Caching Embeddings

Embeddings can be stored or temporarily cached to avoid needing to recompute them.

Caching embeddings can be done using a `CacheBackedEmbeddings`.

The cache backed embedder is a wrapper around an embedder that caches embeddings in a key-value store.

The text is hashed and the hash is used as the key in the cache.

The main supported way to initialized a `CacheBackedEmbeddings` is `from_bytes_store`. This takes in the following parameters:

- underlying_embedder: The embedder to use for embedding.
- document_embedding_cache: The cache to use for storing document embeddings.
- namespace: (optional, defaults to `""`) The namespace to use for document cache. This namespace is used to avoid collisions with other caches. For example, set it to the name of the embedding model used.

Attention: Be sure to set the `namespace` parameter to avoid collisions of the same text embedded using different embeddings models.

```
from langchain.storage import InMemoryStore, LocalFileStore, RedisStore
from langchain.embeddings import OpenAIEmbeddings,
CacheBackedEmbeddings
```

API Reference:

- InMemoryStore from `langchain.storage`
- LocalFileStore from `langchain.storage`
- RedisStore from `langchain.storage`
- OpenAIEmbeddings from `langchain.embeddings`
- CacheBackedEmbeddings from `langchain.embeddings`

Using with a vectorstore

First, let's see an example that uses the local file system for storing embeddings and uses FAISS vectorstore for retrieval.

```
from langchain.document_loaders import TextLoader
from langchain.embeddings.openai import OpenAIEmbeddings
from langchain.text_splitter import CharacterTextSplitter
from langchain.vectorstores import FAISS
```

API Reference:

- TextLoader from `langchain.document_loaders`
- OpenAIEmbeddings from `langchain.embeddings.openai`
- CharacterTextSplitter from `langchain.text_splitter`
- FAISS from `langchain.vectorstores`

```
underlying_embeddings = OpenAIEmbeddings()
```

```
fs = LocalFileStore("./cache/")

cached_embedder = CacheBackedEmbeddings.from_bytes_store(
    underlying_embeddings, fs, namespace=underlying_embeddings.model
)
```

The cache is empty prior to embedding

```
list(fs.yield_keys())
```

```
    []
```

Load the document, split it into chunks, embed each chunk and load it into the vector store.

```
raw_documents = TextLoader("../state_of_the_union.txt").load()
text_splitter = CharacterTextSplitter(chunk_size=1000, chunk_overlap=0)
documents = text_splitter.split_documents(raw_documents)
```

create the vectorstore

```
db = FAISS.from_documents(documents, cached_embedder)
```

```
    CPU times: user 608 ms, sys: 58.9 ms, total: 667 ms
    Wall time: 1.3 s
```

If we try to create the vectostore again, it'll be much faster since it does not need to re-compute any embeddings.

```
db2 = FAISS.from_documents(documents, cached_embedder)
```

```
    CPU times: user 33.6 ms, sys: 3.96 ms, total: 37.6 ms
    Wall time: 36.8 ms
```

And here are some of the embeddings that got created:

```
list(fs.yield_keys())[:5]
```

```
    ['text-embedding-ada-002614d7cf6-46f1-52fa-9d3a-740c39e7a20e',
     'text-embedding-ada-0020fc1ede2-407a-5e14-8f8f-5642214263f5',
     'text-embedding-ada-002e4ad20ef-dfaa-5916-9459-f90c6d8e8159',
     'text-embedding-ada-002a5ef11e4-0474-5725-8d80-81c91943b37f',
     'text-embedding-ada-00281426526-23fe-58be-9e84-6c7c72c8ca9a']
```

In Memory

This section shows how to set up an in memory cache for embeddings. This type of cache is primarily useful for unit tests or prototyping. Do **not** use this cache if you need to actually store the embeddings.

```
store = InMemoryStore()
```

```
underlying_embeddings = OpenAIEmbeddings()
embedder = CacheBackedEmbeddings.from_bytes_store(
    underlying_embeddings, store, namespace=underlying_embeddings.model
)
```

```
embeddings = embedder.embed_documents(["hello", "goodbye"])
```

```
    CPU times: user 10.9 ms, sys: 916 µs, total: 11.8 ms
    Wall time: 159 ms
```

The second time we try to embed the embedding time is only 2 ms because the embeddings are looked up in the cache.

```
embeddings_from_cache = embedder.embed_documents(["hello", "goodbye"])
```

```
    CPU times: user 1.67 ms, sys: 342 µs, total: 2.01 ms
    Wall time: 2.01 ms
```

```
embeddings == embeddings_from_cache
```

```
    True
```

File system

This section covers how to use a file system store.

```
fs = LocalFileStore("./test_cache/")
```

```
embedder2 = CacheBackedEmbeddings.from_bytes_store(
    underlying_embeddings, fs, namespace=underlying_embeddings.model
)
```

```
embeddings = embedder2.embed_documents(["hello", "goodbye"])
```

```
    CPU times: user 6.89 ms, sys: 4.89 ms, total: 11.8 ms
    Wall time: 184 ms
```

```
embeddings = embedder2.embed_documents(["hello", "goodbye"])
```

```
    CPU times: user 0 ns, sys: 3.24 ms, total: 3.24 ms
    Wall time: 2.84 ms
```

Here are the embeddings that have been persisted to the directory `./test_cache`.

Notice that the embedder takes a namespace parameter.

```
list(fs.yield_keys())
```

```
    ['text-embedding-ada-002e885db5b-c0bd-5fbc-88b1-4d1da6020aa5',
     'text-embedding-ada-0026ba52e44-59c9-5cc9-a084-284061b13c80']
```

Redis Store

```
from langchain.storage import RedisStore
```

API Reference:

- RedisStore from `langchain.storage`

```
# For cache isolation can use a separate DB
# Or additional namepace
store = RedisStore(redis_url="redis://localhost:6379", client_kwargs=
{'db': 2}, namespace='embedding_caches')

underlying_embeddings = OpenAIEmbeddings()
embedder = CacheBackedEmbeddings.from_bytes_store(
    underlying_embeddings, store, namespace=underlying_embeddings.model
)
```

```
embeddings = embedder.embed_documents(["hello", "goodbye"])
```

```
    CPU times: user 3.99 ms, sys: 0 ns, total: 3.99 ms
    Wall time: 3.5 ms
```

```
embeddings = embedder.embed_documents(["hello", "goodbye"])
```

```
    CPU times: user 2.47 ms, sys: 767 µs, total: 3.24 ms
    Wall time: 2.75 ms
```

```
list(store.yield_keys())
```

```
    ['text-embedding-ada-002e885db5b-c0bd-5fbc-88b1-4d1da6020aa5',
     'text-embedding-ada-0026ba52e44-59c9-5cc9-a084-284061b13c80']
```

```
list(store.client.scan_iter())
```

```
    [b'embedding_caches/text-embedding-ada-002e885db5b-c0bd-5fbc-88b1-
4d1da6020aa5',
     b'embedding_caches/text-embedding-ada-0026ba52e44-59c9-5cc9-a084-
284061b13c80']
```