable (such as a list) of paths to the images to embed.
- **model** (*str*) – What model to use (i.e. the kind of image being embedded).
- **version** (*str*) – What version of that model to use.
- **batch_size** (*int*) – How many instances to send to the server at a time.
- **opts** (*Dict[str, Any]*) – Options specific to the model/version you chose.
- **opts["dimensions"]** (*int*) – Number of dimensions to return. PCA will be used to reduce the number of dimensions with minimal information loss.
- **opts["normalize_l2"]** (*bool*) – Whether or not each instance should be scaled to have unit L2 norm. (This is sometimes useful for instance retrieval tasks.) Defaults to False.
- **opts["normalize_mean"]** (*bool*) – Whether or not to normalize each feature in the embedding to have mean 0 across our sample dataset. Defaults to True when *dimensions* is set, or False otherwise.
- **opts["normalize_variance"]** (*bool*) – Whether or not to normalize each feature in the embedding to have unit variance across our sample dataset. Defaults to True when *dimensions* is set, or False otherwise.
- **timeout** (*int*) – HTTP timeout for request.

Returns A generator of embeddings.

Return type Generator[List[float]]

```
>>> with basilica.Connection('SLOW_DEMO_KEY') as c:
...   for embedding in c.embed_image_files(['img1.jpg', 'img2.jpg']):
...     print(embedding)
[0.6246702671051025, ...]
[-0.03025037609040737, ...]
```

embed_images(*images*, *model='generic'*, *version='default'*, *batch_size=32*, *opts={}*, *timeout=30*)

Generate embeddings for JPEG images. Images should be passed as byte strings, and will be sent to the server in batches to be embedded.

Parameters

- **images** (*Iterable[str]*) – An iterable (such as a list) of the images to embed.
- **model** (*str*) – What model to use (i.e. the kind of image being embedded).
- **version** (*str*) – What version of that model to use.
- **batch_size** (*int*) – How many instances to send to the server at a time.
- **opts** (*Dict[str, Any]*) – Options specific to the model/version you chose.
- **opts["dimensions"]** (*int*) – Number of dimensions to return. PCA will be used to reduce the number of dimensions with minimal information loss.

- **opts["normalize_l2"]** (*bool*) – Whether or not each instance should be scaled to have unit L2 norm. (This is sometimes useful for instance retrieval tasks.) Defaults to False.
- **opts["normalize_mean"]** (*bool*) – Whether or not to normalize each feature in the embedding to have mean 0 across our sample dataset. Defaults to True when *dimensions* is set, or False otherwise.
- **opts["normalize_variance"]** (*bool*) – Whether or not to normalize each feature in the embedding to have unit variance across our sample dataset. Defaults to True when *dimensions* is set, or False otherwise.
- **timeout** (*int*) – HTTP timeout for request.

Returns A generator of embeddings.

Return type Generator[List[float]]

```
>>> with basilica.Connection('SLOW_DEMO_KEY') as c:
...   images = []
...   for filename in ['img1.jpg', 'img2.jpg']:
...     with open(filename, 'rb') as f:
...     images.append(f.read())
...   for embedding in c.embed_images(images):
...     print(embedding)
[0.6246702671051025, ...]
[-0.03025037609040737, ...]
```

embed_sentence(*sentence*, *model='english'*, *version='default'*, *opts={}*, *timeout=5*)

Generate the embedding for a sentence.

Parameters

- **sentence** (*str*) – The sentence to embed.
- **model** (*str*) – What model to use (i.e. the kind of sentence being embedded).
  - **generic:** Generic English text embedding (the default.)
  - **reddit:** Text embedding specialized for English Reddit posts.
  - **twitter:** Text embedding specialized for English tweets.
  - **email:** Text embedding specialized for English emails.
  - **product-reviews:** Text embedding specialized for English product reviews.
- **version** (*str*) – What version of that model to use.
- **opts** (*Dict[str, Any]*) – Options specific to the model/version you chose.
- **opts["dimensions"]** (*int*) – Number of dimensions to return. PCA will be used to reduce the number of dimensions with minimal information loss.
- **opts["normalize_l2"]** (*bool*) – Whether or not each instance should be scaled to have unit L2 norm. (This is sometimes useful for instance retrieval tasks.) Defaults to False.
- **opts["normalize_mean"]** (*bool*) – Whether or not to normalize each feature in the embedding to have mean 0 across our sample dataset. Defaults to True when *dimensions* is set, or False otherwise.

- **opts["normalize_variance"]** (*bool*) – Whether or not to normalize each feature in the embedding to have unit variance across our sample dataset. Defaults to True when *dimensions* is set, or False otherwise.
- **timeout** (*int*) – HTTP timeout for request.

Returns An embedding.

Return type List[float]

```
>>> with basilica.Connection('SLOW_DEMO_KEY') as c:
...   print(c.embed_sentence('This is a sentence.')
[0.6246702671051025, ...]
```

embed_sentences(*sentences*, *model='english'*, *version='default'*, *batch_size=64*, *opts={}*, *timeout=15*)

Generate embeddings for sentences.

Parameters

- **sentences** (*Iterable[str]*) – An iterable (such as a list) of sentences to embed.
- **model** (*str*) – What model to use (i.e. the kind of sentence being embedded).
  - **generic:** Generic English text embedding (the default.)
  - **reddit:** Text embedding specialized for English Reddit posts.
  - **twitter:** Text embedding specialized for English tweets.
  - **email:** Text embedding specialized for English emails.
  - **product-reviews:** Text embedding specialized for English product reviews.
- **version** (*str*) – What version of that model to use.
- **batch_size** (*int*) – How many instances to send to the server at a time.
- **opts** (*Dict[str, Any]*) – Options specific to the model/version you chose.
- **opts["dimensions"]** (*int*) – Number of dimensions to return. PCA will be used to reduce the number of dimensions with minimal information loss.
- **opts["normalize_l2"]** (*bool*) – Whether or not each instance should be scaled to have unit L2 norm. (This is sometimes useful for instance retrieval tasks.) Defaults to False.
- **opts["normalize_mean"]** (*bool*) – Whether or not to normalize each feature in the embedding to have mean 0 across our sample dataset. Defaults to True when *dimensions* is set, or False otherwise.
- **opts["normalize_variance"]** (*bool*) – Whether or not to normalize each feature in the embedding to have unit variance across our sample dataset. Defaults to True when *dimensions* is set, or False otherwise.
- **timeout** (*int*) – HTTP timeout for request.

Returns A generator of embeddings.

Return type Generator[List[float]]

```
>>> with basilica.Connection('SLOW_DEMO_KEY') as c:
...   for embedding in c.embed_sentences(['Sentence one.', 'Sentence two.']):
...     print(embedding)
```

(continues on next page)

(continued from previous page)

```
[0.6246702671051025, ...]
[-0.03025037609040737, ...]
```

Index

C

E