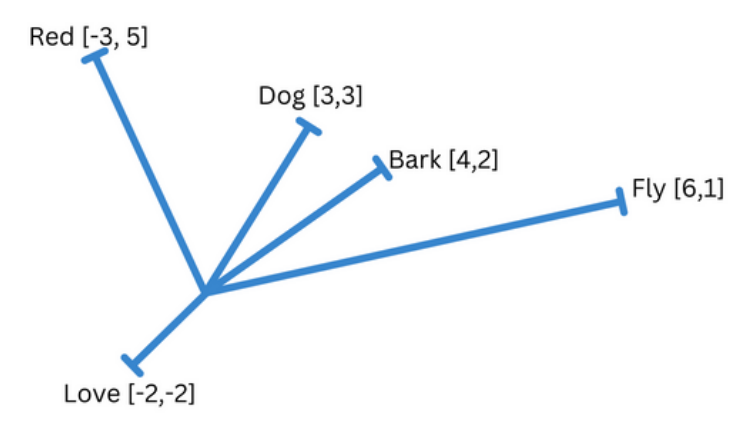
What are Embeddings?

All Machine Learning/AI models work with numerical data. Before the performance of any operation all text/image/audio/video data has to be transformed into a numerical representation. Embeddings are vector representations of data that capture meaningful relationships between entities. As a general definition, embeddings are data that has been transformed into n-dimensional matrices for use in deep learning computations. A word embedding is a vector representation of words.



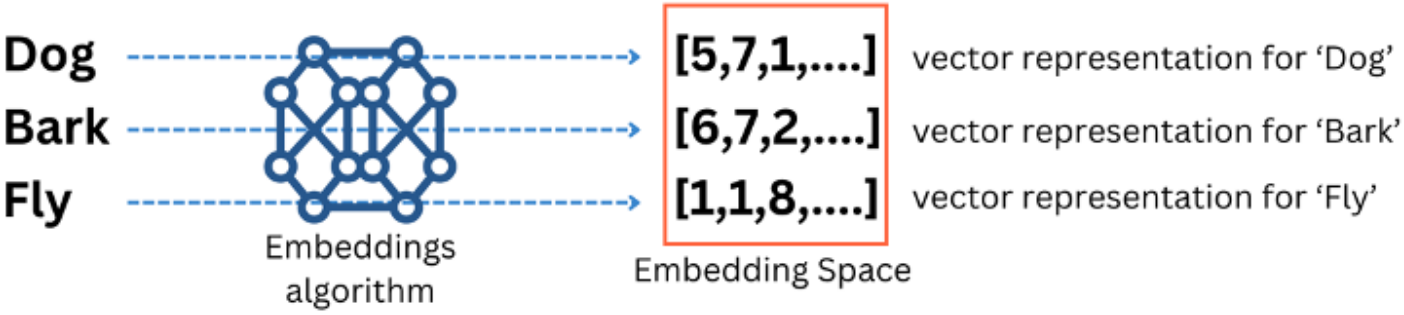
Words in a two-dimensional vector space

What are Vectors?

In computer science and machine learning, the idea of a vector is an abstract representation of data, and the representation is an array or list of numbers. These numbers represent the features or attribute of the data. In NLP, a vector can represent a document, a sentence or even a word. The length of the array or list is the number of dimensions in the vector

Words in a unidimensional vector

The process of embedding **transforms** data (like text) into vectors, **compresses** the input information resulting in an **embedding space** *specific to the training data*

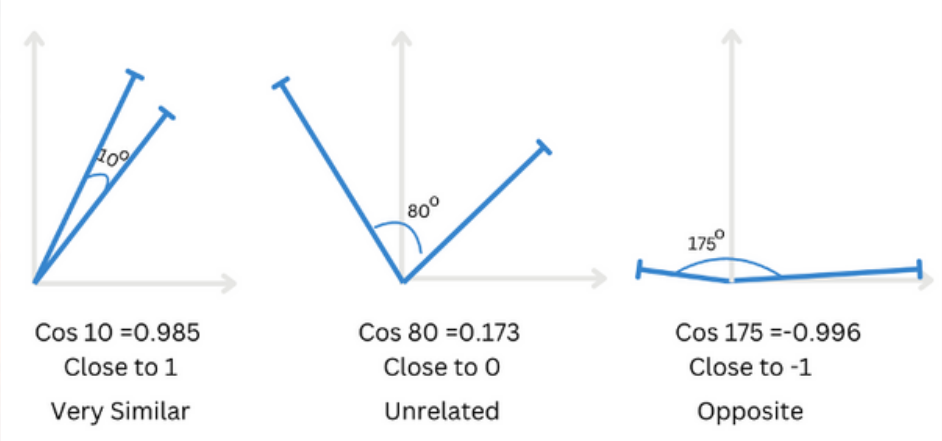


Embeddings in RAG

The reason why embeddings are popular is because they help in **establishing semantic relationship between words, phrases, and documents**. In the simplest methods of searching or text matching, we use keywords and if the keywords match, we can show the matching documents as results of the search. However, this approach fails to consider the semantic relationships or the meanings of the words while searching. This challenge is overcome by using embeddings.

In the retrieval step, the user query is matched with relevant documents based on how similar the documents are with the query

How is similarity calculated using embeddings?



Cosine similarity of vectors in two-dimensional vector space is the cosine of the angle between them

Closeness to each other is calculated by the distance between the points in the vector space. One of the most common measures of similarity is Cosine Similarity. Cosine similarity is calculated as the cosine value of the angle between the two vectors. Recall from trigonometry that cosine of parallel lines i.e. angle=0o is 1 and cosine of a right angle i.e. 90o is 0. On the other end, the cosine of opposite lines i.e. angle =180o is -1. Therefore, the cosine similarity lies between -1 and 1 where unrelated terms have a value close to 0, and related terms have a value close to 1. Terms that are opposite in meaning have a value of -1.

Popular Embedding Algorithms

word2vec Google's Word2Vec is one of the most popular pre-trained word embeddings. The official paper - https://arxiv.org/pdf/1301.3781.pdf

GLOVE The 'Global Vectors' model is so termed because it captures statistics directly at a global level. The official paper - https://nlp.stanford.edu/pubs/glove.pdf

fastText Facebook's AI research, fastText builds embeddings composed of characters instead of words. The official paper - https://arxiv.org/pdf/1607.04606.pdf

Elmo Embeddings from Language Models, are learnt from the internal state of a bidirectional LSTM. The official paper - https://arxiv.org/pdf/1802.05365.pdf

BERT Bidirectional Encoder Representations from Transformers is a transformer bases approach. The official paper - https://arxiv.org/pdf/1810.04805.pdf

Common Pre-trained Embeddings Models

The good news for anyone building RAG enabled systems is that embeddings once created can also generalize across tasks and domains. There are a variety of proprietary and open-source pre-trained embeddings models that are available to use. This is also one of the reasons why the usage of embeddings has exploded in popularity across machine learning applications.

OpenAI

text-embedding-ada-002 was released in December 2022. It has a dimension of 1536 meaning that it converts text into a vector of 1536 dimensions.

text-embedding-3-small is the latest small embedding model of 1536 dimensions released in January 2024. The flexibility it provides over ada-002 model is that users can adjust the size of the dimensions according to their needs.

a.text-embedding-3-large is a large embedding model of 3072 dimensions released together with the text-embedding-3-small model. It is the best performing model released by OpenAI yet.

Common Pre-trained Embeddings Models

Gemini

text-embedding-004 (last updated in April 2024) is the model offered by Google Gemini. It offers elastic embeddings size up to 768 dimensions and can be accessed via the Gemini API

VOYAGE AI

Voyage AI embeddings models are recommended by Anthropic, the providers of Claude series of Large Language Models. Voyage offers several embeddings models like –
voyage-large-2-instruct is a 1024-dimension embeddings model that has become a leader in embeddings models.
voyage-law-2 is a 1024-dimension model that has been optimized for legal documents.
voyage-code-2 is a 1536-dimension model that has been optimized for code retrieval.
voyage-large-2 is a 1536-dimension general purpose model optimized for retrieval.

MISTRAL AI_

Mistral is the company behind LLMs like Mistral and Mixtral. They offer a 1024-dimension embeddings model by the name of **mistral-embed**. This is an open-source embeddings model.

cohere

Cohere, the developers of Command, Command R and Command R+ LLMs also offer a variety of embeddings models. Some of these are-
embed-english-v3.0 is a 1024-dimension model that works on embeddings for English only.
embed-english-light-v3.0 is a lighter version of embed-english model that has 384 dimensions.
embed-multilingual-v3.0 offers multilingual support for over 100 languages.

These five models are in no way recommendations but just a list of the popular embeddings models. Apart from these providers, almost all LLM developers like Meta, TII, LMSYS also offer pre-trained embeddings models. One place to check out all the popular embeddings models is the MTEB (Massive Text Embedding Benchmark) Leaderboard on HuggingFace (https://huggingface.co/spaces/mteb/leaderboard). The MTEB benchmark compares the embeddings models on tasks like classification, retrieval, clustering and more.

Want to know more about RAG?

New in MEAP

A SIMPLE GUIDE TO

Retrieval Augmented Generation

Abhinav Kimothi

****Subscribe Now****

Early Access to Chapter 1-3

Raw & Unedited

Ch 1 : LLMs & the need for RAG
Ch 2 : RAG enabled systems & their design
Ch 3 : Indexing Pipeline - Creating a knowledge base for RAG based applications

Launch Offer : Use Code **mlkimothi** for 50% discount

valid till July 4th, 2024

How to Choose Embeddings?

Ever since the release of ChatGPT and the advent of the aptly described LLM Wars, there has also been a mad rush in developing embeddings models. There are many evolving standards of evaluating LLMs and embeddings alike.

When building RAG powered LLM apps, there is no right answer to "Which embeddings model to use?". However, you may notice particular embeddings working better for specific use cases (like summarization, text generations, classification etc.)

	Models	Use Cases
Text similarity: Captures semantic similarity between pieces of text.	text-similarity-{ada, babbage, curie, davinci}-001	Clustering, regression, anomaly detection, visualization
Text search: Semantic information retrieval over documents.	text-search-{ada, babbage, curie, davinci}-{query, doc}-001	Search, context relevance, information retrieval
Code search: Find relevant code with a query in natural language.	code-search-{ada, babbage}-{code, text}-001	Code search and relevance

OpenAI used to recommend different embeddings models for different use cases. However, now they recommend **ada v2** for all tasks.

Average (56 datasets)	Classification Average (12 datasets)	Clustering Average (11 datasets)	Pair Classification Average (3 datasets)	Reranking Average (4 datasets)	Retrieval Average (15 datasets)	STS Average (10 datasets)	Summarization Average (1 dataset)

MTEB Leaderboard at Hugging Face evaluates almost all available embedding models across seven use cases - *Classification*, *Clustering*, *Pair Classification*, *Reranking*, *Retrieval*, *Semantic Textual Similarity (STS)* and *Summarization*.

Another important consideration is **cost**. With OpenAI models you can incur significant costs if you are working with a lot of documents. The cost of open source models will depend on the implementation.

Creating Embeddings

Once you've chosen your embedding model, there are several ways of creating the embeddings. Sometimes, our friends, LlamaIndex and LangChain come in pretty handy to convert documents (*split into chunks*) into vector embeddings. Other times you can use the service from a provider directly or get the embeddings from HuggingFace

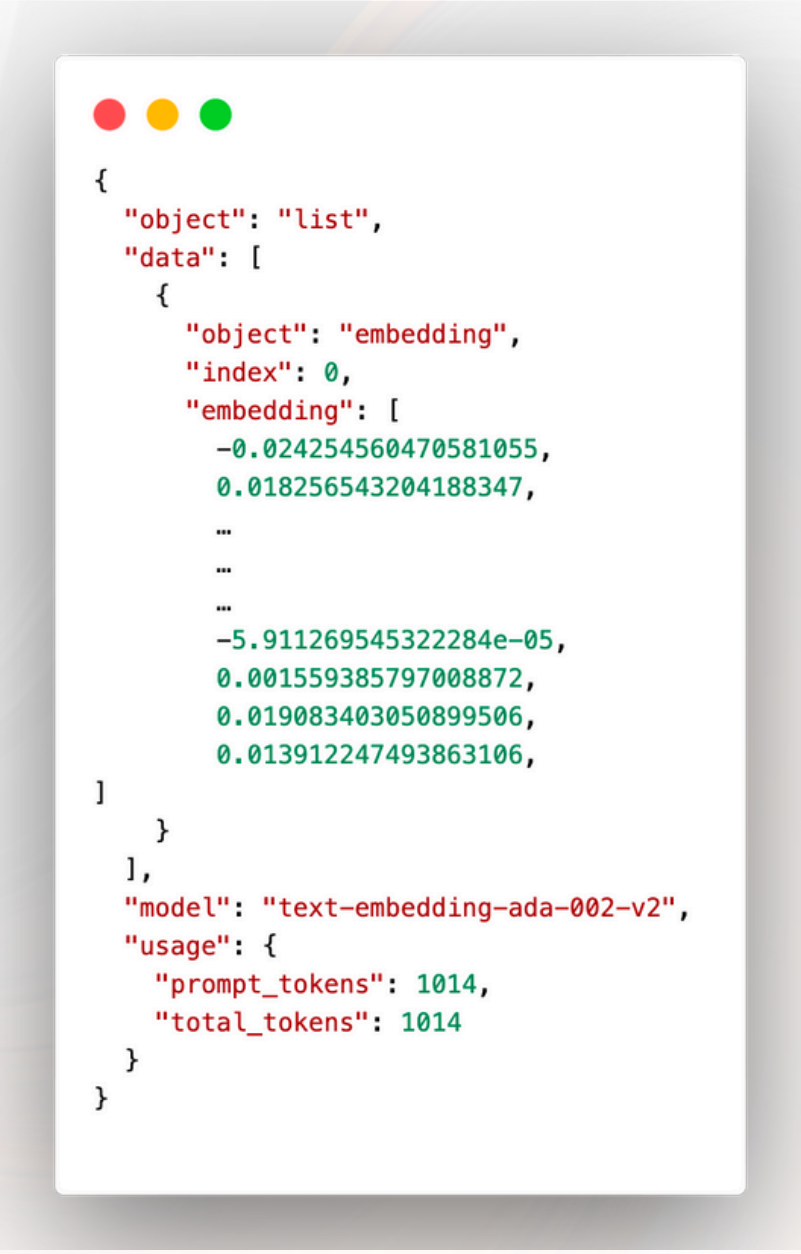
Example : OpenAI text-embedding-ada-002

using Embedding.create() function from openai library

```
with open('../Data/AK_BusyPersonIntroLLM.txt') as f:
    IntroToLLM = f.read()

from langchain.text_splitter import TokenTextSplitter

text_splitter = TokenTextSplitter(
    chunk_size = 1000,
    chunk_overlap  = 20,
    length_function = len,

)

texts = text_splitter.create_documents([IntroToLLM])

import openai
openai.api_key='sk-#####eGeY##########RZj'

response = openai.Embedding.create(
    input=texts[0].page_content,
    model="text-embedding-ada-002"
)

print(response)
```

You'll need an OpenAI apikey to create these embeddings
You can get one here - https://platform.openai.com/api-keys

Example Response



```
{
  "object": "list",
  "data": [
    {
      "object": "embedding",
      "index": 0,
      "embedding": [
        -0.024254560470581055,
        0.018256543204188347,
        …
        …
        …
        -5.911269545322284e-05,
        0.001559385797008872,
        0.019083403050899506,
        0.013912247493863106,
]
    }
  ],
  "model": "text-embedding-ada-002-v2",
  "usage": {
    "prompt_tokens": 1014,
    "total_tokens": 1014
  }
}
```

response.data[0].embedding will give the created embeddings that can be stored for retrieval

Cost

Model	Usage
ada v2	$0.0001 / 1K tokens

In this example, 1014 tokens will cost about $.0001. Recall that for this youtube transcript we got 14 chunks. So creating the embeddings for the entire transcript will cost about 0.14 cents. This may seem low, but when you scale up to thousands of documents being updated frequently, the cost can become a concern.

Example : msmarco-bert-base-dot-v5

using HuggingFaceEmbeddings from langchain.embeddings

```
from langchain.embeddings import HuggingFaceEmbeddings

embeddings = HuggingFaceEmbeddings(
    model_name='sentence-transformers/msmarco-bert-base-dot-v5'
    )
text = texts[0].page_content

query_result = embeddings.embed_query(text)
```

Example : embed-english-light-v3.0

using CohereEmbeddings from langchain.embeddings

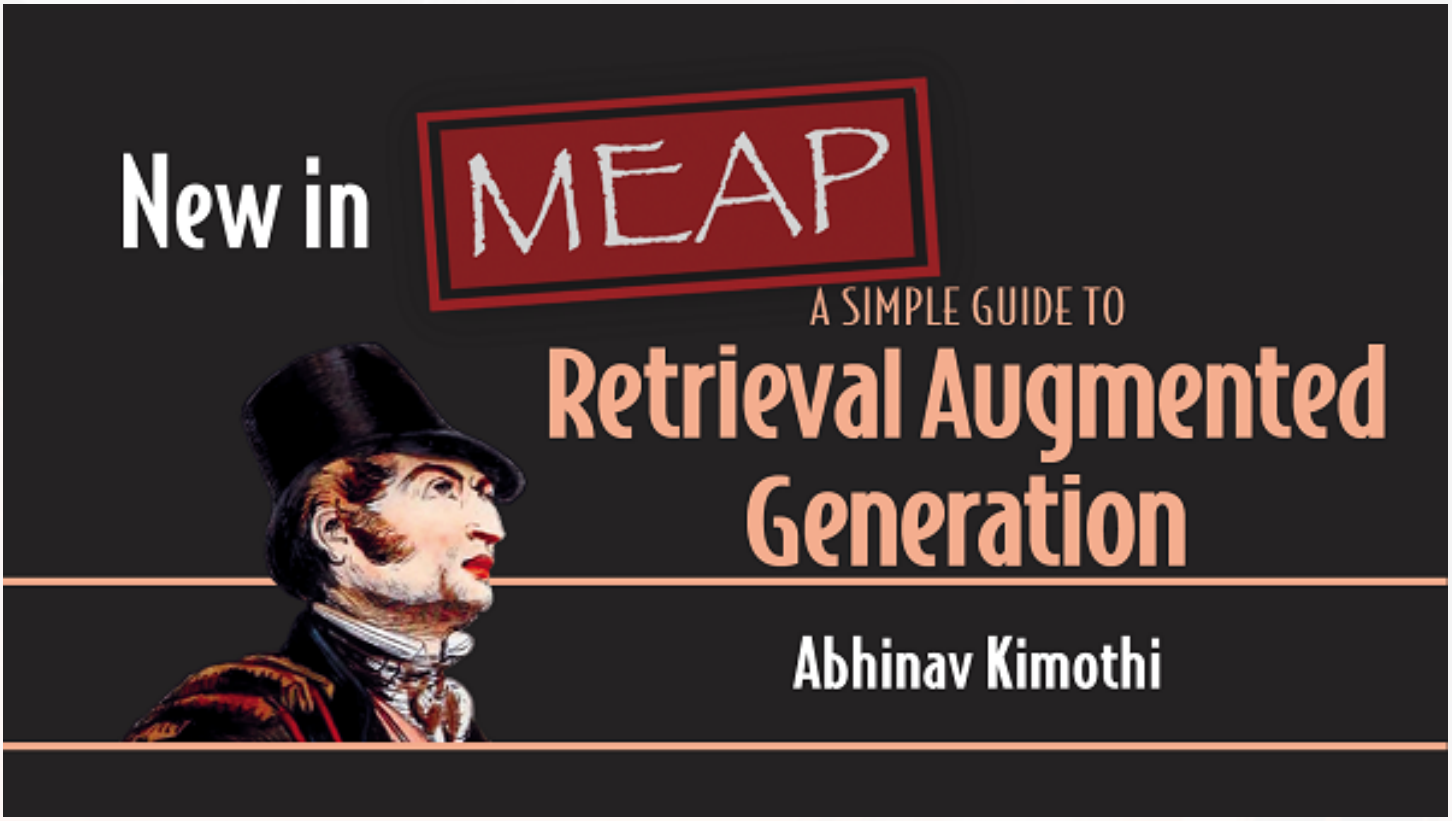
```
from langchain.embeddings import CohereEmbeddings

embeddings = CohereEmbeddings(
    model="embed-english-light-v3.0")

text = texts[0].page_content

query_result = embeddings.embed_query(text)
```

LangChain

All the available embeddings classes on LangChain

Hello!
I'm Abhinav...
A data science and AI professional with over 15 years in the industry. Passionate about AI advancements, I constantly explore emerging technologies to push the boundaries and create positive impacts in the world.

A Simple Guide to Retrieval Augmented Generation is now available for Early Access



****Subscribe Now****

Avail Early Access Discounts to Chapter 1-3

Raw & Unedited

Ch 1 : LLMs & the need for RAG
Ch 2 : RAG enabled systems & their design
Ch 3 : Indexing Pipeline - Creating a knowledge base for RAG based applications

Complete book coming soon

Launch Offer : Use Code **mlkimothi** for 50% discount
valid till July 4th, 2024