TIM 175: BUSINESS STRATEGY AND INFORMATION SYSTEMS
SPRING 2025

Vector Embeddings and RAG introduction

TIM 175 WEEK 7 PRELAB

Now that we understand prompt engineering workflows, we will start learning about Retrieval Augmented Generation. In this prelab, you will learn the core concepts of vector embeddings, vector databases, and RAG. **This individual prelab is due Tuesday 11:59pm.**

Readings: We mark up to two readings with a ⭐that we suggest you read

- Vector Search and Embeddings
- Vector Database Article ⭐
- Grounding for Gemini with Vertex AI Search and DIY RAG
- What is Rag? ⭐
- 2024: The State of Generative AI in Enterprise

Submission Link
Week 7 TIM 175 Submission Form (Spring 2025)

Brief Task Overview
Complete the following activities to understand vector embeddings, vector databases and their role in similarity-based retrieval tasks.

1. Activity 1: Introduction to word embeddings
2. Activity 2: Introduction to sentence embeddings
3. Activity 3: Retrieval as a similarity task
4. Activity 4: Introduction to creating and querying Vector databases
5. Activity 5: Using vector databases for RAG

You can use ChatGPT or other GenAI tools to inform any part of the assignment but: (1) you need to first form your own independent thoughts, (2) every word included in the submission needs to be something you've read, thought about, and decided to include, and (3) you should strive towards submitting the highest quality work you can rather than mediocre work that meets the requirements.

Vector Embeddings and RAG introduction

Uptil now, we were learning about prompting and prompt engineering where our models simply generate outputs based on our instructions in the prompt. Now we want to learn how we can ground the responses from the model in specific contexts by giving it data or files that it utilizes for creating its output. This is where vector databases and RAG come in.

Vector Embeddings: Vector embeddings are a form of transforming raw data into a numerical format that AI systems can understand. This involves converting data, like text or images, into a series of

numbers, known as vectors, in a high-dimensional space, capturing the nuanced characteristics of the data.

Vector Database: Vector databases store these vector embeddings and can be queried to retrieve certain vectors based on the context.

RAG: Retrieval Augmented Generation (RAG) is a natural language processing (NLP) technique that combines the strengths of information retrieval and text generation. By using a vector database to store and index information, RAG can retrieve relevant documents or data snippets and then use these retrieved elements to generate coherent, contextually appropriate responses. This approach enhances the quality and relevance of the generated content, making it more accurate and useful. Think of it as a technique that allows you to harness the power of generative AI upon your own data.

Why this exercise? Writing RAG code using a library like LangChain doesn't really teach you much about how the process works under the hood, since these libraries often abstract away all of the core logic down to a single line of code. Implementing RAG will give you an adequate understanding of how it actually works, such that you won't see RAG as some magical black box when it comes time to use it in your projects.

Before moving, make sure you cover the readings listed above to get a clear understanding of these terms.

Setup:

1. Create a copy of this Google Document as in previous weeks
2. Open and Create a **copy** of TIM175 Week 7 - Vector embeddings and Retreival and make sure you can run code in a Python notebook
3. Follow the instructions on the Colab Notebook and complete the Activities
4. Reflect on each Activity in the reflection boxes below.

Activity 1:

In this activity, you will get an introduction to word embeddings and how they can be derived and used. Follow the instructions on the Google Collab and then answer the reflections below:

List the words you tested in Activity 1.1:

Write the modifications made in Activity 1.2:

```
composite = starter + model.encode("YOUR_WORD_HERE") -
model.encode("YOUR_WORD_HERE")
```

```
composite = starter + model.encode("YOUR_WORD_HERE") -
model.encode("YOUR_WORD_HERE")
```

What did you learn from this activity? Did you find anything strange when doing the comparisons between words or did it all make sense?

Activity 2:

In this activity, you will get an introduction to sentence embeddings and how they can be derived and used. Follow the instructions on the Google Collab and then answer the reflections below:

Your final Guess for Activity 2.1

What was the similarity of your final guess? How did you achieve your final guess? What did you learn from this activity?

Activity 3:

In this activity, you will get an introduction to retrieving using sentence embeddings. Follow the instructions on the Google Collab and then answer the reflections below:

Your database of quotes: [“ “]

5 Queries you tested out:

What were the results like for each of your 5 queries? Were you surprised by any of them? What did you learn from this activity?

Activity 4:

In this activity, you will create a vector database using a document and then retrieve chunks from it. Follow the instructions on the Google Collab and then answer the reflections below:

One line describing what file you uploaded:

3 Queries you tested out:

What were the results like for each of your 3 queries? Were you surprised by any of them? What did you learn from this activity?

Activity 5:

In this activity, you will use the retrieved documents and give them to an LLM to create an output based on the retrieved chunks as context. Follow the instructions on the Google Collab and then answer the reflections below:

Your query: <Insert query here>

Compare the outputs between the responses of the LLM without RAG and response of the LLM with RAG? What differences do you see? What did you learn from this comparison?

After refining the basic prompt:

Your final prompt? How did you refine it? Did you play around with any settings in the RAG pipeline? What did you learn from it?

Submission Instructions:

After completing all the activities, fill out the submission form including a link to your copy of **this google document** on it.