RAG generate embeddings phase

01/10/2025

In the previous steps of your Retrieval-Augmented Generation (RAG) solution, you divided your documents into chunks and enriched the chunks. In this step, you generate embeddings for those chunks and any metadata fields on which you plan to perform vector searches.

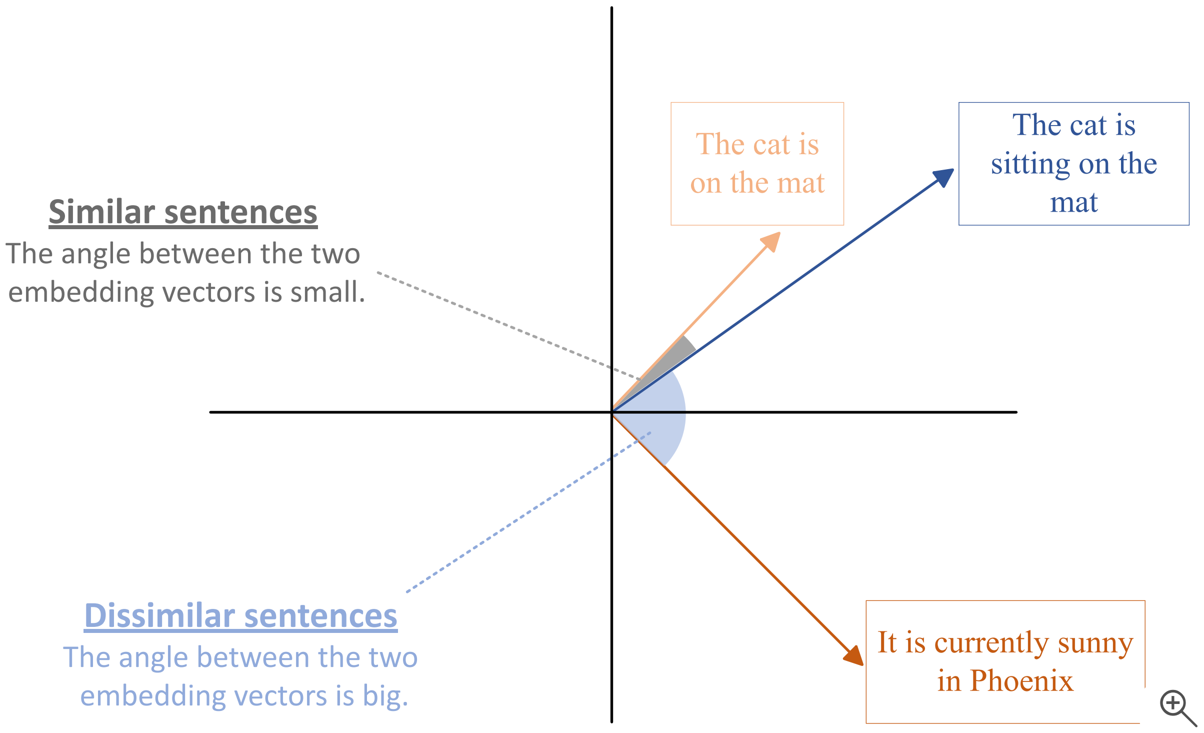
This article is part of a series. Read the introduction.

An embedding is a mathematical representation of an object, such as text. When a neural network is being trained, many representations of an object are created. Each representation has connections to other objects in the network. An embedding is important because it captures the semantic meaning of the object.

The representation of one object has connections to representations of other objects, so you can compare objects mathematically. The following example shows how embeddings capture semantic meaning and relationships between each other:

```
embedding (king) - embedding (man) + embedding (woman) = embedding (queen)
```

Embeddings are compared to one another by using the notions of similarity and distance. The following grid shows a comparison of embeddings.


Similar sentences
The angle between the two embedding vectors is small.
The cat is on the mat
The cat is sitting on the mat
Dissimilar sentences
The angle between the two embedding vectors is big.
It is currently sunny in Phoenix

In a RAG solution, you often embed the user query by using the same embedding model as your chunks. Then, you search your database for relevant vectors to return the most semantically relevant chunks. The original text of the relevant chunks is passed to the language model as grounding data.

Note

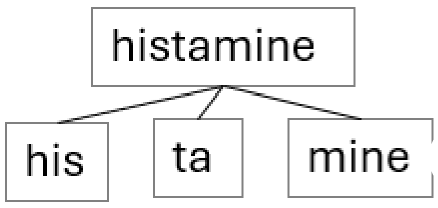
Vectors represent the semantic meaning of text in a way that allows for mathematical comparison. You must clean the chunks so that the mathematical proximity between vectors accurately reflects their semantic relevancy.

The importance of the embedding model

The embedding model that you choose can significantly affect the relevancy of your vector search results. You must consider the vocabulary of the embedding model. Every embedding model is trained with a specific vocabulary. For example, the vocabulary size of the BERT model is about 30,000 words.

The vocabulary of an embedding model is important because it handles words that aren't in its vocabulary in a unique manner. If a word isn't in the model's vocabulary, it still calculates a vector for it. To do this, many models break down the words into subwords. They treat the subwords as distinct tokens, or they aggregate the vectors for the subwords to create a single embedding.

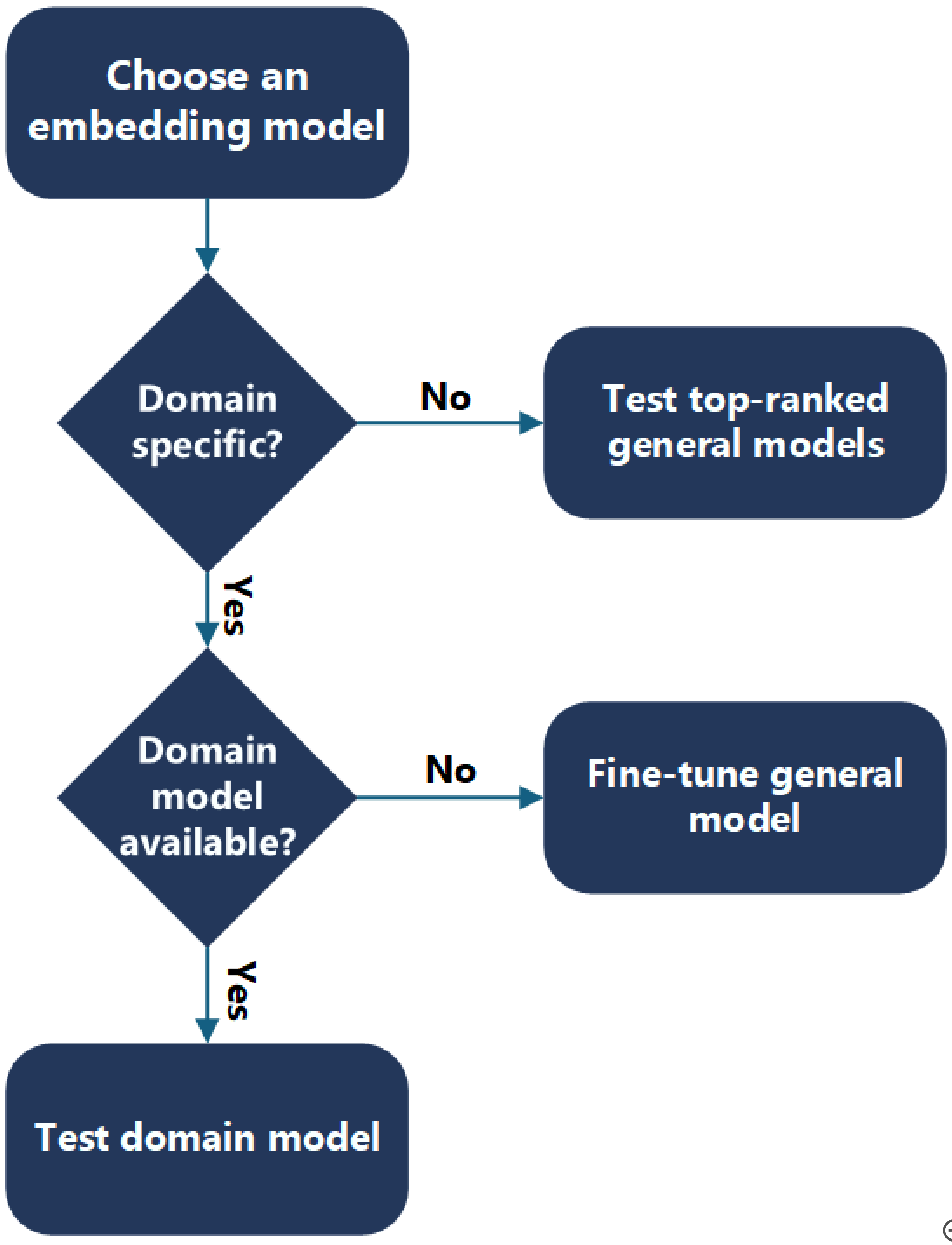
For example, the word *histamine* might not be in an embedding model's vocabulary. The word *histamine* has a semantic meaning as a chemical that your body releases, which causes allergy symptoms. The embedding model doesn't contain *histamine*. So, it might separate the word into subwords that are in its vocabulary, such as *his*, *ta*, and *mine*.



The semantic meanings of these subwords are far from the meaning of *histamine*. The individual or combined vector values of the subwords result in poorer vector matches compared to if *histamine* were in the model's vocabulary.

Choose an embedding model

Determine the right embedding model for your use case. Consider the overlap between the embedding model's vocabulary and your data's words when you choose an embedding model.



First, determine whether you have domain-specific content. For example, are your documents specific to a use case, your organization, or an industry? A good way to determine domain specificity is to check whether you can find the entities and keywords in your content on the internet. If you can, a general embedding model likely can, too.

General or non-domain-specific content

When you choose a general embedding model, start with the Hugging Face leaderboard. Get up-to-date embedding model rankings. Evaluate how the models work with your data, and start with the top-ranking models.

Domain-specific content

For domain-specific content, determine whether you can use a domain-specific model. For example, your data might be in the biomedical domain, so you might use the BioGPT model. This language model is pretrained on a large collection of biomedical literature. You can use it for biomedical text mining and generation. If domain-specific models are available, evaluate how these models work with your data.

If you don't have a domain-specific model, or the domain-specific model doesn't perform well, you can fine-tune a general embedding model with your domain-specific vocabulary.

Important

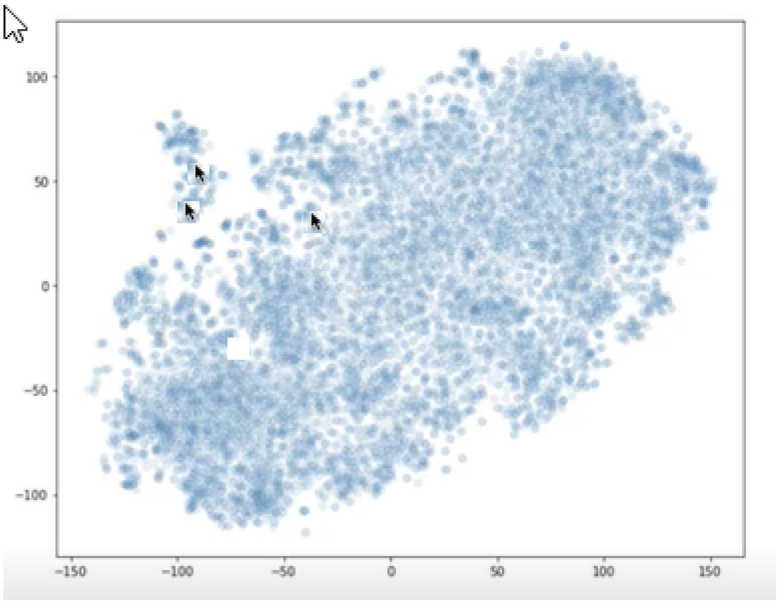
For any model that you choose, you need to verify that the license suits your needs and the model provides the necessary language support.

Evaluate embedding models

To evaluate an embedding model, visualize the embeddings and evaluate the distance between the question and chunk vectors.

Visualize embeddings

You can use libraries, such as t-SNE, to plot the vectors for your chunks and your question on an X-Y graph. You can then determine how far the chunks are from one another and from the question. The following graph shows chunk vectors plotted. The two arrows near one another represent two chunk vectors. The other arrow represents a question vector. You can use this visualization to understand how far the question is from the chunks.



Calculate embedding distances

You can use a programmatic method to evaluate how well your embedding model works with your questions and chunks. Calculate the distance between the question vectors and the chunk vectors. You can use the Euclidean distance or the Manhattan distance.

Embedding economics

When you choose an embedding model, you must navigate a trade-off between performance and cost. Large embedding models usually have better performance on benchmarking datasets. But, the increased performance adds cost. Large vectors require more space in a vector database. They also require more computational resources and time to compare embeddings. Small embedding models usually have lower performance on the same benchmarks. They require less space in your vector database and less compute and time to compare embeddings.

When you design your system, you should consider the cost of embedding in terms of storage, compute, and performance requirements. You must validate the performance of the models through experimentation. The publicly available benchmarks are mainly academic datasets and might not directly apply to your business data and use cases. Depending on the requirements, you can favor performance over cost or accept a trade-off of good-enough performance for lower cost.

Next step

Information-retrieval phase

Related resources

- Understand embeddings in Azure OpenAI Service
- Tutorial: Explore Azure OpenAI Service embeddings and document search