You'll see how each word in the index is assigned a unique vocabulary and is represented as a vector where all elements are zero, except for the position corresponding to the word's index, which is set to one. So, the size of the vector would be equal to the vocabulary size. This makes it excessively large in most cases.

In contrast, embeddings usually have a dimension between 50 and 300. Add to this the fact that words that are semantically similar are closer in the embedding space, which makes them much more efficient in terms of storage and computation.

# THE IMPORTANCE OF EMBEDDINGS

Embeddings form the foundation for achieving precise and contextually relevant LLM outputs across different tasks. Let's explore the diverse applications where embeddings play an indispensable role.

## Question Answering

Embeddings play a crucial role in enhancing the performance of Question Answering (QA) systems within RAG applications. By encoding questions and potential answers into high-dimensional vectors, embeddings allow the efficient retrieval of relevant information. The semantic understanding captured by embeddings facilitates accurate matching between queries and context, enabling the QA system to provide more precise and contextually relevant answers.

## Conversational Search

Conversations involve dynamic and evolving contexts, and embeddings help represent the nuances and relationships within the dialogue. By encoding user queries and system responses, embeddings enable the RAG system to retrieve relevant information and generate context-aware responses.

## InContext Learning (ICL)

The model's effectiveness in InContext Learning is highly dependent on the choice of few shot demonstrations. Traditionally, a fixed set of demonstrations was employed, limiting the adaptability of the model. Rather than relying on a predetermined set of examples, this novel approach involves retrieving demonstrations relevant to the context of each input query.

The implementation of this demonstration retrieval is relatively straightforward, utilizing existing databases and retrieval systems. This dynamic approach enhances the learning process's efficiency and scalability and addresses biases inherent in manual example selection.

## Tool Fetching

Tool fetching involves retrieving relevant tools or resources based on user queries or needs. Embeddings encode the semantics of the user's request and the available tools, enabling the RAG system to perform effective retrieval and present contextually relevant tools. The use of embeddings enhances the accuracy of tool recommendations, contributing to a more efficient and user-friendly experience.

# IMPACT OF EMBEDDINGS ON RAG PERFORMANCE

Which encoder you select to generate embeddings is a critical decision that will hugely impact the overall success of the RAG system. Low-quality embeddings lead to poor retrieval. Let's review some selection criteria to consider before making your decision.

## Vector Dimension and Performance Evaluation

When selecting an embedding model, consider the vector dimension, average retrieval performance, and model size. The Massive Text Embedding Benchmark (MTEB) provides insights into popular embedding models from OpenAI, Cohere, and Voyager, among others. However, custom evaluation on your dataset is essential for accurate performance assessment.

## Private vs. Public Embedding Model

Although the embedding model provides ease of use, it entails certain trade-offs. The private embedding API, in particular, offers high availability without the need for intricate model hosting engineering. However, this convenience is counterbalanced by scaling limitations. It's crucial to verify the rate limits and explore options for increasing them. A notable advantage is that model improvements come at no extra cost.

Companies such as OpenAI, Cohere, and Voyage consistently release enhanced embedding models. Simply run your benchmark for the new model and implement a minor change in the API, making the process exceptionally convenient.