Learn Vector Database using Python, Pinecone, LangChain, Open AI, Hugging Face and build out AI, ML , Chat applications

**What you'll learn**

* Pinecone Vector Database, LangChain, Transformer Models for vector embedding, Generative AI, Open AI API Usage, Hugging Face Models
* Master the essential techniques for vector data embedding, indexing, and retrieval.
* A Practical Code Along with Semantic Search Use Case in Detail with Named Entity Recognition
* Developing an AI Chat Bot for Cognitive Search on Private Data Using LangChain
* Understand the fundamentals of vector databases and their role in AI, generative AI, and LLM (Language Model Models).
* Explore various vector database technologies, including Pinecone, and learn how to set up and configure a vector database environment.
* Learn how vector databases enhance AI workflows by enabling efficient similarity search and nearest neighbor retrieval.
* Gain practical knowledge on integrating vector databases with Python, utilizing popular libraries like NumPy, Pandas, and scikit-learn.
* Implement code along exercises to build and optimize vector indexing systems for real-world applications.
* Explore practical use cases of vector databases in AI, generative AI, and LLM, such as recommendation systems, content generation, and language translation.
* Understand how vector databases can handle large-scale datasets and support real-time inference.
* Gain insights into performance optimization techniques, scalability considerations, and best practices for vector database implementation.

Goal of this assignment is to make yourself familiar how to use the hugging face documentation. In this particular assignment I will encourage you to go over this URL, read, understand and explore by yourself other models. <https://huggingface.co/sentence-transformers/all-MiniLM-L12-v2>

Named Entity Recognition (NER)– proper names, organisations, places, location, dates, brands, product names, amounts of money as opposed to general nouns and verbs

Python module ‘spacy’ can be used to recognize named entities.

Use Cases:

Information extraction

Improved search and recommendation systems

Entity linking and knowledge graphs

Text summarisation and document organisation

Named entity disambiguation

Sentiment Analysis and opinion mining

Legal and Conpliance applications

Social media monitoring

Implementation

START -> Text Processing -> Part of Speech tagging (POS) -> Named Entity Recognition -> Entity Classification -> Post Processing -> Entity Disambiguation -> Output Generation -> END

[Course: Master Vector Database with Python for AI & LLM Use Cases | Udemy Business](https://wipro.udemy.com/course/vector-db/learn/practice/1506108/introduction#overview)

**Assignment: Build a video search engine based on audio content where "text" is search input.**

100 minutes to complete

2 student solutions

You will architect and implement how you can build a video search back end using concepts learned in this course. Let's say you have all your office meetings recorded and transcribed. Now you will build a cognitive/semantic search engine for those recordings.

**Assignment instructions**

100 minutes to complete

2 student solutions

Here are few high level steps you can use as a guideline.

1. Identify your video sources: Organize them in such you with id and their url so you can identify them later on. Also keep in mind your hardware resources. Better to use smaller videos e.g. 1-5 mins.
2. Transcribe your videos: Convert your videos to text. If you don't have transcription tool you can extract audio from a video using opensource tools like "shotcut". And then you can use Open AI whisper to transcribe.
3. Create a semantic text search back end: You will find semantic search technique taught in this course useful for this purpose.
4. Push to GitHub and share your repository with the class in a comment/question to this assignment.

#### Questions for this assignment

Have you identified your video sources? What are those sources?

How many video files you have identified?

What is the average video length of your files?

Have you successfully transcribed all your video files to text? What are the tools and techniques you used?

What is average words/token per video?

What transformer model you choose and why?

What is the vector dimension of your index?

How good your video search application? How did you determine that?

What could be improved on this project?

Can you summarize your learning from this capstone project and identify future work?

Did you share your project with rest of the worlds and your peers in this class? You could share by pushing your codes and readme file in GitHub or in a blog post. Please share the URL with your class?

TLDR READING LIST

**Modern client-side routing: the Navigation API (16 minute read)** 

<https://developer.chrome.com/docs/web-platform/navigation-api?utm_source=tldrwebdev>

SPAs (Single Page Applications) are defined by a core feature: they dynamically rewrite their content as users interact with them instead of the default method of loading entirely new pages from the server. SPAs are able to provide this feature via the History API, a clunky API developed long before SPAs were the norm. The Navigation API is a newly proposed API that completely overhauls this space that promises to be easier to adapt. This page describes the Navigation API at a high level with code examples on popular usages.

**Coding interviews are effective (2 minute read)**

https://www.pcloadletter.dev/blog/coding-interviews/?utm\_source=tldrwebdev  
  
Coding interviews are effective for companies despite their limitations because they help minimize the risk of hiring bad candidates. While there are downsides to coding interviews, like pressure and filtering out good candidates who don't perform well, they can be useful for assessing a candidate's problem-solving skills and their potential as a future teammate. Automated assessments aren’t as reliable as they lack human interaction and can’t always catch cheating.

|  |
| --- |
| **Adding Keyboard Shortcuts to a 24 Year Old Government Website with Userscripts (7 minute read)**  <https://wcedmisten.fyi/post/keyboard-shortcuts-userscripts/?utm_source=tldrwebdev>  This author spent a year cleaning medical device data from the FDA's 510k database website. The website's interface was outdated and lacked functionalities like fuzzy search and keyboard shortcuts. To save time and improve their workflow, the author wrote userscripts to automate repetitive actions by adding shortcuts for opening search, focusing on search input, and copying device ID. |

|  |
| --- |
| **You should be playing with GPTs at work (11 minute read)**  <https://www.lennysnewsletter.com/p/you-should-be-playing-with-gpts-at?r=2bjtip&utm_medium=ios&utm_campaign=post&utm_source=tldrwebdev>  This article provides twenty examples of how people are using custom GPTs in the workplace to make their teams more productive. It walks through how to create a custom GPT and demonstrates some interesting real-world use cases, such as figuring out internal ownership and technical dependencies, learning from past user research studies, and grading the relevance of search results. |