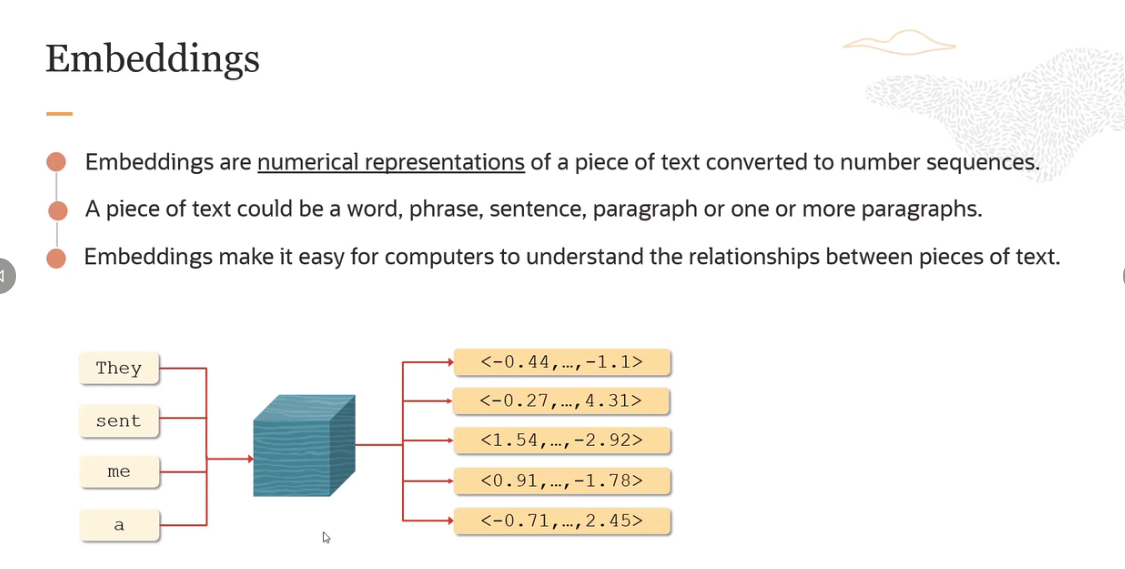
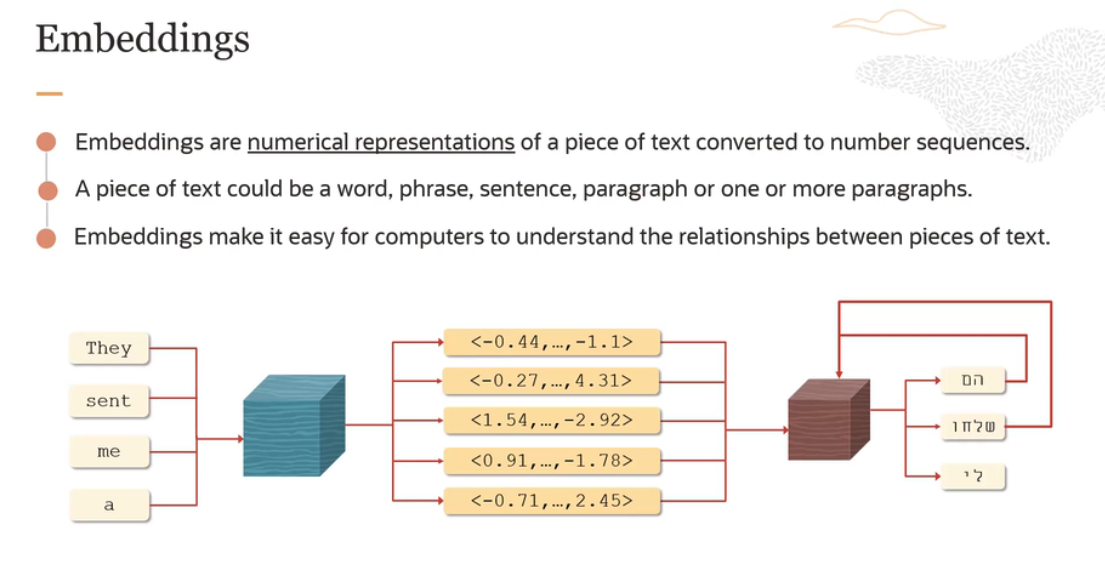
**Embedding Models available in the OCI Generative AI service.**



In the example "They sent me a," or rather, the phrase, "They sent me a," and I sent this to an encoder. It's an LLM architecture.

Now, encoders are really designed to take that text and convert it into a vector. So there are five vectors. There is a vector for each of the words, and there is also a vector representation for the sentence. So that's basically what the illustration is showing here.

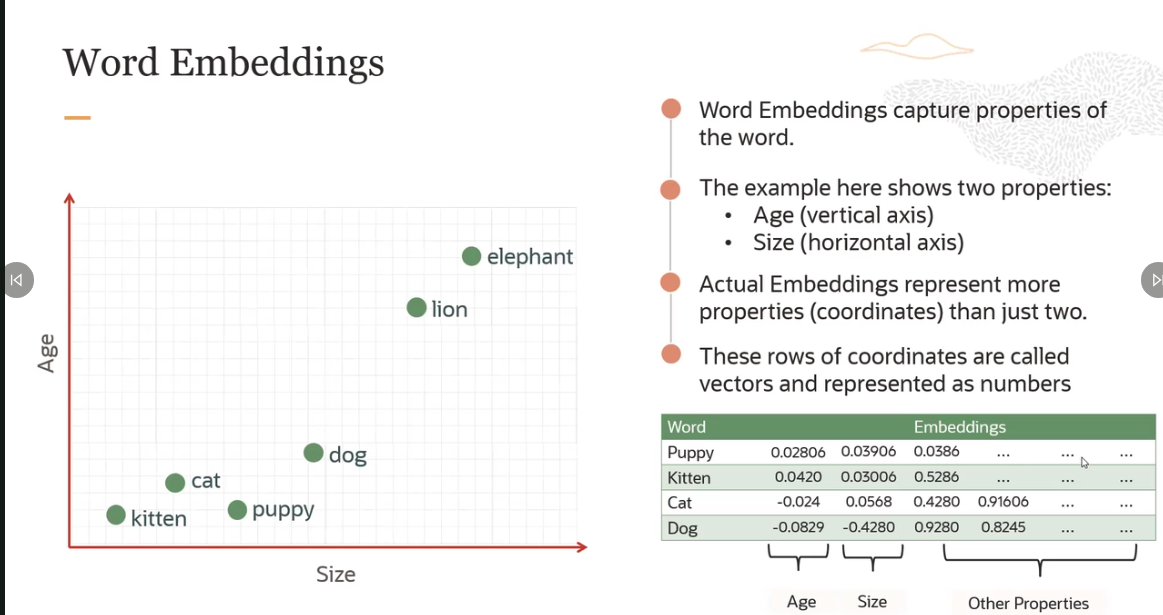
**what is the use of doing so? Well, the real use case comes through when you are doing something like translation which is a sequence-to-sequence task.**



So you have an encoder here, and then you have a decoder here. So the set of tokens is passed to the model here, the encoder, set of tokens or the words.

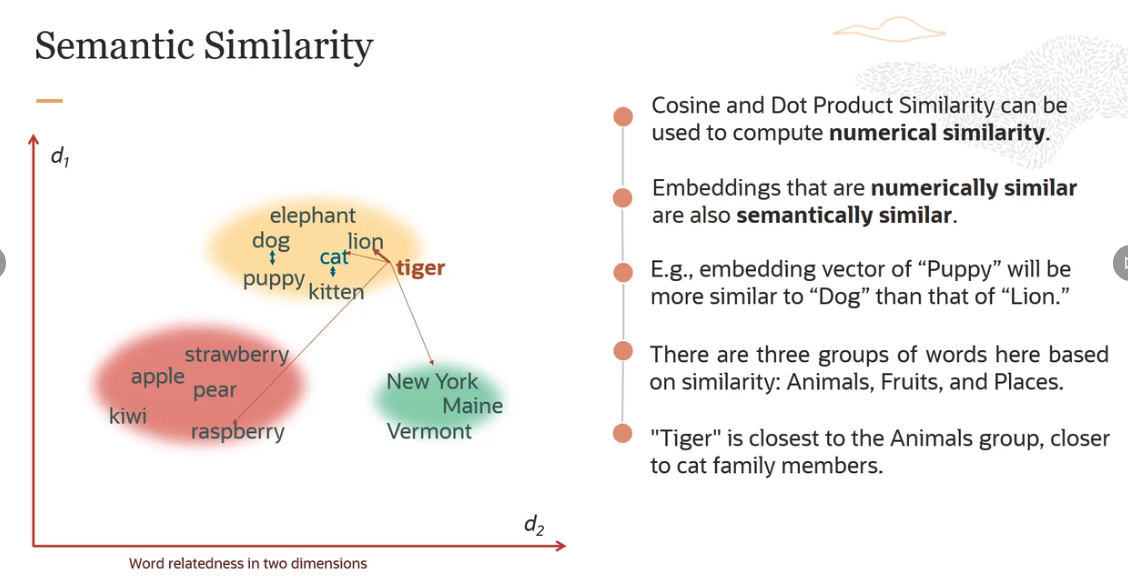
And it takes these words or tokens and converts them into vectors. And these vectors are then fed into a decoder, which starts emitting tokens in a self-referential loop.

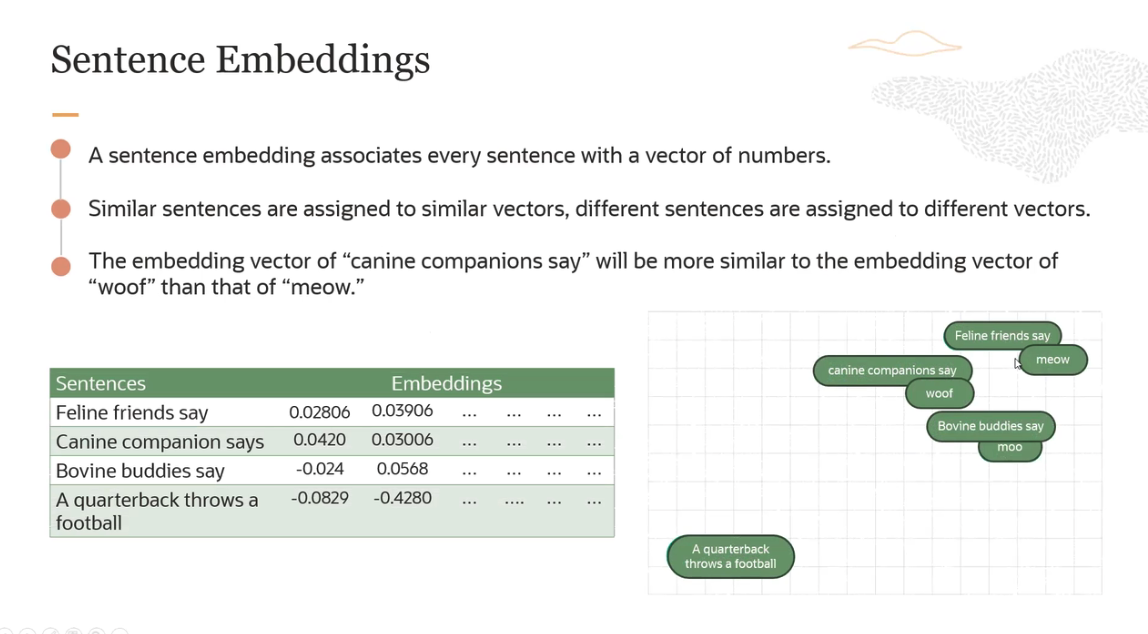
It translate text from English to Hebrew.

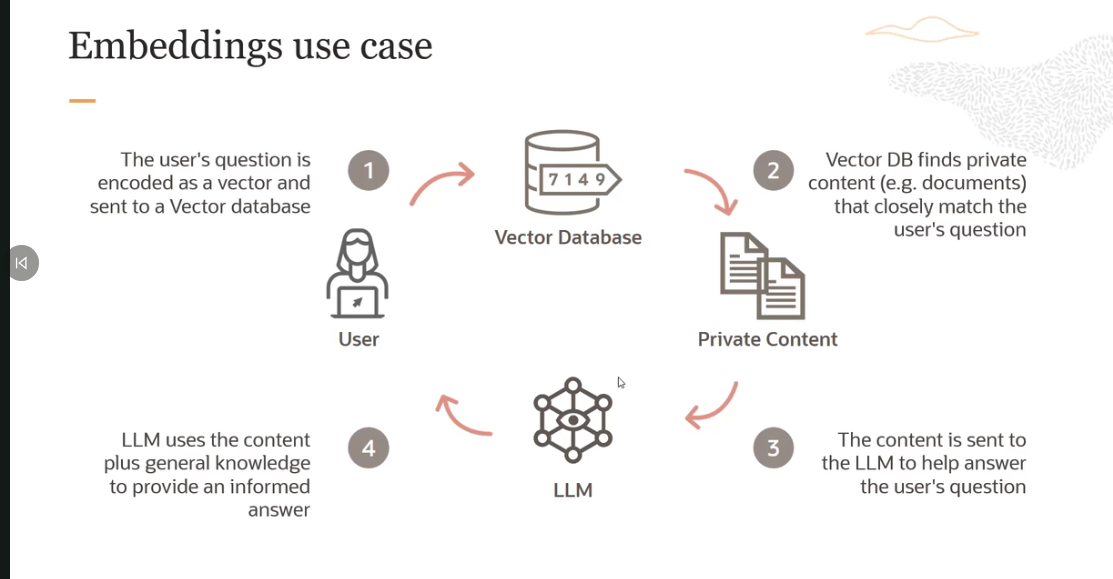


**Semantic Similarity:**

Semantically similar basically means how close their meaning is or how closely they are related.



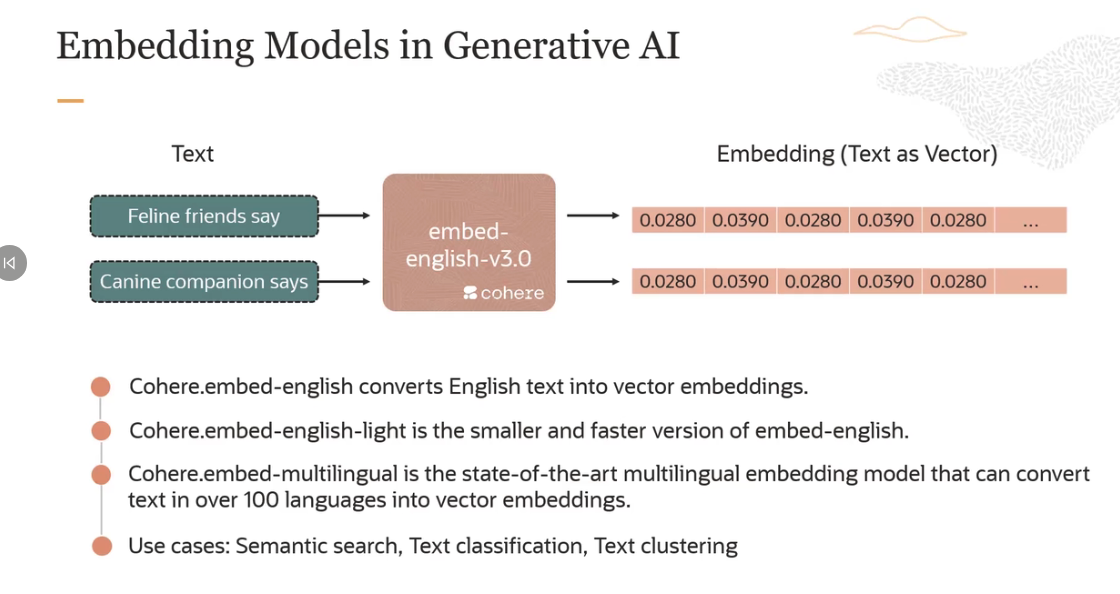


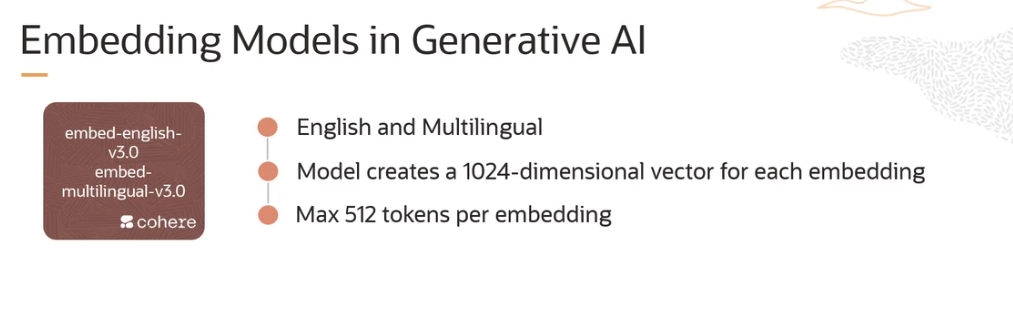


- Let's say you have a user who has some question which cannot be answered by LLM. Maybe it's related to your customer support calls or something. So the user question is encoded as a vector and sent to the vector database.

- Now vector database can run a nearest match in the vector database to identify the most closely associated documents or paragraphs. And it finds this is the private content which closely matches the user query. And then what it can do is it can take those documents or those paragraphs and insert those into a prompt to be sent to the large language model. And basic idea is to help answer the user question by changing the prompt. And then the LLM uses the content which has been given by the vector database plus its general knowledge to provide an informed answer.

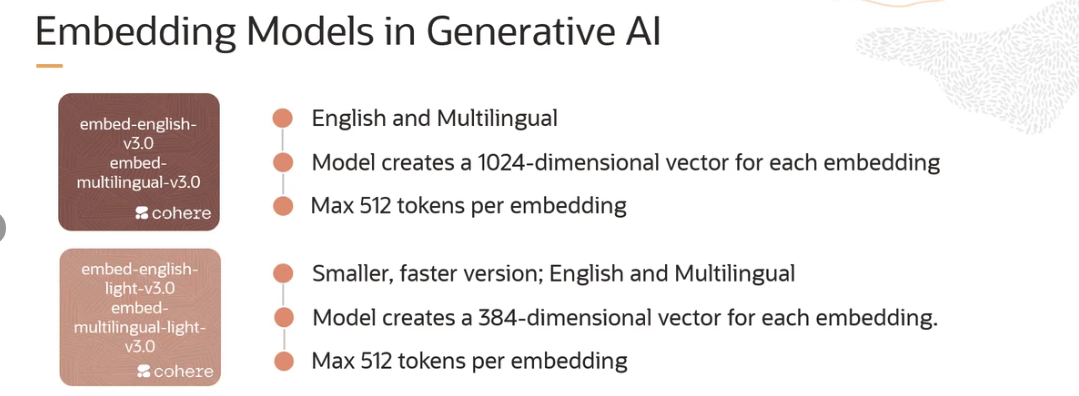
- So this is how RAG systems work. And good retrieval quality is essential to make this work. And this is where embeddings play a major role.

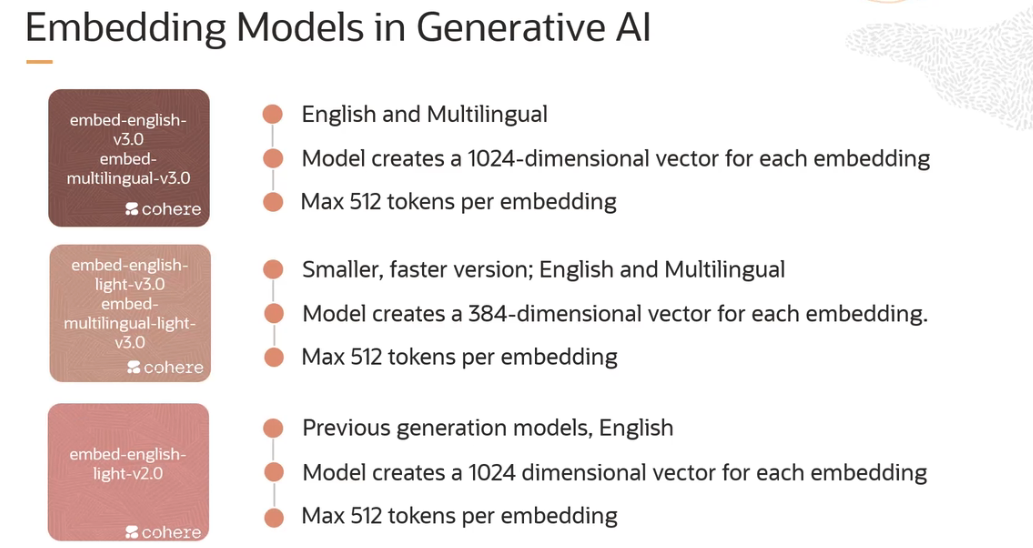




They support English and multilingual languages. The model creates a 1,024-dimensional vector for each embedding. So every embedding, whether it's a sentence, a phrase, or paragraph, gets converted into 1,024-dimensional vector. And the model takes a max of 512 tokens per embedding.

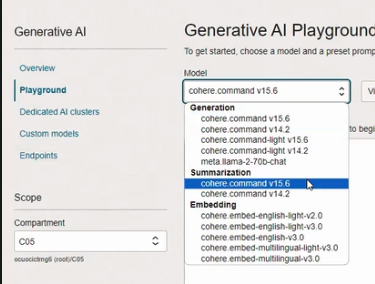
The v3 embed model, this is something which Cohere launched recently. And one of the key improvements in embed v3 is **its ability to evaluate how well a query matches a document's topic and assess the overall quality of the content.**

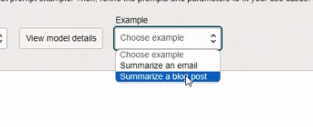






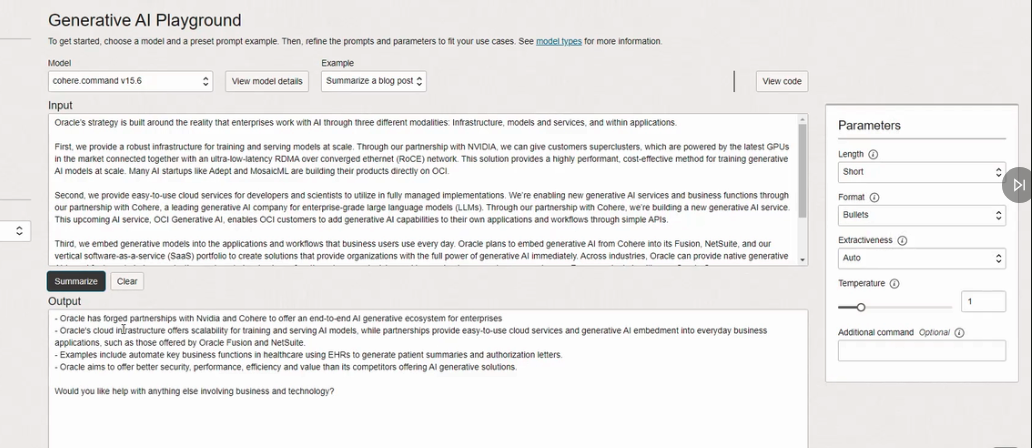
Go to playground:



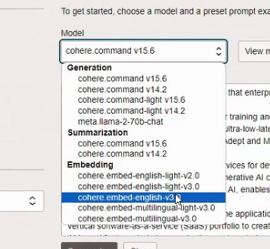


Summarize a blog post

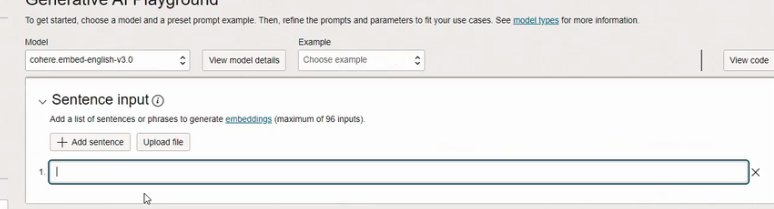
click Summarize in bottom

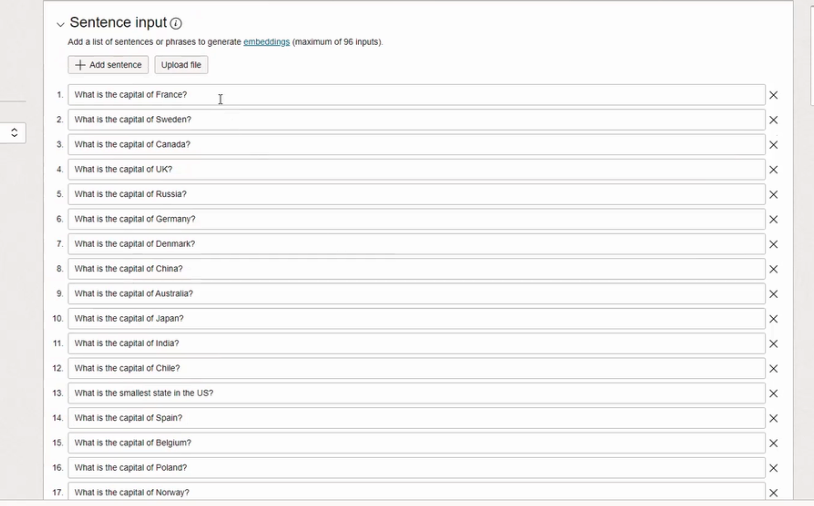


For Embedding Models:

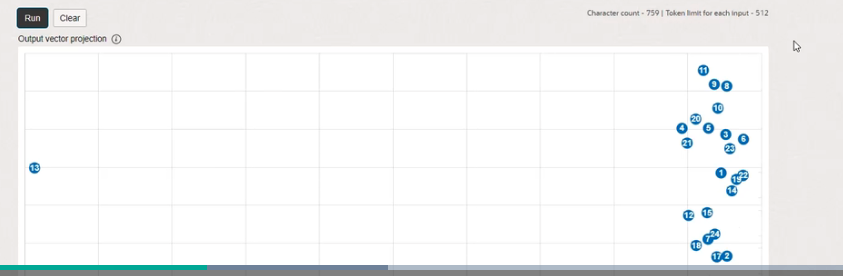


Provide a list of sentences / upload file where you have listed sentences in the box: (Embeddings are numerical representations of a piece of text converted to number sequences.)

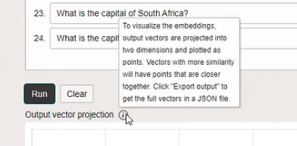




Hit Run to get an output:



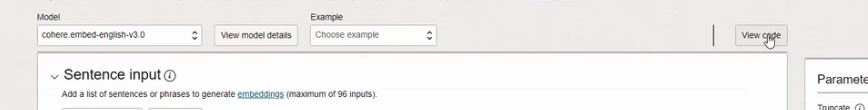
The model is capturing the context and meaning of each piece of text and it is representing them as embeddings. Each dimension of an embedding represents a certain universal characteristic about the text, according to how the model understands it.

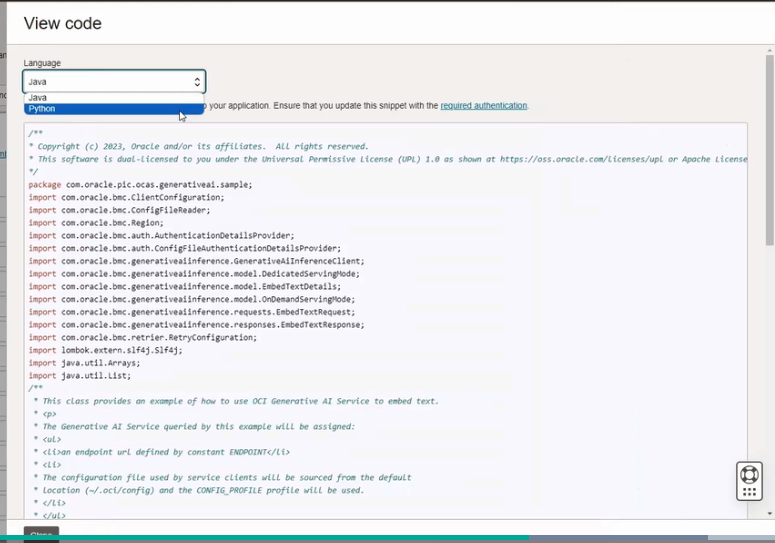


Embeddings that are numerically similar are also semantically similar, meaning text of similar meaning are indeed located close together.

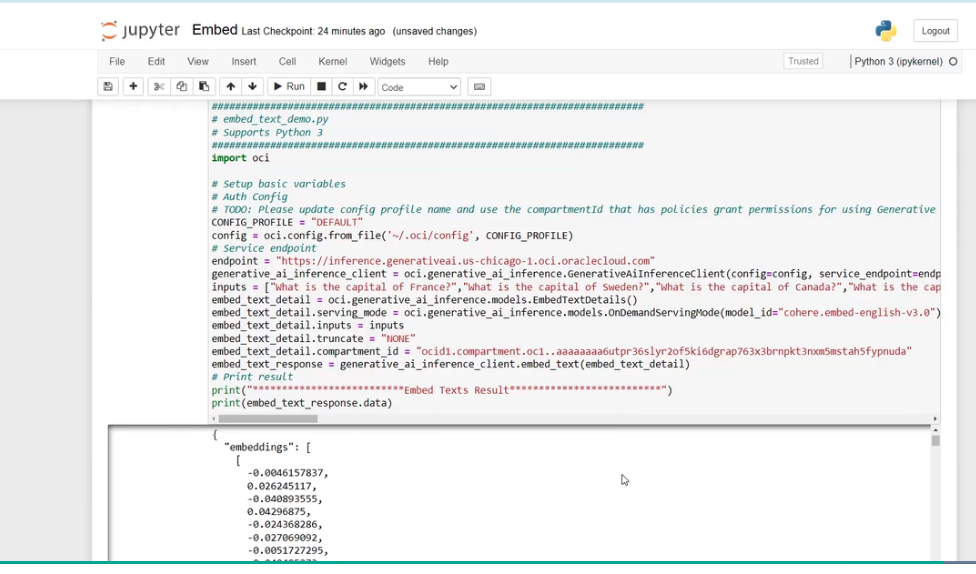
The model creates 1,024 dimensional vector for each embedding. How to see that

Click view code.





copy this code and run this in Jupyter Notebook



we have 26 embeddings (sentences) in total. And for each of the embeddings, the model is creating 1,024 dimensional vector. And these vectors are floating point numbers. So it's difficult to visualize. To do that copy that code in visual studio code.

