Member-only story

Embedding: Types, Use cases and Evaluation (Part 3 of RAG Series)

Making computers understand Text

Chandan Durgia · Follow

7 min read · Feb 1, 2024

--

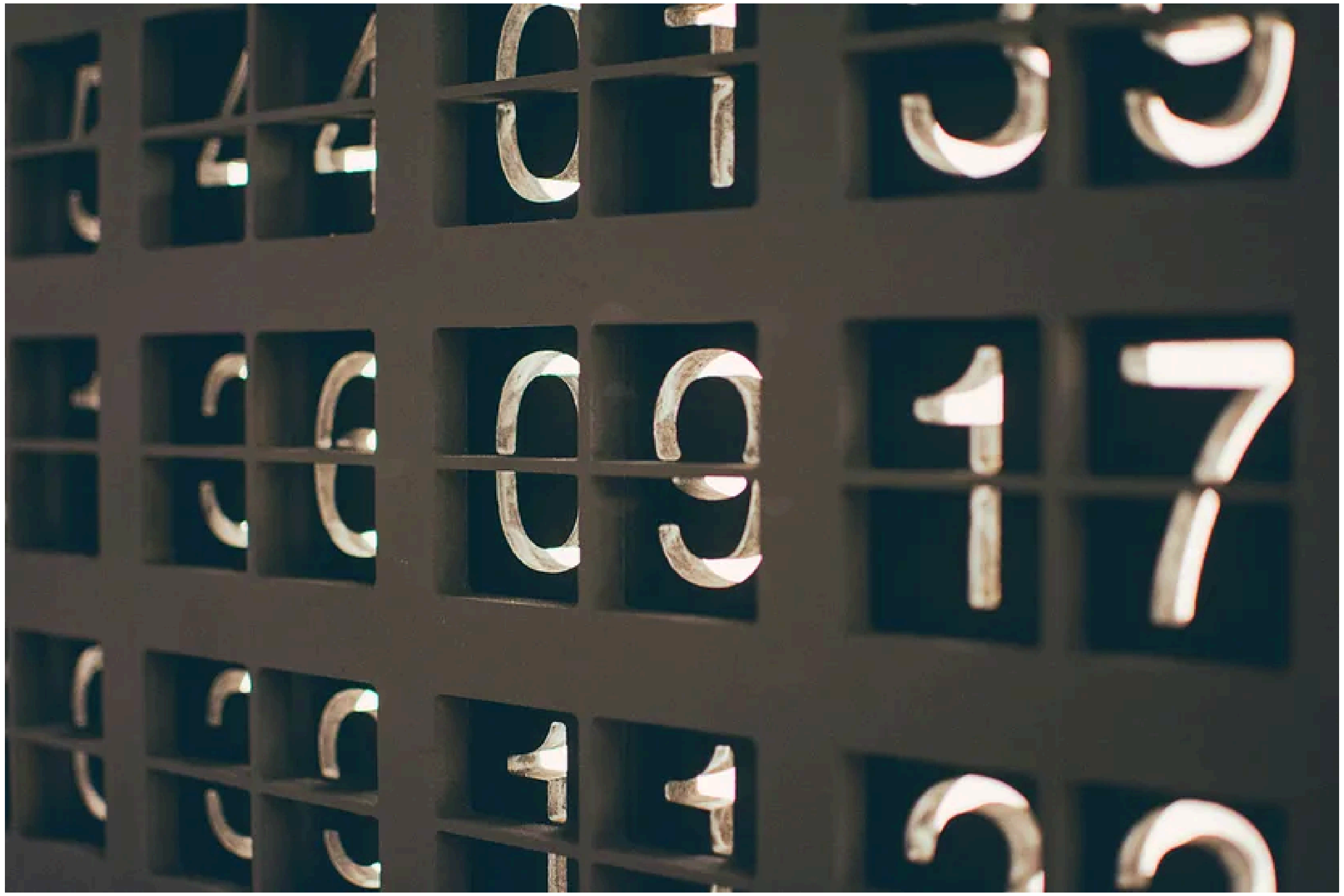


Photo by Nick Hillier on Unsplash

This is part 3 of the "Retrieval-Augmented Generation (RAG) — Basics to Advanced Series". Links to other blogs in the series are at the bottom of this blog. Taking forward from part 1 (RAG Basics) and part 2 (Chunking), in this blog we will focus on the "Embedding" component which is relevant for embedding of chunks in the source content and the query. (highlighted in

Blue). Since, fundamentally the concept is similar, we will cover this together.

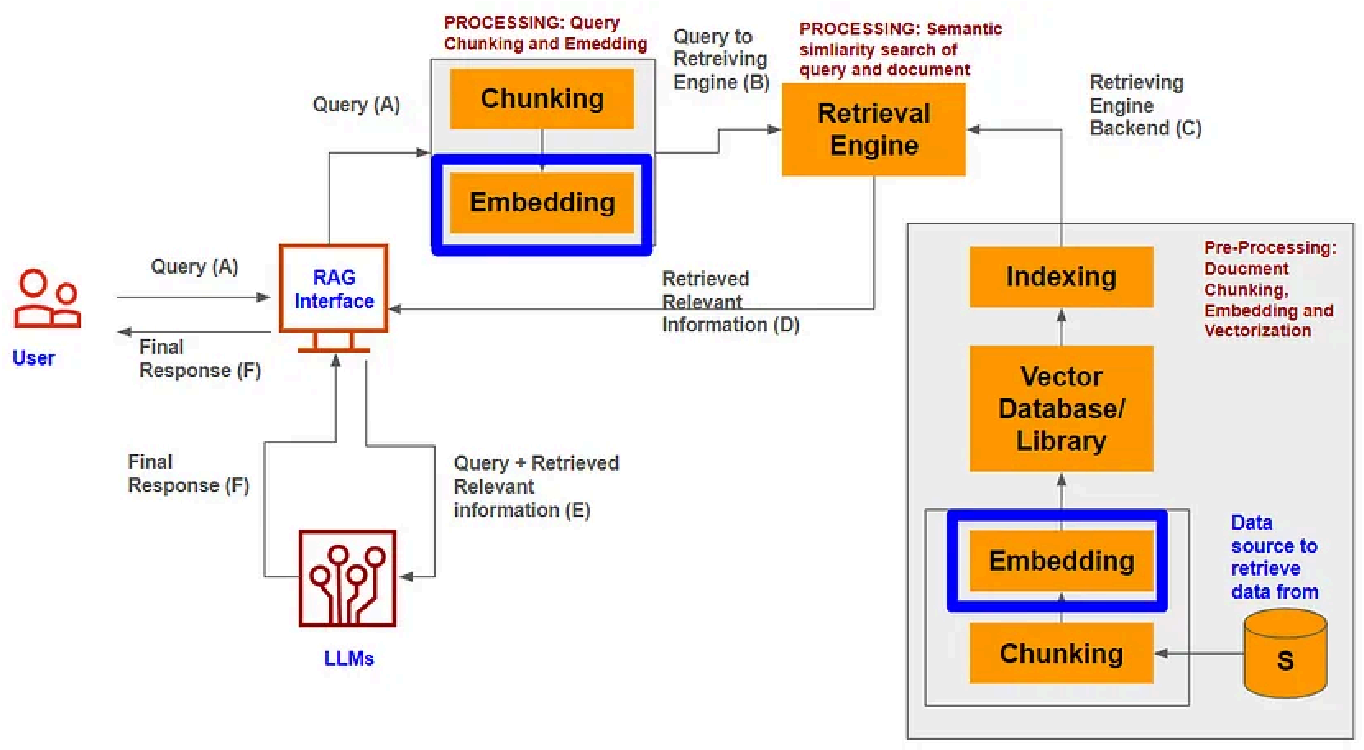

PROCESSING: Query Chunking and Emedding
Chunking
Embedding
Query to Retreiving Engine (B)
PROCESSING: Semantic similiarity search of query and document
Retrieval Engine
Retrieving Engine Backend (C)
Query (A)
Query (A)
RAG Interface
User
Final Response (F)
Retrieved Relevant Information (D)
Indexing
Pre-Processing: Doucment Chunking, Embedding and Vectorization
Vector Database/ Library
Final Response (F)
Query + Retrieved Relevant information (E)
Embedding
Data source to retrieve data from
LLMs
Chunking
S

Image by Author

What is Embedding?

In the last blog (Chunking), we discussed how we can break the source content (S) and the query into small chunks using various chunking strategies.

Now, as we know computers understand numbers, so these chunks of text have to be encoded to numbers (some mathematical form) which the computers can read, understand and process. Furthermore, we also would expect numbers to ensure that there are relationships between each word/chunk with the other words/chunks as well.

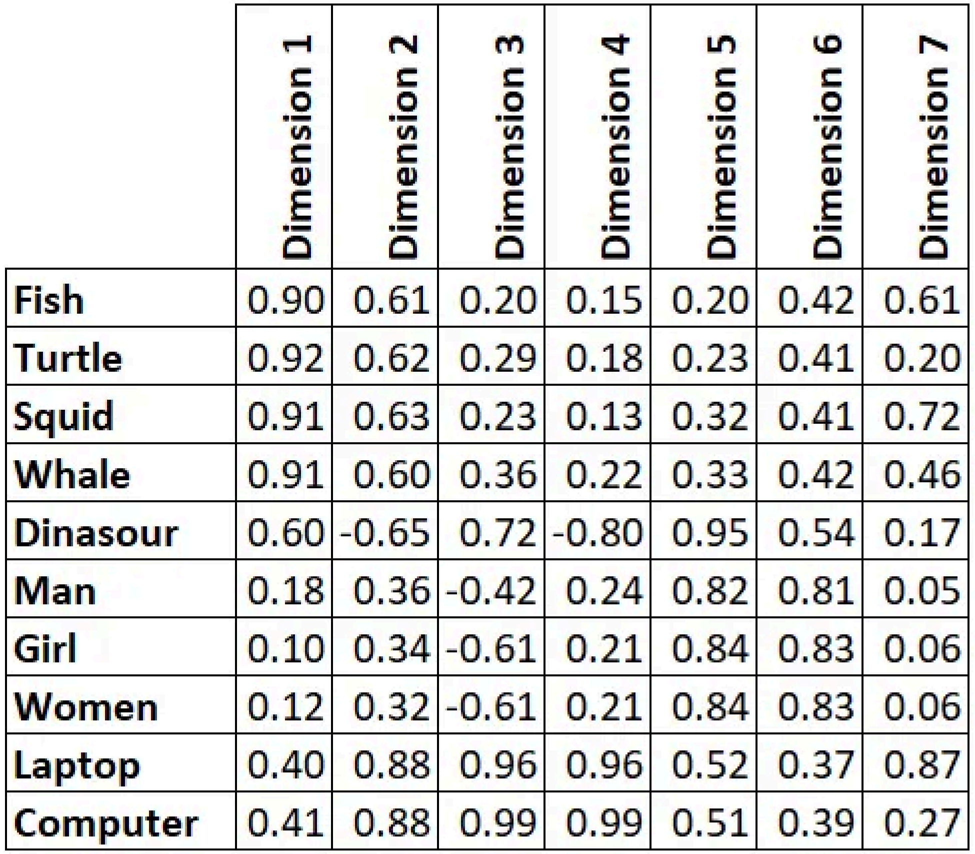
' *This conversion of chunks of text into a mathematical form is called embedding.* "

Since the models would use these numbers for computation, the numbering should be — meaningful and structured. To enable this there are some core principles behind embedding techniques:

- ✓ Ensure that number retains the contextual understanding of the text
- ✓ Ensure that number retains the semantic and syntactic properties of the text
- ✓ Ensure that number retains the linear relationship between words

In order to adhere to these principles, Embedding algorithms necessitate more than simply assigning numerical values to words/chunks. It requires a broader representation of these numbers which is achieved through high dimensional vectors.

Given below is an illustration of how embedding works in real world applications. As mentioned, embedding converts the words/chunks/sentences into vectors and the values of these vectors are such that the numbers in the vectors retain the contextual, semantic and syntactic properties. For example: as seen in the table below, the sea animals vectors are very similar to each other and are distinct from other embedding vectors for laptop/computer, man/women/girl. Note that embedding vectors can go into a very high n-dimensional space.



	Dimension 1	Dimension 2	Dimension 3	Dimension 4	Dimension 5	Dimension 6	Dimension 7
Fish	0.90	0.61	0.20	0.15	0.20	0.42	0.61
Turtle	0.92	0.62	0.29	0.18	0.23	0.41	0.20
Squid	0.91	0.63	0.23	0.13	0.32	0.41	0.72
Whale	0.91	0.60	0.36	0.22	0.33	0.42	0.46
Dinasour	0.60	-0.65	0.72	-0.80	0.95	0.54	0.17
Man	0.18	0.36	-0.42	0.24	0.82	0.81	0.05
Girl	0.10	0.34	-0.61	0.21	0.84	0.83	0.06
Women	0.12	0.32	-0.61	0.21	0.84	0.83	0.06
Laptop	0.40	0.88	0.96	0.96	0.52	0.37	0.87
Computer	0.41	0.88	0.99	0.99	0.51	0.39	0.27

Image by Author — Text Embedding

Note that Embedding is certainly not a new (RAG) concept in fact all NLP models like RNNs, LSTMs, ELMo, BERT, AlBERT and not GPT have thrived on the basis of evolution of Embedding methodologies.

It would not be incorrect to say that the better the sophistication of Embedding, the better would be model outcome — as embedding helps the model understand the "relationship" between the texts — both in terms of context and semantics. Over the years, the embedding algorithms have made significant progress, resulting in faster processing speeds and improved accuracy in generating language sequences and other downstream tasks.

Types of Embedding

While we have used text embedding as a specific example, Embeddings can also be utilized for various types of data, such as images, graphs, and more.

1. **Word Embeddings**: Embedding of Individual words. **Models:** Word2Vec, GloVe, and FastText
2. **Sentence Embeddings** Embedding of entire sentences as vectors that capture the overall meaning and context of the sentences. **Models:** Universal Sentence Encoder (USE) and SkipThought.
3. **Document Embeddings** Embedding of entire sentences capturing the semantic information and context of the entire document. **Models:** Doc2Vec and Paragraph Vectors
4. **Image Embeddings** — captures different visual features. **Models:** CNNs, ResNet and VGG

5. **User/Product Embeddings** represent users/products in a system as vectors. Capture user/products preferences, behaviors, attributes and characteristics. These are primarily used in recommendation systems.

Embedding — key use cases

Having performed embedding it is critical to make a note of how these embeddings are used downstream:

1. **Semantic Similarity and Distance:** One of the most popular applications of RAG (Retrieval-Augmented Generation) is measuring the semantic similarity between texts based on a given query. By organizing the vectors in a meaningful manner, embedding enables the underlying relationships between the query and the document, simplifying the process of comparing and retrieving relevant results.
2. **Dimensionality Reduction:** Text embeddings result in the conversion of text into lower-dimensional vectors. This conversion is advantageous as it reduces the dimensionality of the inputs, leading to improved computational efficiency.
3. **Transfer Learning:** Text embeddings facilitate transfer learning by enabling a model to be pre-training on a large dataset and then fine-tune

on a smaller, domain-specific dataset.

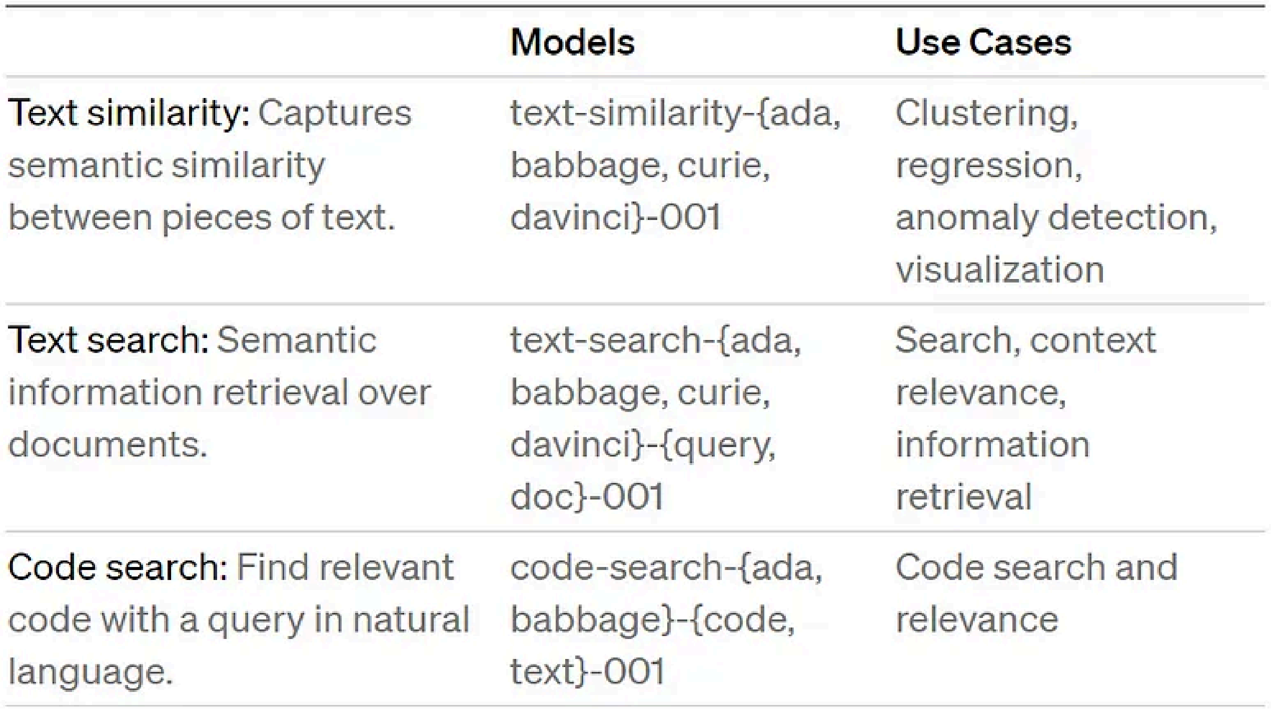
4. **Machine Learning Models:** Embedding is a key step for any machine learning modeling — clustering, classification etc.
5. **Multi-modal support:** Vector databases can store embeddings of multimodal data — like image captioning, visual question answering, or speech recognition. Given the vectors are stored in similar manner this enables LLMs to integrate different cross functional tasks.

Common Embedding models (from RAG perspective)

By training a neural network on a large corpus of text, embeddings are obtained. This training allows the network to assign similar vectors in the embedding space to words that are used in similar contexts. Consequently, the embeddings capture the semantic relationships between words and the specific contextual nuances in which they are employed.

As you can expect, there are a plethora of models available for embedding. Given below are some key models from established players like Open AI, Mistral etc.

1. **Cohere's Embedding:** Powerful for processing short texts with under 512 tokens. For longer texts, the API truncates input to fit the maximum context length and embed the text.
2. **Mistral Embedding:** Strong embedding for AI/ML modeling like text classification, sentiment analysis etc. Embeddings are renowned for their user-friendliness, resilience, and capacity to handle vast amounts of data. This empowers businesses to utilize language embeddings for improved data insights and innovative AI-driven solutions.
3. **Open AI Embeddings:** Open AI is currently one of the market leaders for Embedding Algorithms. Of the all, OpenAI second-gen text-embedding model, ada-002, has proven to give top-notch results across various use cases. Furthermore, the model is cost-effectiveness, and user-friendliness. Open AI Embedding models are trained for specific use cases like Text Similarity, Text Search and Code search. The common Embedding algorithms are ada, babbage, curie, davinci 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

Source: Open AI

One key aspect to emphasize here is that the costing of these Embedding models are very different and could be steep for some strong models — these are a function of Sequence length and Embedding dimensions — therefore it is imperative that appropriate Embedding algorithms are used for the given use cases.

Note that it is becoming really common to create your own customized Embedding model for specific use cases.

Evaluating the performance of Embeddings models

As you would have acknowledged by now that there are a lot of Embeddings algorithms available in the market and more are being developed everyday. This raises a key question, how can we measure the effectiveness of one Embedding algorithm over others. One of the most common and sophisticated way of how this is done is using Hugging Face's Massive Text Embedding Benchmark (MTEB) Leaderboard. MTEB is a useful tool for assessing the effectiveness of different text embedding models in a wide range of embedding tasks.

MTEB offers a comprehensive explanation of the leaderboard's objective and provides valuable insights into various text embedding models. MTEB evaluates the performance of the Embedding model across 8 tasks (text mining, classification, clustering, pair classification, reranking, retrieval, semantic textual similarity (STS), and summarization) and 58 datasets.

P S: I plan to write a blog on MTEB sometime soon.

Conclusion

In conclusion, embedding plays a crucial role in the development and success of NLP models. By transforming words or sentences into numerical representations, embedding allows NLP models to understand and process textual data effectively. Embedding not only captures the semantic meaning of words but also enables the models to recognize relationships and similarities between different words or sentences. With the availability of pre-trained embedding models and resources like MTEB, researchers and developers can evaluate and compare the performance of various embedding techniques. As NLP continues to advance, embedding will remain a key component in enhancing the accuracy and performance of NLP models, enabling them to tackle complex language tasks with greater precision.

Here is a view of what's in the series, feel free to let me know if you would like me to cover any specific aspect.

1. **Retrieval-Augmented Generation (RAG) — Basics to Advanced Series (Part 1)**

2. **Pre-processing block — Chunking (part 2): strategies, considerations and optimization**