

WEBEX Amplified

Enabling Hybrid Work with Webex and ThousandEyes

Omer Ilyas

Omer Ilyas - Technical Marketing Engineer - oilyas@cisco.com

Table of contents

1. Lab	3
1.1 Overview - Understanding AI and Its Integration with Webex	3
1.2 Pre-Requisites and Setup	9
1.3 AI/ML Revolution Unveiled	22
1.4 Tokenization	26
1.5 Task 4: Embeddings and Vector Database	44
1.6 Task 5: Context Windows and Retrieval Augmented Generation (RAG)	68
1.7 Analyzing ThousandEyes EPA Data	76
1.8 Understanding Fine-Tuning - Large Language Models	86
1.9 Deep Dive into Quantization, LoRA and SFT	88
1.10 Task 8a - Configuring and Fine Tuning - Using llama2	100
1.11 Task 8b - Configuring and Fine Tuning - Using llama3	0
1.12 Task 8c - Configuring and Fine Tuning - Using ChatGPT	0
1.13 Language Model Merging	0
1.14 Logging Out and Ending the Lab Session	0
1.15 Conclusion	0

1. Lab

1.1 Overview - Understanding AI and Its Integration with Webex

Catch me up in a meeting

Late to the meeting?
Stepped away from the meeting? You can quickly catch up on what you missed.



Artificial Intelligence (AI) is transforming the way we work, enabling innovative solutions and enhancing productivity. Cisco is committed to innovating responsibly, with responsible AI being non-negotiable. Our approach is grounded in the principles of transparency, fairness, accountability, reliability, security, and privacy. Cisco's Responsible AI Framework, aligned with the AI Risk Management Framework, ensures that all AI initiatives undergo rigorous assessments, particularly around privacy.

Cisco's AI strategy emphasizes building ethical and trustworthy systems that mitigate risks while enhancing innovation across products and services.

Please refer to the below info for more detailed insights:

- Responsible AI is built on a foundation of privacy
- Cisco's Responsible Artificial Intelligence Principles
- Cisco's Responsible Approach to Governing Artificial Intelligence
- Cisco's Responsible Artificial Intelligence Framework

Responsible AI Framework

Cisco's goal is to provide clarity and consistency in informing users when AI is employed in our technologies

Cisco Trust Portal

trustportal.cisco.com



Your Data Belongs to You



Privacy, Security, and Human Rights by Design, Not by Chance



Trust Is Earned, Transparency Is Paramount

1.1.1 The Evolution of AI at Cisco

At Cisco, innovation is woven into the fabric of everything we do. We've been at the forefront of AI development long before it became a buzzword. Our journey spans decades, from early developments in audio and video intelligence, like echo cancellation, media resilience, to recent breakthroughs like noise removal, face recognition, and immersive experiences, just to name a few. We continue to push boundaries, enhancing AI across multiple domains to deliver immersive and intelligent experiences. Innovation in AI isn't just a trend for us—it's a longstanding commitment to excellence and progress.



What we want everyone to understand is that AI is the core fabric that powers our platforms, enabling reimagining of work, workspaces, and customer experiences. Whether through the Webex Suite, advanced devices, or contact center solutions, AI drives the seamless, intelligent, and immersive experiences that define modern collaboration. Cisco's AI-powered Webex platform is designed to transform the way we work, ensuring that every interaction is smarter, more efficient, and more personalized.



REIMAGINE WORK WITH WEBEX SUITE



REIMAGINE WORKSPACES WITH DEVICES



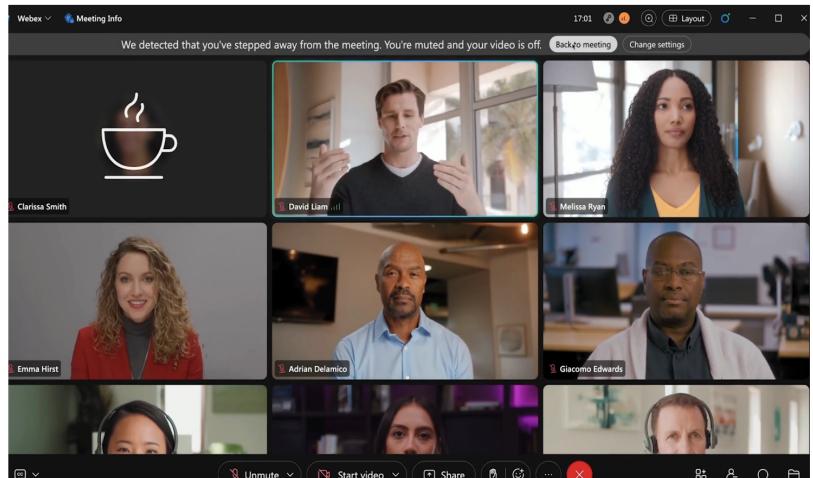
REIMAGINE CUSTOMER EXPERIENCE WITH CONTACT CENTRE

AI-powered Webex Platform

In recent years, we've expanded our platform with several new AI capabilities. Here are a few highlights:

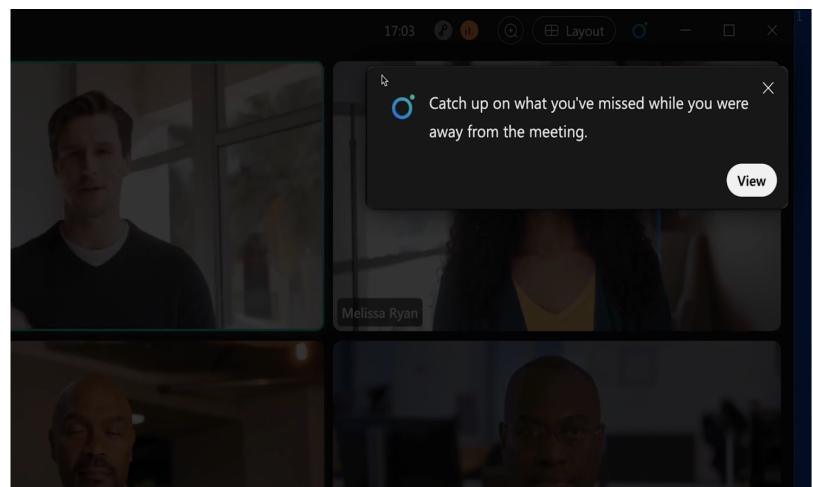
Catch me up in a meeting

Late to the meeting?
Stepped away from the meeting? You can quickly catch up on what you missed.



Catch me up in a meeting

Late to the meeting?
Stepped away from the meeting? You can quickly catch up on what you missed.



Catch me up in a meeting

Late to the meeting?
Stepped away from the meeting? You can quickly catch up on what you missed.



Translate Messages with Ease



Cinematic Meetings

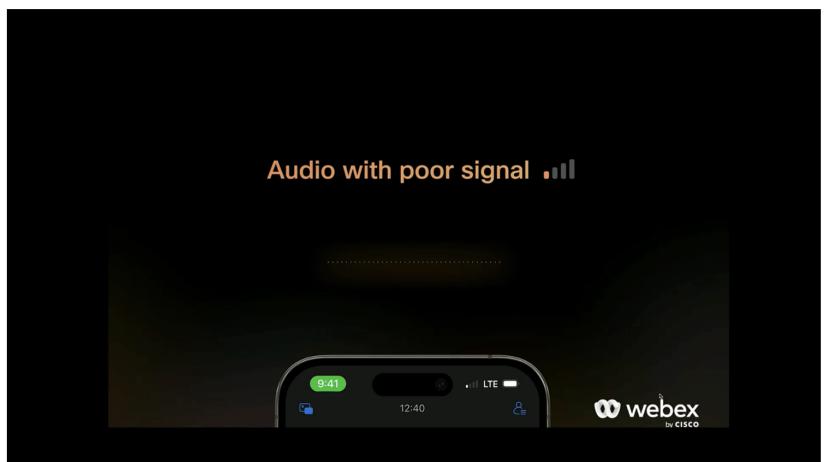
Keep everyone engaged and show the best view of people in the room from different angles through adaptive AI directed framing



Webex AI Codec

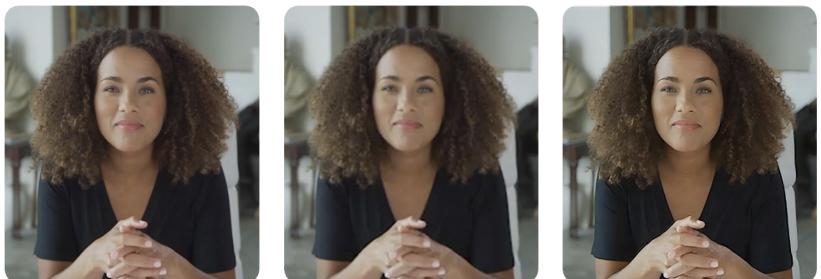
Be heard and seen clearly

New AI codec and Super resolution transform audio and video even in the poorest network connection



Be heard and seen clearly

New AI codec and Super resolution transform audio and video even in the poorest network connection



1.1.2 Summary of newly integrated AI features within our platform

Reimagine work and workspaces

Optimize for voice	Meeting Summaries
Smart Chapters	Space Summaries
Background Noise Removal	Rewrite/Translate messages
AI Codec	Multi Stream
HD Voice	Smart Relighting
Be right back	People Recognition in Rooms
Super Resolutions	<u>Slido</u> Topic Summaries
	Video Highlights

Reimagine Customer Experience

New Agent Desktop
New Supervisor Desktop
Topic Analysis
Intelligent Summary
Suggested Responses
Coaching Highlights
Agent Burnout Detection

REMINDER: Our product team is hosting AI sessions. Please review the agenda and join the ones that interest you.

1.1.3 Lab Guide Overview

In this lab guide you will go through the fundamental concepts of AI, various techniques, and how AI can be integrated with Webex to create efficient workflows.

1.1.4 Upon completion of this lab you will be able to

- Understand the basics of AI and its applications
- Learn about embedding techniques
- Explore vector databases
- Gain insights into Generative AI models
- Familiarize yourself with different AI frameworks
- Integrate AI with Webex to create seamless workflows
- Develop hands-on skills through practical exercises
- Understand Fine-tuning and Quantization
- Deploy Fine Tuning techniques

1.1.5 Prerequisites

- Basic understanding of AI concepts is helpful but not required.

1.1.6 Disclaimer

The lab design and configuration examples provided are for reference purposes only. This is a sample deployment, and not all recommended features are used or enabled optimally. For design-related questions, please contact your representative at Cisco or a Cisco partner or TME team.

1.1.7 Lab Overview - Enabling Hybrid Work with ThousandEyes

- Lab Login and Setup
- Quick AI Overview
- Configure and Access the Lab Systems
- Deploy AI Models and Techniques
- Use POSTMAN for embeddings
- Set up and Configure AI Monitoring
- Review AI Integration with Webex
- Analyze Data and Optimize Workflows
- Wrap up and End the Lab

Let's get started! Click on Task 1 - Google Collab- Accessing Google Collab and creating account.

1.2 Pre-Requisites and Setup

1.2.1 Google Collab- Accessing Google Collab and creating account

Google Colab is a free, cloud-based platform that provides a convenient environment for running notebooks. If you want to create a machine learning model but don't have a computer that can handle the workload, Google Colab is the platform for you. In our lab, we will be using Google Colab to test and run our code. However, if you have your own Python environment and prefer to run the code on your local machine, please feel free to do so.

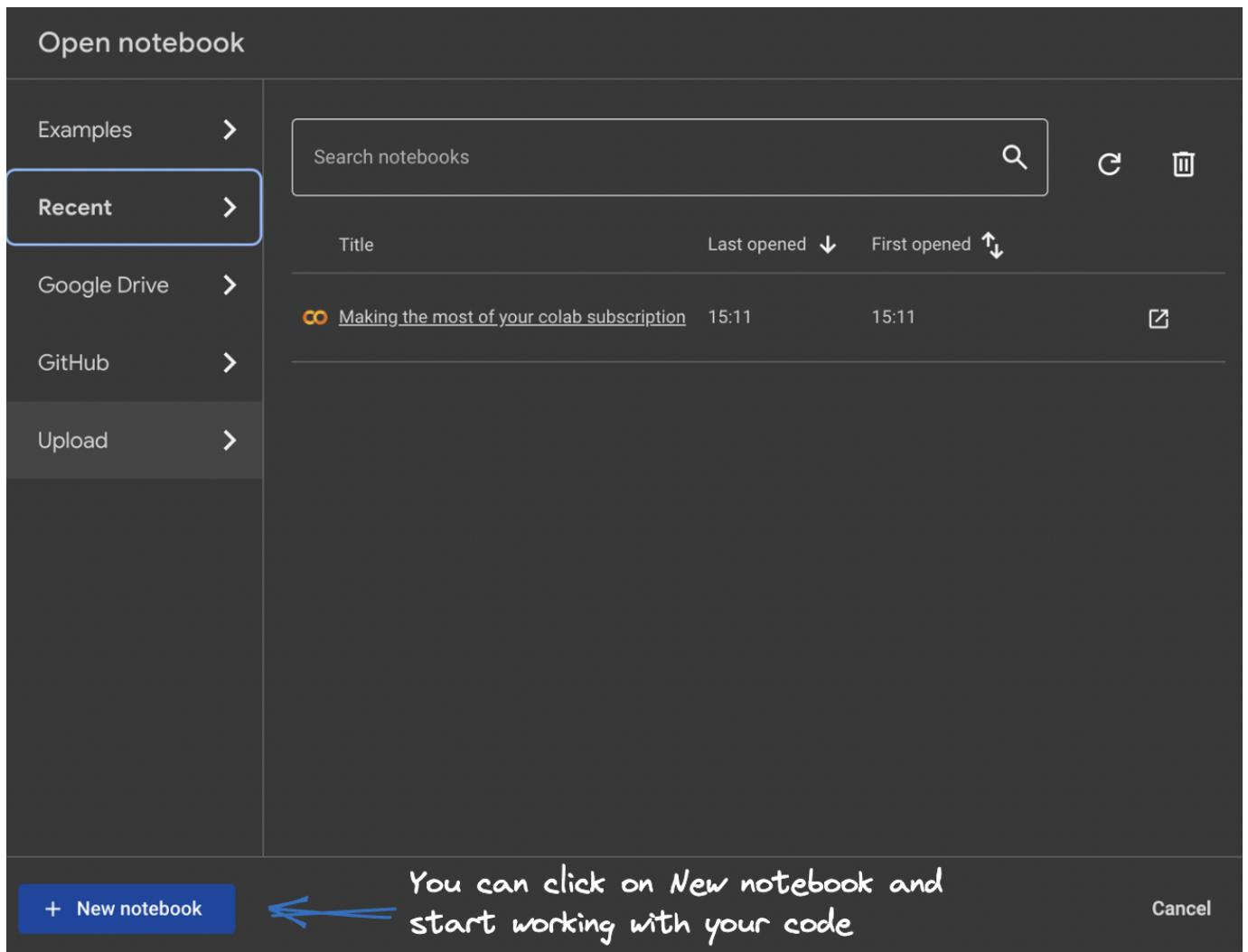
Here are some reasons why using Google Colab can be beneficial for this lab:

- Free Access to GPUs and TPUs: Google Colab offers free access to powerful GPUs and TPUs, which can significantly accelerate the training and fine-tuning of machine learning models.
- No Setup Required: With Colab, there is no need to set up your local environment. Everything runs in the cloud, which saves time and avoids configuration issues.
- Easy Collaboration: Colab notebooks can be easily shared and collaborated on with team members, making it an ideal tool for collaborative projects.
- Integration with Google Drive: Colab integrates seamlessly with Google Drive, allowing you to save and manage your work conveniently.
- Pre-installed Libraries: Many popular machine learning libraries, including TensorFlow and PyTorch, come pre-installed in Colab, making it easy to start working on your projects immediately.

1.2.2 Getting Started With Google Colab

To start working with Google Collaboratory Notebook you first need to log in to your Google account, then go to this link [Google Colab](#)

- Create a new Jupyter Notebook



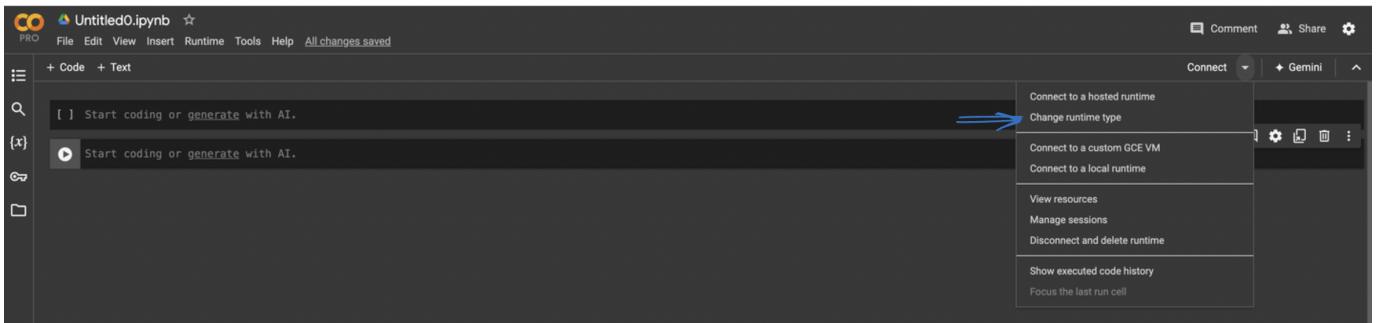
- On creating a new notebook, it will create a Jupyter notebook with Untitled0.ipynb and save it to your google drive in a folder named Colab Notebooks. Now as it is essentially a Jupyter Notebook, all commands of Jupyter Notebooks will work here.



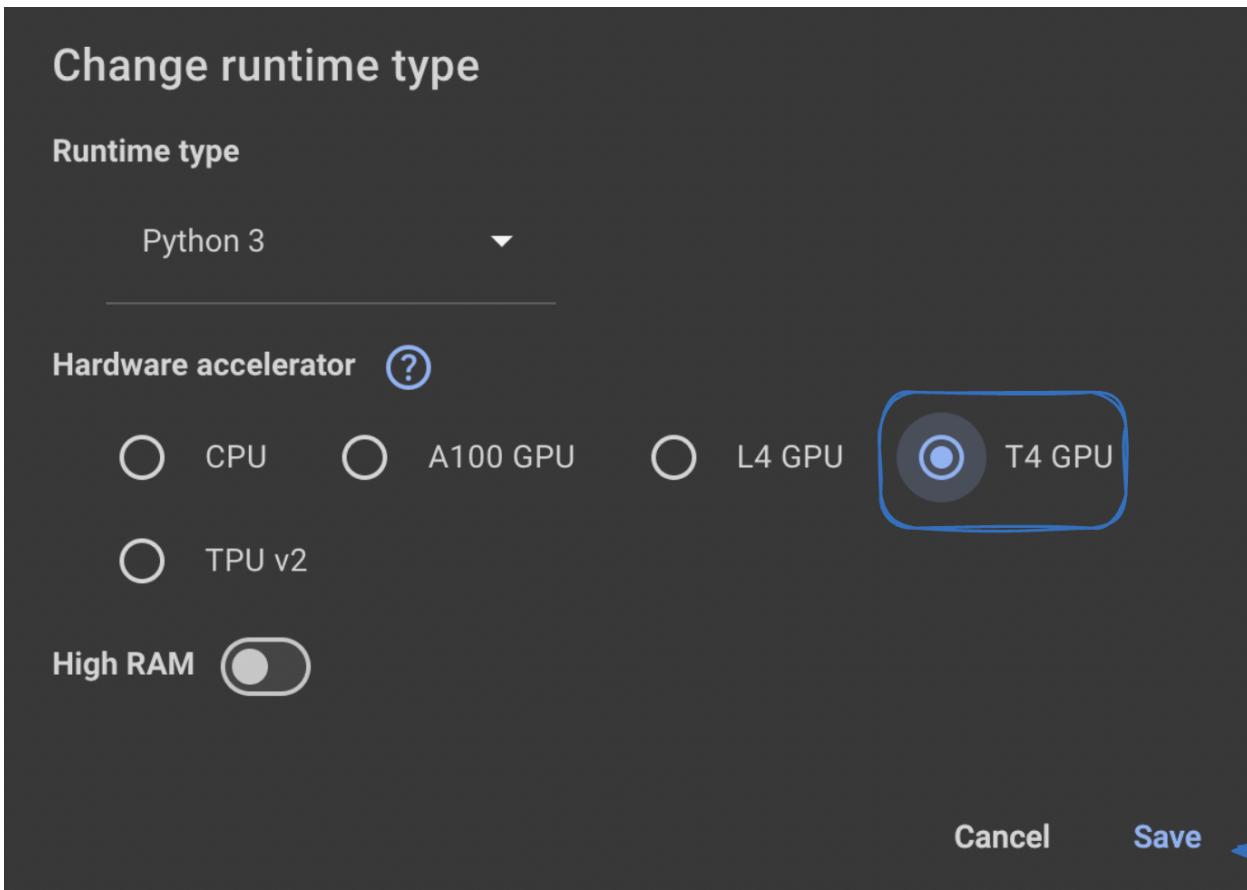
- There might be times when we need to fine-tune models or perform specific tasks that require changing the runtime environment in Colab. Google Colab offers different runtime environments that can be selected based on your requirements:
- Python Versions: You can select between different versions of Python (e.g., Python 2 or Python 3) depending on the compatibility of the code and libraries. We will be using Python3 for our lab.
- Hardware Accelerators: Colab provides access to hardware accelerators, which can be particularly useful for intensive computations. You can choose between:

```
None: No hardware acceleration, suitable for basic tasks.  
GPU: Accelerate your computations with a Graphics Processing Unit.  
TPU: Use a Tensor Processing Unit for even faster performance, especially beneficial for deep learning tasks.
```

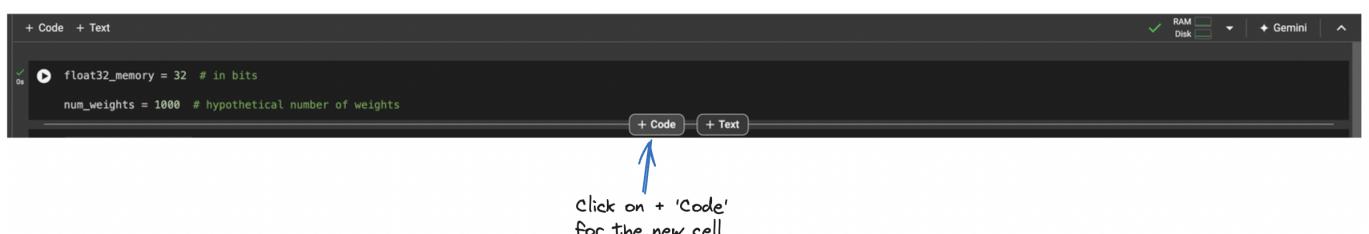
- Change Runtime Environment: Click the “Runtime” dropdown menu at the top of the Colab interface.



- Select “Change runtime type”: This will open a dialog box where you can configure the runtime environment.
- Select Python Version: Choose Python 3 from the “Runtime type” dropdown menu.
- Select Hardware Accelerator: From the “Hardware accelerator” dropdown menu, choose GPU, or TPU based on your needs.



- Save Settings: Click “Save” to apply the changes.
- New Cell: Whenever you want to copy the code in Google Colab and run it, be sure to click on + Code to add a new code cell.



- Execute Code: Click the play button to the left of the code, or use the keyboard shortcut "Command/Ctrl+Enter" while the cell is selected.



1.2.3 Notes: On GPU and TPU Access:

While Google Colab offers free access to GPUs and TPUs, there are limitations. For more consistent access to high-performance GPUs and TPUs, you might need to subscribe to Colab Pro or Colab Pro+ accounts. These paid plans provide priority access to better hardware, longer runtimes, and more memory.

1.2.4 Using Huggingface Hub to share our Datasets

In this lab, we will be utilizing the Hugging Face Hub to load our custom datasets. Hugging Face provides an extensive repository of datasets that can be easily integrated into your machine learning workflows. For the purposes of this lab, we will demonstrate how to access/upload and use our custom datasets effectively.

However, when fine-tuning models in your own work environment, especially if you are using private data, there are important considerations to keep in mind:

- Private Datastores: If you are working with proprietary or sensitive data, it is crucial to use your organization's secure datastores. Ensure that all data handling complies with your organization's data privacy policies and regulations.
- Hugging Face Datasets: If you prefer to use Hugging Face for dataset storage and management, make sure to mark your datasets as private. This setting ensures that your data cannot be accessed by anyone outside your organization, maintaining the confidentiality and integrity of your information. Please refer to Huggingface documentation for more info.

Few more Consideration

- Upload Dataset: When uploading your dataset to Hugging Face, choose the appropriate privacy settings. You can set your dataset to private during the upload process.
- Check Permissions: Regularly review and manage the permissions of your datasets to ensure they remain private and secure.
- Collaborator Access: If you need to share the dataset with specific team members, use the Hugging Face interface to grant access to trusted collaborators only.

By following these guidelines, you can ensure that your data remains secure while leveraging the powerful tools and resources provided by Hugging Face. This approach not only enhances your workflow efficiency but also upholds the best practices in data security and privacy.

1.2.5 Accessing Hugging Face Hub and creating account

Hugging Face can be accessed by browsing to huggingface.co

The screenshot shows the Hugging Face Hub homepage. At the top, there's a search bar with placeholder text "Search models, datasets, users...". Below the search bar, there are navigation links for "Models", "Datasets", "Spaces", "Posts", "Docs", "Solutions", "Pricing", "Log In", and "Sign Up". A banner at the top left says "NEW! AI Tools are now available in HuggingChat". The main content area features a large yellow emoji of a smiling face. Below it, the text "The AI community building the future." is displayed in a large, bold, white font. To the right, there's a sidebar with categories like "Tasks", "Libraries", "Datasets", "Languages", "Licenses", and "Other". Under "Tasks", there are sections for "Text-to-Image", "Text-to-Video", "Document Question Answering", "Computer Vision", "Natural Language Processing", "Audio", "Tabular", and "Reinforcement Learning". On the right side, there's a list of models with their names, descriptions, and statistics. Some examples include "meta-llama/Llama-2-70b", "stabilityai/stable-diffusion-xl-base-0.9", and "tiiuae/falcon-40b-instruct".

Signing up

- Browse to Hugging Face home page and click on Sign up. Follow the instructions as per below images

The screenshot shows the "Join Hugging Face" sign-up form. It has a yellow emoji at the top. The form asks for an "Email Address" (with placeholder "omer.lyas@boldbetz.com") and a "Password" (with placeholder "*****"). Below the password field are three validation rules: "Must contain at least 8 characters", "Must contain uppercase, lowercase letters, and numbers", and "If less than 12 characters, must contain a special character". At the bottom of the form is a "Next" button and a link "Already have an account? Log in". Blue arrows point from the text "Follow the instructions as per below images" to the "Email Address" field, the "Password" field, and the "Next" button.

Complete your profile

One last step to join the community

Username

Full name

Avatar (optional)

Twitter username (optional)

Upload file

Twitter account

GitHub username (optional)

LinkedIn profile (optional)

GitHub username

LinkedIn profile

Homepage (optional)

Homepage

AI & ML interests (optional)

AI & ML interests

I have read and agree with the [Terms of Service](#) and the [Code of Conduct](#)

Create Account

- Please check your email address for a confirmation link and click to verify your account

Your email address has been verified successfully.

Join an organization

Hugging Face is way more fun with friends and colleagues!

Being part of an organization lets you:

- >Show your university, lab or company work to the Hugging Face community
- Collaborate on private datasets, models and spaces
- Subscribe to a paid plan and use Hugging Face hosted training and inference services

Search for an organization

or Create a new organization

- Organization Creation (Optional): While you can upload datasets and fine-tune models directly on Hugging Face without creating an organization, you have the option to create an organization on Hugging Face. This can be particularly useful for team collaboration, as it allows you to upload all your datasets and models in one centralized location.

Optional Step

New Organization

Complete your organization profile

Organization Username: WebexOne

Organization type: Classroom

Homepage (optional): Homepage

Logo (optional): Upload a file

Organization Full name: WebexOne

GitHub username (optional): GitHub alias

Twitter username (optional): Twitter account

AI & ML interests (optional): AI & ML interests

Create Organization

- At this stage, you will see no models or datasets created under your account.

The screenshot shows the Hugging Face Hub organization page for 'WebexOne'. At the top, there's a search bar and a navigation bar with links for Models, Datasets, Spaces, Posts, Docs, Solutions, Pricing, and a profile icon. Below the header, it says 'Classroom' and 'Upgrade to Enterprise'. On the left, there's a sidebar with sections for AI & ML interests (None defined yet), Team members (1), and a profile picture. The main content area has a heading 'No organization card' with a sub-section 'Customize this page and let people know more about your organization by creating an organization card!' and a 'Create a Card' button. Below this, there are sections for Models (None yet) and Datasets (None yet). Two blue arrows point from the text 'No models or Datasets created yet' to the 'None yet' entries for Models and Datasets.

No organization card

Customize this page and let people know more about your organization by creating an organization card!

Create a Card

Models None yet

Datasets None yet

No models or Datasets created yet

The screenshot shows the user profile menu on the right side of the header. It includes a profile picture, the name 'Webex', and sections for Notifications (Inbox (0)), New Model, New Dataset, New Space, and New Collection. There are also links for Create organization, Settings, and Sign Out. A blue arrow points from the text 'Access Your Models and Datasets' to the 'Profile' section of the menu.

Profile

Webex

Notifications
Inbox (0)

New Model

New Dataset

New Space

New Collection

Create organization

Settings

Sign Out

Access Your Models and Datasets: The same can be accessed by clicking your profile picture on the top right corner of the Hugging Face website. This will take you to your personal dashboard where you can view and manage your models and datasets.

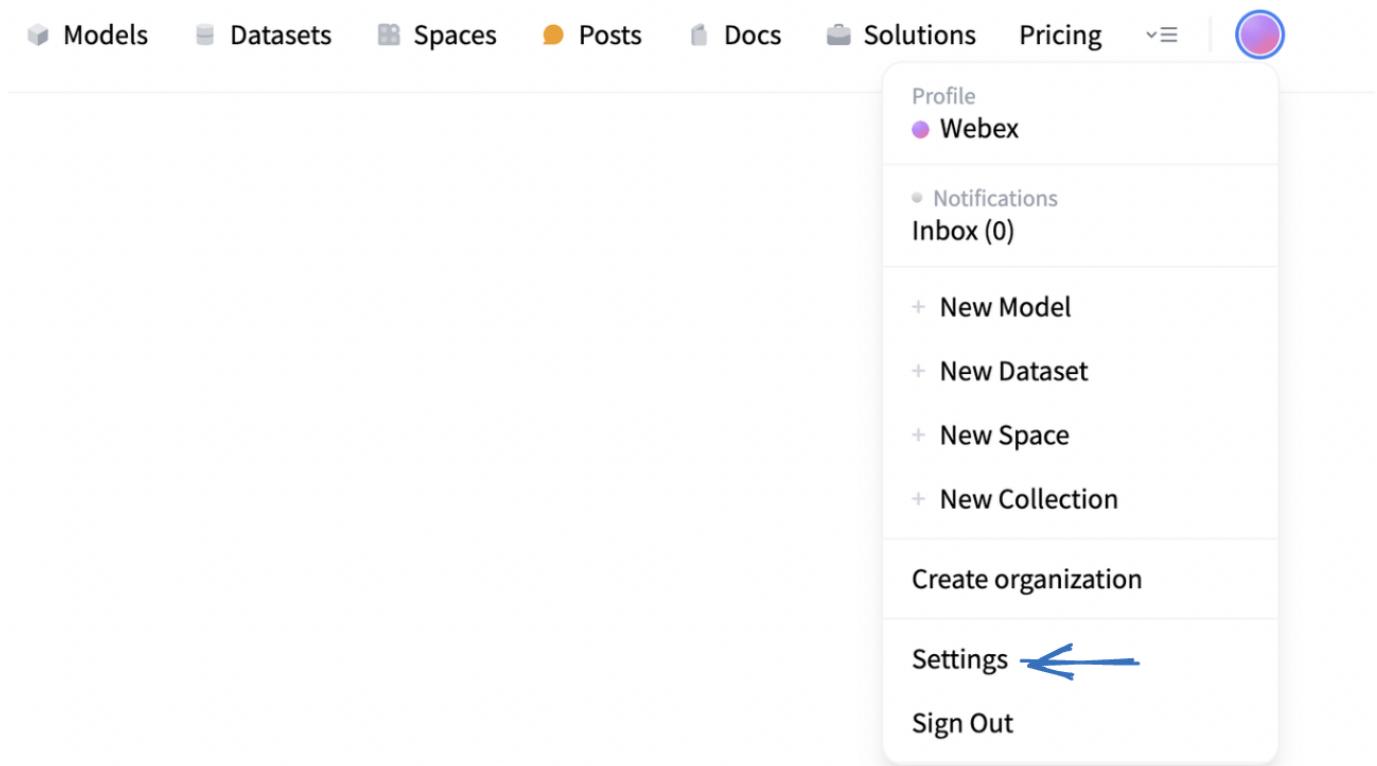
Hugging Face API Keys

Create an API Key: As we will be uploading our datasets to the Hugging Face Hub, we need to create an API key for our account. This API key will be used to authenticate and interact with the Hugging Face services programmatically.

you can browse to [huggingface API Key](#)

or

Click on your profile picture > Settings > Access Tokens



The screenshot shows the 'Access Tokens' section of the Hugging Face Hub settings. It includes a 'User Access Tokens' section with a note about token authentication and a 'Create new token' button. The 'Access Tokens' link in the sidebar is circled with a blue circle labeled '1'. A blue arrow points from the 'Create new token' button to another circled '2'.

Under the "Access Tokens" section, click on "Create new token." You will see options to select the token type and provide a token name. For example, you might name your token "Webexone" and select the appropriate permissions.

- fine-grained: tokens with this role can be used to provide fine-grained access to specific resources, such as a specific model or models in a specific organization. This type of token is useful in production environments, as you can use your own token without sharing access to all your resources.
- read: tokens with this role can only be used to provide read access to repositories you could read. That includes public and private repositories that you, or an organization you're a member of, own. Use this role if you only need to read content from the Hugging Face Hub (e.g. when downloading private models or doing inference).
- write: tokens with this role additionally grant write access to the repositories you have write access to. Use this token if you need to create or push content to a repository (e.g., when training a model or modifying a model card).

As we have a lab environment we will be using the "Write" permission. This token will have read and write access to all your resources and can make calls to inference API on your behalf, as shown in the image below.

Create new Access Token

Token type

Fine-grained Read Write ← 1

! This cannot be changed after token creation.

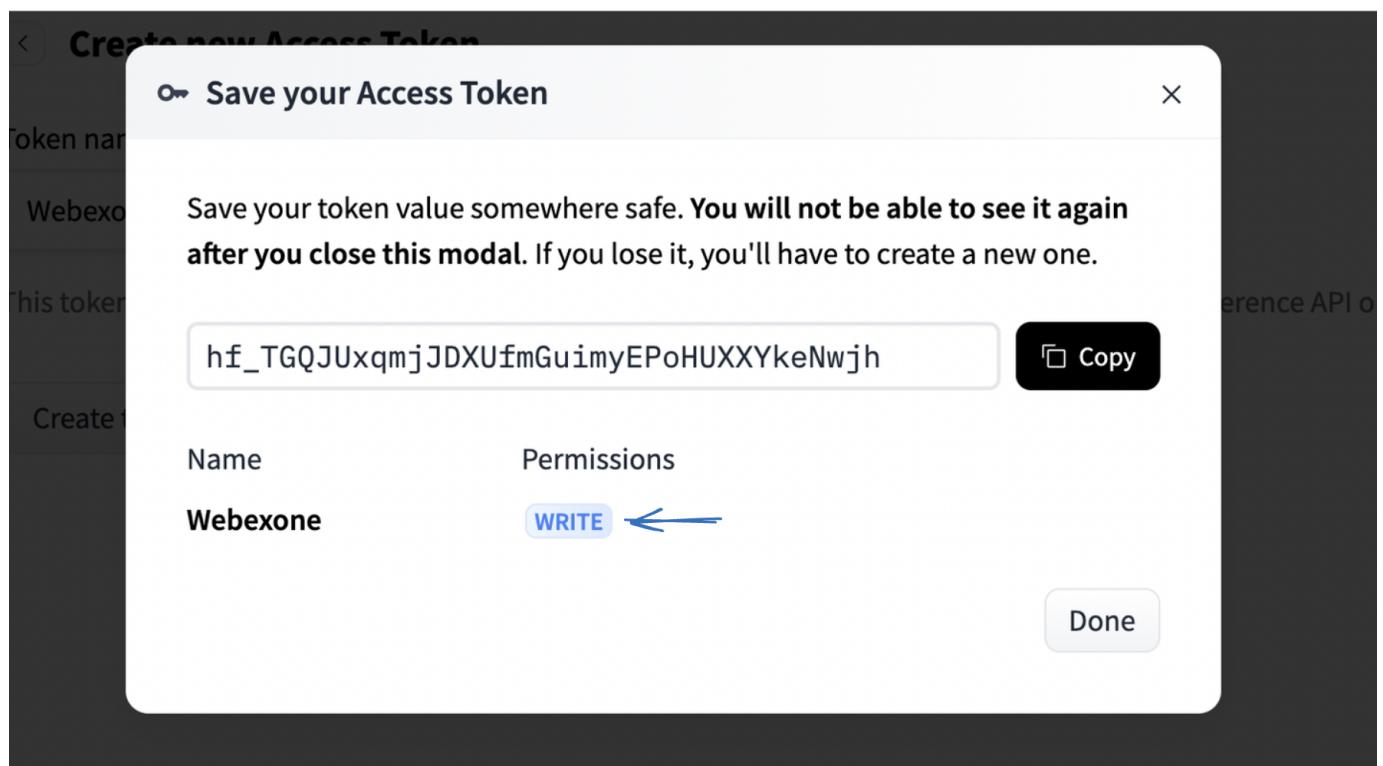
Token name

Webexone

This token has read and write access to all your and your orgs resources and can make calls to inference API on your behalf.

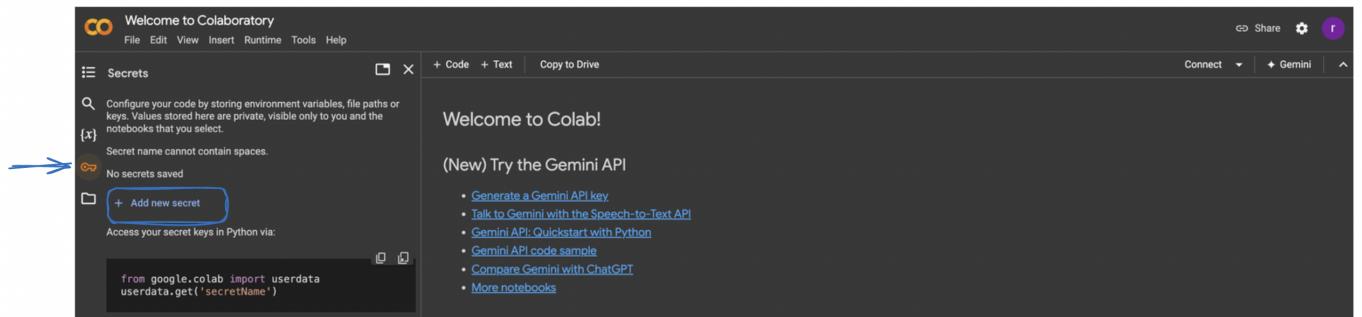
Create token ← 2

Save and Secure the Token: Once the token is generated, save it securely. This token will be required for accessing and managing your datasets via the API. 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



1.2.6 Accessing Hugging Face API in Google Colab

- Open the Google Colab notebook and navigate to the new “Secrets” section in the sidebar.



- Click on “Add a new secret.” Enter the name example: HF_TOKEN and value of the secret. Note: The name is permanent once set.
- The list of secrets is global across all your notebooks.
- Use the “Notebook access” toggle to grant or revoke access to a secret for each notebook.

The screenshot shows the 'Secrets' sidebar in Google Colab. It includes a search icon, a note about secret names, and a table for managing secrets. The table has columns for 'Notebook access', 'Name', 'Value', and 'Actions'. A specific row for 'HF_TOKEN' is highlighted with a blue border. Handwritten annotations include arrows pointing to the 'Add new secret' button and the 'Value' field, with text below them: 'Give it a meaningful name' and 'The secret from Huggingface you created earlier'.

Notebook access	Name	Value	Actions
<input checked="" type="checkbox"/>	HF_TOKEN	hf_TGQJUxq [REDACTED]	

Give it a
meaningful
name

The secret from Huggingface
you created earlier

Optional Steps below

Incorporating Secrets into Your Code - We will use it later in our lab

- To use a secret in your notebook, use the following code snippet

```
from google.colab import userdata  
my_secret_key = userdata.get('HF_TOKEN')
```

- Replace with your secret's name.

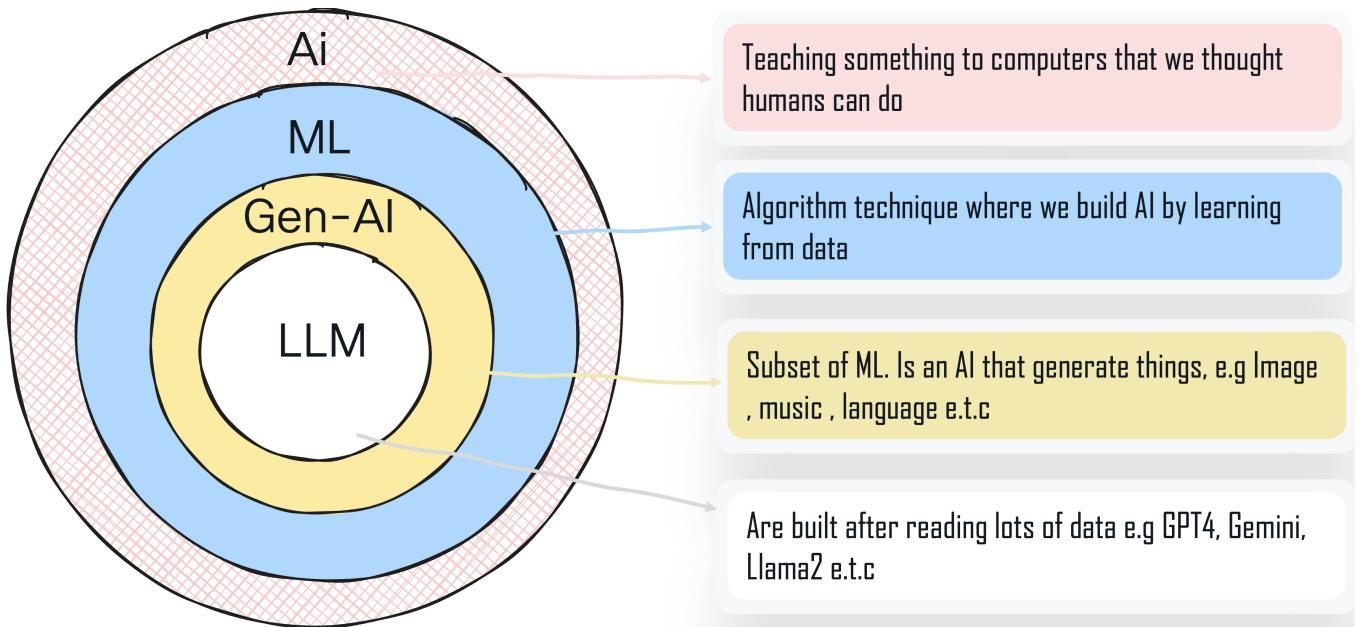
Using Secrets as Environment Variables - Optional Step , we will use it later in our lab

- For Python modules requiring API keys as environment variables, use the below snippet:

```
# Import Colab Secrets userdata module  
  
from google.colab import userdata  
import os  
  
# Set other API keys similarly  
os.environ["HF_TOKEN"] = userdata.get('HF_TOKEN')
```

1.3 AI/ML Revolution Unveiled

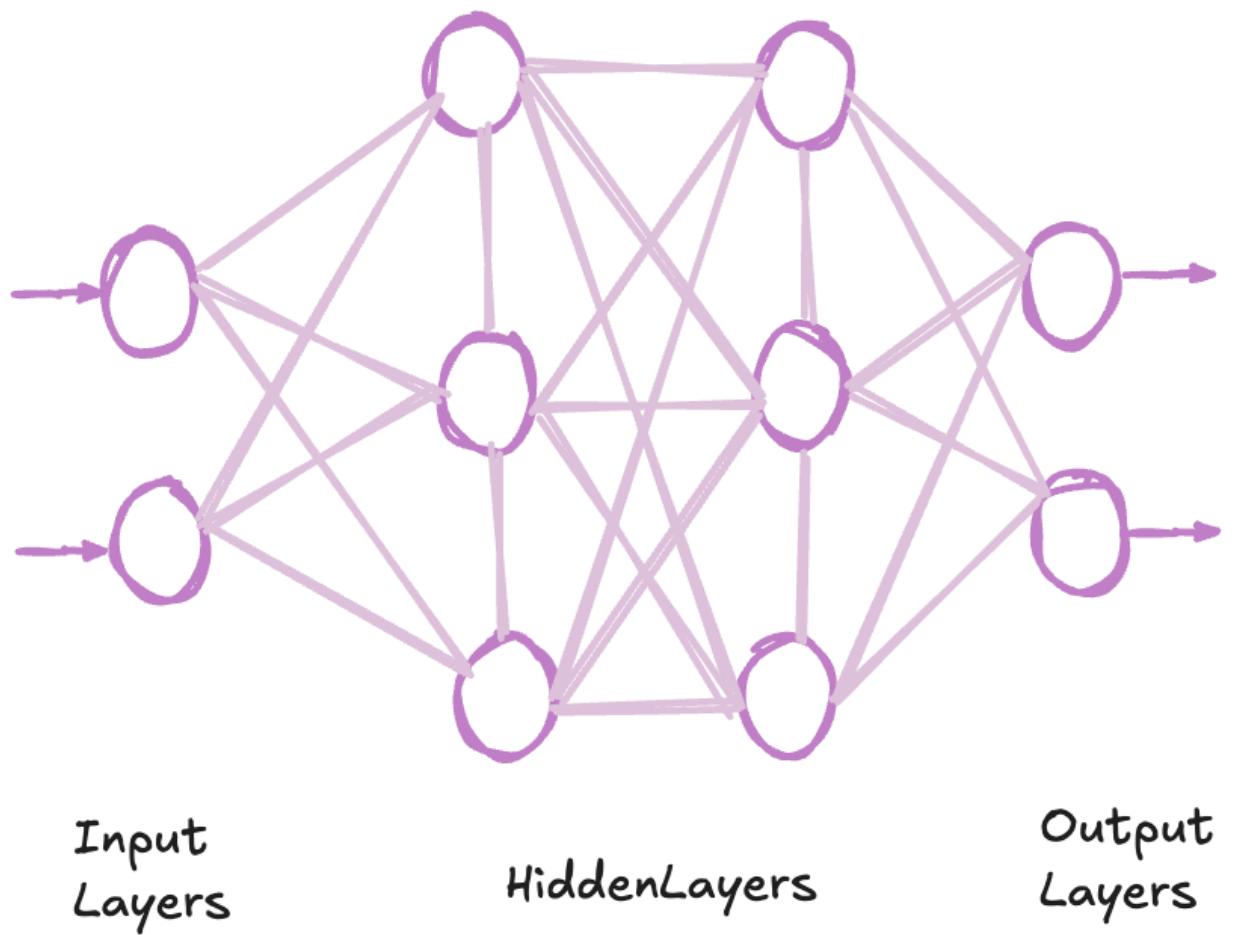
1.3.1 Different Types of AI



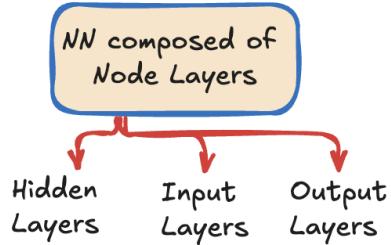
Remember: Subset of AI is ML and kind of ML is Gen-AI . LLM e.g GPT are subset of Gen-AI

1.3.2 Neural Network

Artificial Neural Network



Note: The way nodes are connected called weights



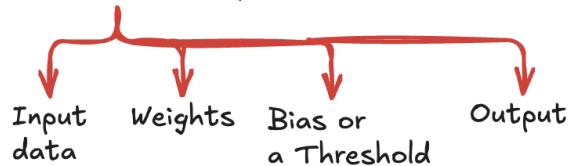
These neural network reflects the behaviour of human brain allowing computers to recognize patterns and solve common problems

Each neuron has its own Linear Regression Model

Linear Regression is the mathematical model that is used to predict future events

The weights of the connections between the nodes determine how much influence each input has on output

Each **Node** is composed of

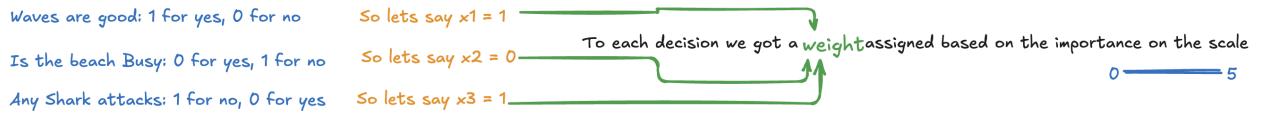


Data Passed between Nodes called FFN

The data is passed in the neural network from one layer to another which is known as feed forward propagation

Example: Lets see how a single node in NN looks like when deciding if you wanna go surfing. The decision to go or not will be our predictive outcome or lets call it our $y\hat{}$

Assume there are 3 factors influencing our decision:



Lets assign weights to our decisions:

So Waves are good we will assign a weight $w_1 = 5$

Beach not busy not of great importance, $w_2 = 2$

No Shark attacks, lets give it a weight $w_3 = 4$

Overall Result:

$$\begin{array}{lll} x_1 = 1 & x_2 = 0 & x_3 = 1 \\ w_1 = 5 & w_2 = 2 & w_3 = 4 \end{array}$$

We can plug these values into our formulae to get the desired output:

$$y\hat{=} (1*5) + (0*2) + (1*4)$$

$$y\hat{=} 9 - 3 = 6$$

|
Threshold or Bias

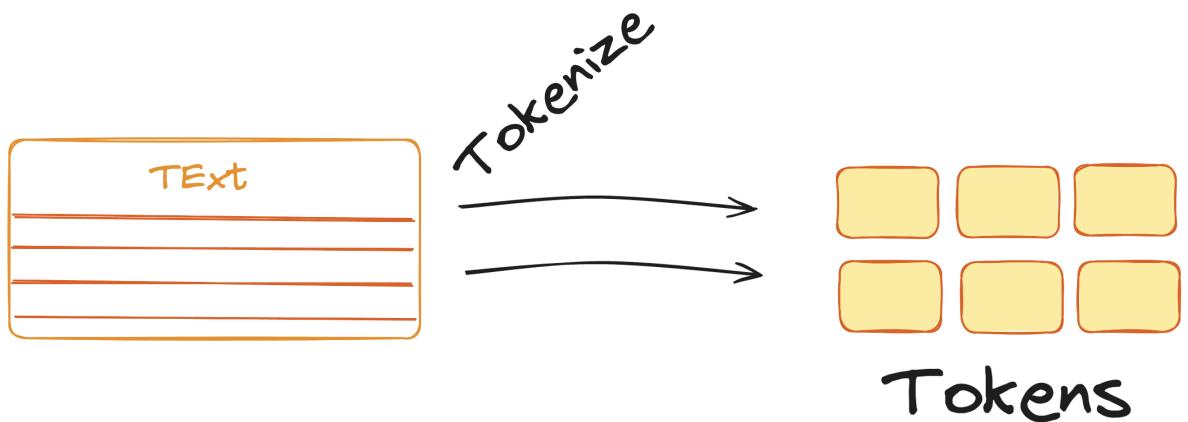
As 6 is greater than 0 so the output of this node is 1. We are going surfing

We can always adjust weights to get different outcomes

Note: Neural network rely on training data to learn and improve their accuracy over time. We used supervised learning to train the algorithm. There are multiple types of Neural network other than the Feed forward network that we have defined here. Example: CNN: Convolutional Neural Network - Unique architecture for identifying patterns like image recognition, or RNN: Recurrent neural network, that uses time series data to make prediction about future events like sales forecasting.

1.4 Tokenization

What is Tokenization and why its needed?



- Language models have a limit on how much text they can handle at once, known as their context window. While these limits are expanding, research shows that LLM's often perform better when provided with less, but more relevant, information. However, selecting the most relevant information is straightforward for humans but challenging for computers.

A common approach to manage large amounts of data is to break it down into smaller, more manageable parts a process often referred to as Tokenization or Chunking. Tokenization is a key step in this process, where raw text is divided into smaller units, called tokens, which can then be processed by a neural network.

Tokenization

Different ways to Tokenize

Word-Based

Welcome

to

Cisco

Character

W e l c o m e

t o c i s c o

Sentence

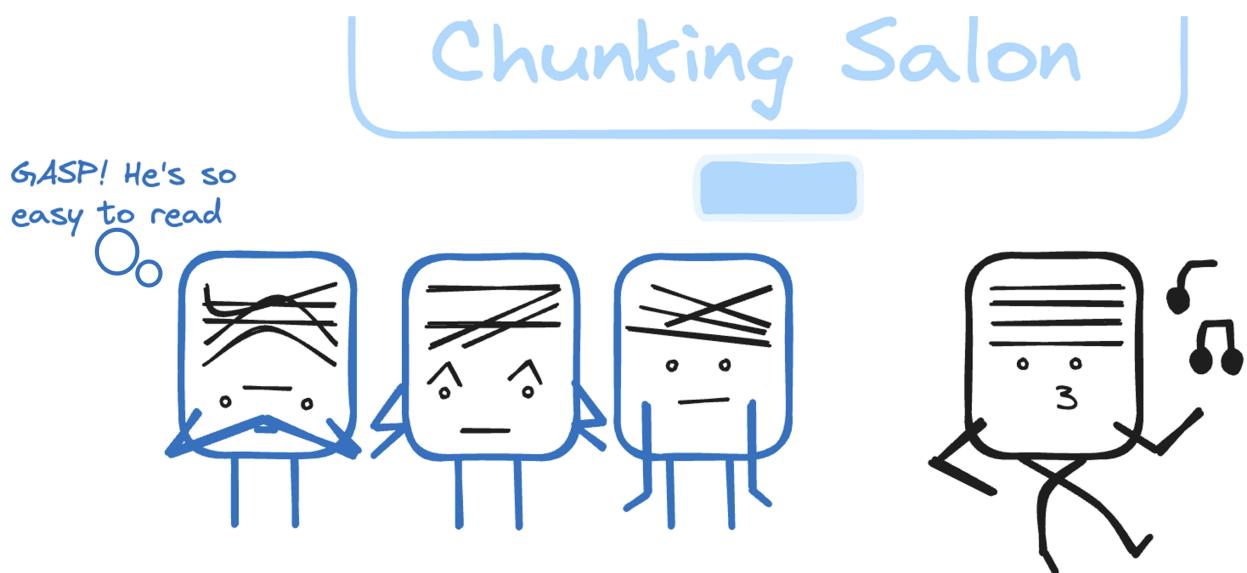
Welcome to Cisco.

In order to do this you need to pick a chunk strategy. Just to name a few:

- Word-Based: A simple and straightforward method that most of us would propose is to use word-based tokens, splitting the text by spaces.
- Character based tokenization: Individual words are considered as tokens . Lot of computing resources needed as now e.g for a 3 word sentence where you might need 3 tokens now 15 – 20 tokens needed
- Sentence Based: We need a .(fullstop) for it to work

Note: We've all heard of GPT and OpenAI. They utilize a tokenization method called Byte Pair Encoding (BPE), which is a middle ground between word-based and character-based tokenization. In BPE, words are broken down into smaller character sequences that the model encountered during training, allowing it to make informed predictions.

1.4.1 Why we need Tokenization or Text Splitting?



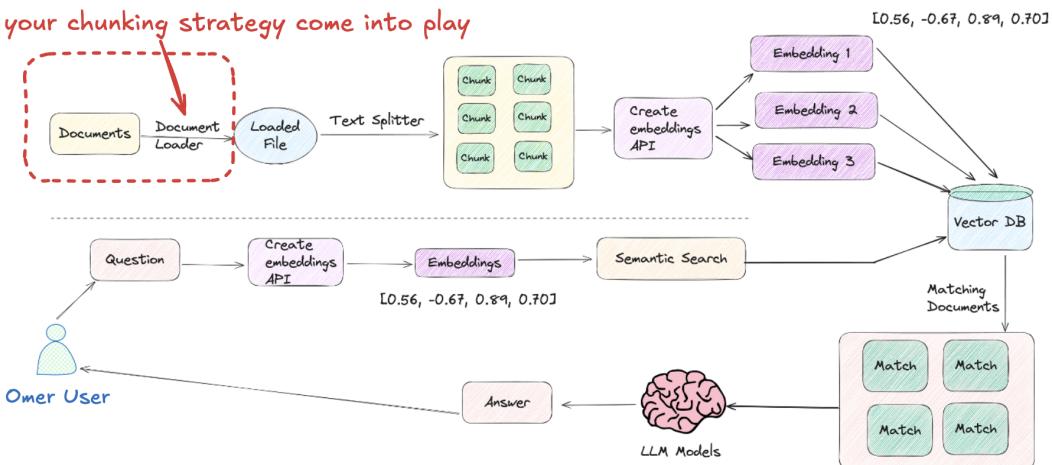
Historically, applications perform better when they are provided with your own data. However, you can't feed unlimited data to your LLMs due to two key limitations:

- Context window limit: LLMs have a finite context window for processing data.
- Signal-to-noise ratio (SNR): LLMs perform better when the SNR is high, meaning the information provided is useful, relevant, and clear. Clear, unambiguous instructions help the model deliver more accurate and detailed results, while ambiguous or complex input can lead to less accurate or incomplete outputs.

As noted, chunking or splitting refers to breaking your data into smaller, manageable pieces.

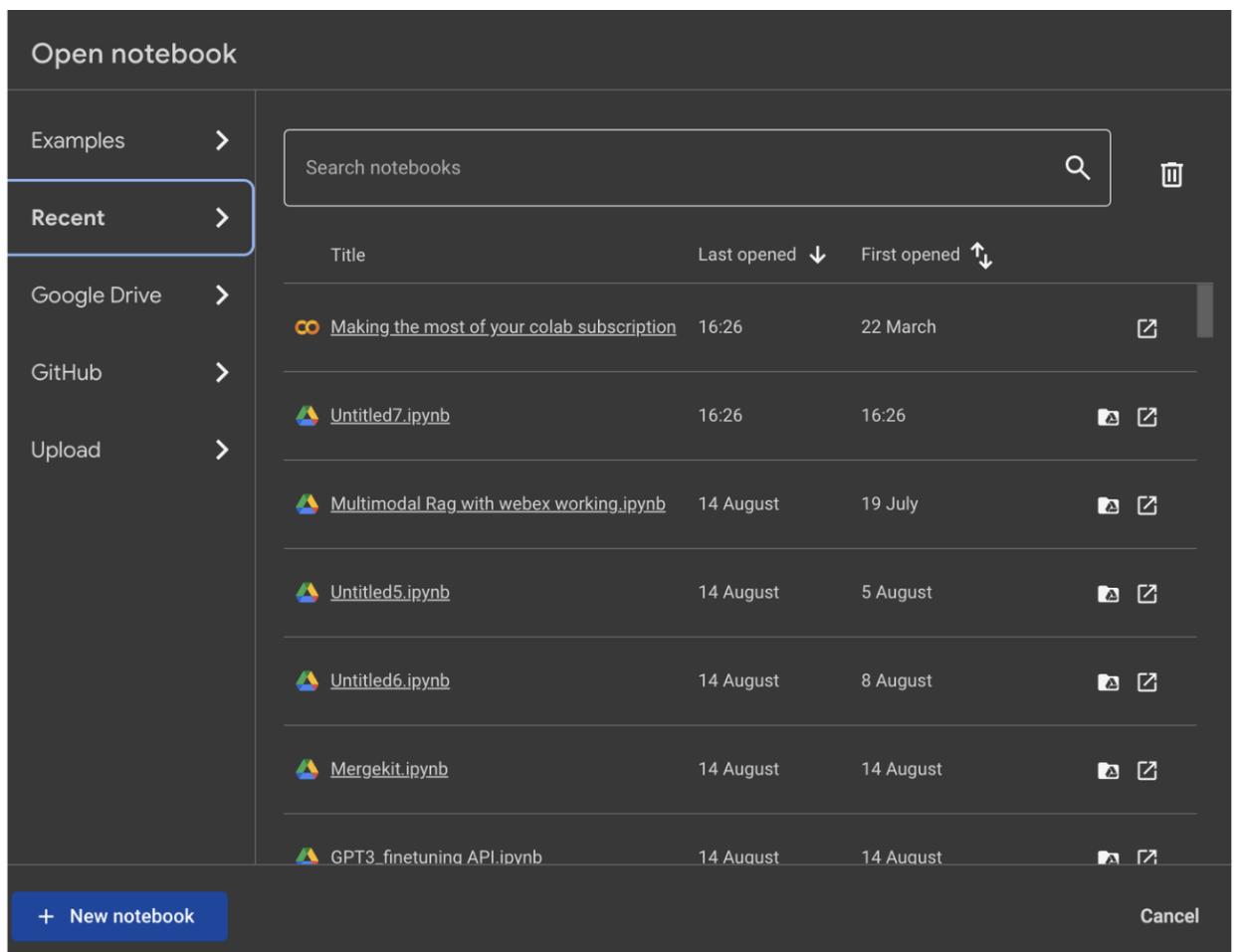
AI Full Stack Frame-work

That's where your chunking strategy come into play

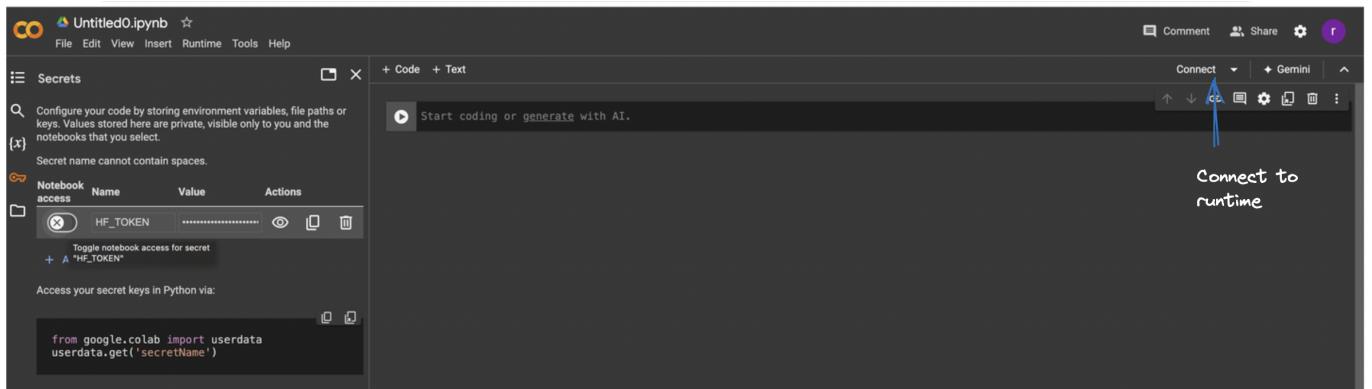


1.4.2 Task 1: Log into the Lab Environment

- Open Google Colab and create a new notebook. Click on "File" > "New notebook". Please refer to the following section to create Google Colab account.



- Make sure you are connected to a runtime. For this task, you can use the CPU as the runtime environment.



Reminder: Whenever you want to copy the code in Google Colab and run it, be sure to click on + Code to add a new code cell.



Reminder: Click the play button to the left of the code, or use the keyboard shortcut "Command/Ctrl+Enter" while the cell is selected.



Manual testing for Chunking

We will create Chunks of 35 characters, so first 35 characters as chunk 1 , next 35 characters chunk2 and so on

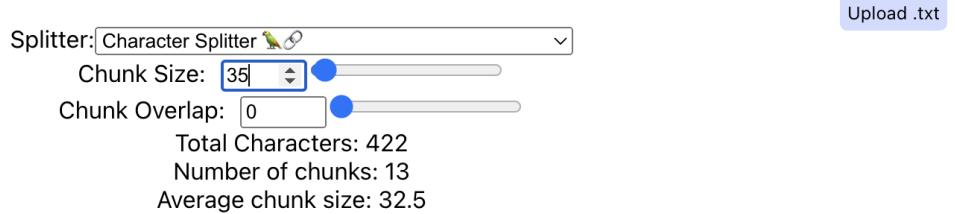
```

1 # Manual Splitting
2 text = """WebexOne is an annual in-person and virtual event that takes place over four days. It's an event focused on AI collaboration and customer
3 experience.
4 It features a range of activities such as insightful breakout sessions, technical training courses, hands-on labs, inspiring keynotes, epic
5 entertainment, a solutions showcase and expo, customer awards, meet the experts, 1:1 executive meetings, a partner program, and more!"""
6
7 # Create a list that will hold your chunks
8 chunks = []
9
10 chunk_size = 35
11
12 # Run through the a range with the length of your text and iterate every chunk_size you want
13 for i in range(0, len(text), chunk_size):
14     chunk = text[i:i + chunk_size]
15     chunks.append(chunk)

print(chunks)

```

WebexOne is an annual in-person and virtual event that takes place over four days. It's an event focused on AI collaboration and customer experience. It features a range of activities such as insightful breakout sessions, technical training courses, hands-on labs, inspiring keynotes, epic entertainment, a solutions showcase and expo, customer awards, meet the experts, 1:1 executive meetings, a partner program, and more!



WebexOne is an annual in-person and virtual event that takes place over four days. It's an event focused on AI collaboration and customer experience. It features a range of activities such as insightful breakout sessions, technical training courses, hands-on labs, inspiring keynotes, epic entertainment, a solutions showcase and expo, customer awards, meet the experts, 1:1 executive meetings, a partner program, and more!

Note: The text contained a total of 422 characters, divided into 13 chunks. Problem with the above chunking technique is that it got split at 'r'. How we know when to chunk . Before we look into that. Lets look into Langchain Splitter

LangChain Text Splitter

- Let's retrieve the langchain library from the Python Package. In the example below, we'll configure the chunk_overlap parameter, which ensures that our chunks are blended together—meaning the end of chunk 1 will overlap with the beginning of chunk 2.

```
1 !pip install langchain
```

```
Collecting langchain
  Downloading langchain-0.2.14-py3-none-any.whl.metadata (7.1 kB)
Requirement already satisfied: PyYAML>=5.3 in /usr/local/lib/python3.10/dist-packages (from langchain) (6.0.2)
Requirement already satisfied: SQLAlchemy<3,>=1.4 in /usr/local/lib/python3.10/dist-packages (from langchain) (2.0.32)
Requirement already satisfied: aiohttp<4.0.0,>=3.8.3 in /usr/local/lib/python3.10/dist-packages (from langchain) (3.10.2)
Requirement already satisfied: async-timeout<5.0.0,>=4.0.0 in /usr/local/lib/python3.10/dist-packages (from langchain) (4.0.3)
Collecting langchain-core<0.3.0,>=0.2.32 (from langchain)
  Downloading langchain_core-0.2.33-py3-none-any.whl.metadata (6.2 kB)
Collecting langchain-text-splitters<0.3.0,>=0.2.0 (from langchain)
  Downloading langchain_text_splitters-0.2.2-py3-none-any.whl.metadata (2.1 kB)
Collecting langsmith<0.2.0,>=0.1.17 (from langchain)
  Downloading langsmith-0.1.99-py3-none-any.whl.metadata (13 kB)
Requirement already satisfied: numpy<2,>=1 in /usr/local/lib/python3.10/dist-packages (from langchain) (1.26.4)
Requirement already satisfied: pydantic<3,>=1 in /usr/local/lib/python3.10/dist-packages (from langchain) (2.8.2)
Requirement already satisfied: requests<,>=2 in /usr/local/lib/python3.10/dist-packages (from langchain) (2.32.3)
Collecting tenacity!=8.4.0,<9.0.0,>=8.1.0 (from langchain)
  Downloading tenacity-8.5.0-py3-none-any.whl.metadata (1.2 kB)
Requirement already satisfied: aiohttp<4.0.0,>=3.8.3->langchain (2.3.5)
Requirement already satisfied: aiosignal<1.2.1 in /usr/local/lib/python3.10/dist-packages (from aiohttp<4.0.0,>=3.8.3->langchain) (1.3.1)
Requirement already satisfied: attrs>=17.3.0 in /usr/local/lib/python3.10/dist-packages (from aiohttp<4.0.0,>=3.8.3->langchain) (24.2.0)
Requirement already satisfied: frozenlist>=1.1.1 in /usr/local/lib/python3.10/dist-packages (from aiohttp<4.0.0,>=3.8.3->langchain) (1.4.1)
Requirement already satisfied: multidict<7.0,>=4.5 in /usr/local/lib/python3.10/dist-packages (from aiohttp<4.0.0,>=3.8.3->langchain) (6.0.5)
Requirement already satisfied: yarl<2.0,>=1.0 in /usr/local/lib/python3.10/dist-packages (from aiohttp<4.0.0,>=3.8.3->langchain) (1.9.4)
Collecting jsonpatch<2.0,>=1.33 (from langchain-core<0.3.0,>=0.2.32->langchain)
  Downloading jsonpatch-1.33-py3-none-any.whl.metadata (3.0 kB)
Requirement already satisfied: packaging<25,>=23.2 in /usr/local/lib/python3.10/dist-packages (from langchain-core<0.3.0,>=0.2.32->langchain) (24.1)
Requirement already satisfied: typing_extensions>=4.7 in /usr/local/lib/python3.10/dist-packages (from langchain-core<0.3.0,>=0.2.32->langchain) (4.12.2)
```

```
1 text = """WebexOne is an annual in-person and virtual event that takes place over four days. It's an event focused on AI collaboration and customer
2 experience.
3 It features a range of activities such as insightful breakout sessions, technical training courses, hands-on labs, inspiring keynotes, epic
4 entertainment, a solutions showcase and expo, customer awards, meet the experts, 1:1 executive meetings, a partner program, and more!"""

```

```
1 from langchain.text_splitter import CharacterTextSplitter
2
3 text_splitter = CharacterTextSplitter(chunk_size=35, chunk_overlap=3, separator='', strip_whitespace=False) # separate means you split by character
4
5 a = text_splitter.create_documents([text])
6
7 print(a)
```

Note: Separators are characters sequence you wanna split on.

WebexOne is an annual in-person and virtual event that takes place over four days. It's an event focused on AI collaboration and customer experience. It features a range of activities such as insightful breakout sessions, technical training courses, hands-on labs, inspiring keynotes, epic entertainment, a solutions showcase and expo, customer awards, meet the experts, 1:1 executive meetings, a partner program, and more!

The screenshot shows a text splitting interface with the following settings:

- Splitter: Character Splitter
- Chunk Size: 35
- Chunk Overlap: 3

Statistics displayed:

- Total Characters: 461
- Number of chunks: 14
- Average chunk size: 32.9

The text "WebexOne is an annual in-person and virtual event that takes place over four days. It's an event focused on AI collaboration and customer experience. It features a range of activities such as insightful breakout sessions, technical training courses, hands-on labs, inspiring keynotes, epic entertainment, a solutions showcase and expo, customer awards, meet the experts, 1:1 executive meetings, a partner program, and more!" is shown, divided into two chunks labeled "Chunk 1" and "Chunk 2". A red arrow labeled "overlap" points from the end of "Chunk 1" to the start of "Chunk 2", indicating the overlap between them.

Recursive Charector Text Splitting

In the previous example, when we used Character Splitting, we split the text based on a fixed number of characters. Specifically, we divided the text into chunks of 35 characters each.

However, with Recursive Text Splitting (RTS), the process is more dynamic and considers the physical structure of the text to determine the appropriate chunk size.

Here's how RTS works:

- Instead of relying on a static number of characters, RTS examines the structure of the document.
- It begins by identifying the largest logical divisions, such as paragraphs, and splits the text at each double newline (indicating paragraph breaks).
- If any of these chunks are still too large, RTS will then move to the next level of separation, which is single newlines (often indicating sentences or list items).
- If necessary, it will continue to break down the text further, using spaces and eventually individual characters as separators.

With RTS, you don't need to manually specify the chunk size by character count. You simply pass your text to the RTS process, and it will intelligently determine how to split the text based on its structure, resulting in chunks that are logically organized and more contextually meaningful.

- "\n\n" - Double new line, or most commonly paragraph breaks
- "\n" - New lines
- " " - Spaces
- "" - Characters

```

1 text = """WebexOne is an annual in-person and virtual event that takes place over four days. It's an event focused on AI collaboration and customer
2 experience.
It features a range of activities such as insightful breakout sessions, technical training courses, hands-on labs, inspiring keynotes, epic
entertainment, a solutions showcase and expo, customer awards, meet the experts, 1:1 executive meetings, a partner program, and more!"""

```

```

1 from langchain.text_splitter import RecursiveCharacterTextSplitter
2 text_splitter = RecursiveCharacterTextSplitter(chunk_size = 35, chunk_overlap = 0)
3 # how many chunks we have
4 print(len(text_splitter.create_documents([text])))
5
6 text_splitter.create_documents([text])

```

Note: We avoid splitting in the middle of words by using spaces as one of the separators, ensuring that each chunk ends on a complete word. This is crucial because RCS (Recursive Character Splitting) helps maintain the context within sentences by keeping related words together.

The screenshot shows a text input area containing a paragraph about WebexOne. Below the input is a configuration section for a 'Recursive Character Text Splitter'. The 'Chunk Size' is set to 35, and the 'Chunk Overlap' is set to 0. The results of the split are displayed below, showing the total number of chunks (14), total characters (422), and average chunk size (30.1). A red arrow points from the text input to the resulting chunks, with the annotation 'RTS maintains words together' written in red. Another red arrow points from the 'Total Number of Chunks' text to the number 14.

WebexOne is an annual in-person and virtual event that takes place over four days. It's an event focused on AI collaboration and customer experience. It features a range of activities such as insightful breakout sessions, technical training courses, hands-on labs, inspiring keynotes, epic entertainment, a solutions showcase and expo, customer awards, meet the experts, 1:1 executive meetings, a partner program, and more!

Upload .txt

Splitter: Recursive Character Text Splitter

Chunk Size: 35

Chunk Overlap: 0

Total Number of Chunks: 14

Total Characters: 422

Average chunk size: 30.1

RTS maintains words together

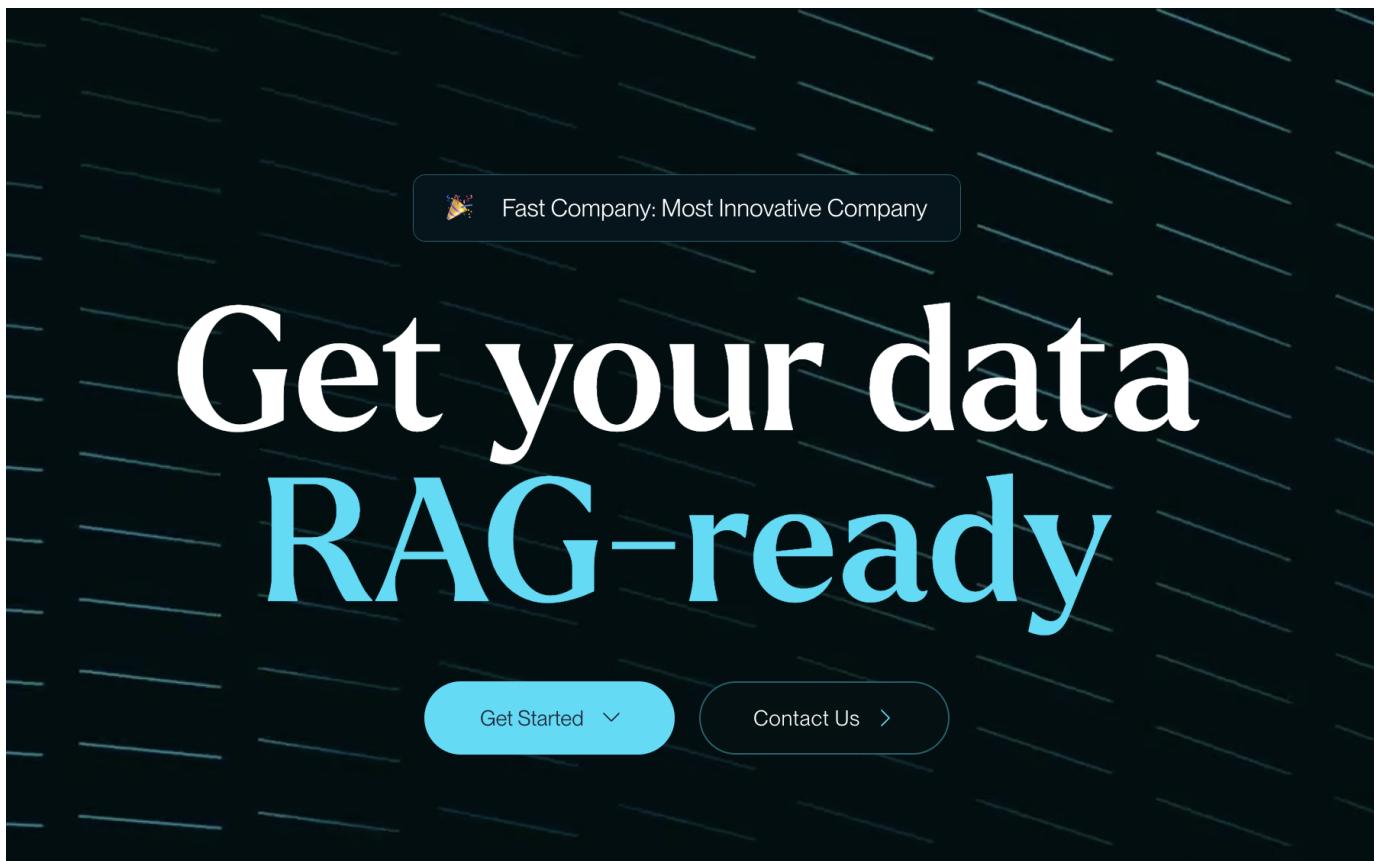
Note: If you're new to AI, I would personally recommend starting with Recursive Text Splitting (RTS).

Document Level Splitting

So far, we've been working with splitting regular documents. But what if we have markdown files or PDF or Python documentation? There's a better way to handle those cases, and that's where specialized document splitting comes into play.

PDF WITH TABLE

PDFs often contain tables and other structured data that can be challenging to split accurately using character-based methods. For PDFs, it's important to extract and chunk all elements, including tables, effectively. We'll accomplish this using the Unstructured library, which is specifically designed for handling such tasks. If you have a large collection of PDFs, Unstructured is an excellent tool to manage them efficiently.



- Install relevant libraries for Unstructured

```
1  !pip install scikit-learn langchain_community unstructured[all-docs] unstructured pdfminer pdfminer.six pdf2image pillow_heif opencv-python
unstructured_inference pytesseract unstructured_pytesseract python-dotenv openai
```

```
Requirement already satisfied: scikit-learn in /usr/local/lib/python3.10/dist-packages (1.3.2)
Requirement already satisfied: langchain_community in /usr/local/lib/python3.10/dist-packages (0.2.12)
Requirement already satisfied: unstructured in /usr/local/lib/python3.10/dist-packages (0.15.5)
Requirement already satisfied: pdfminer in /usr/local/lib/python3.10/dist-packages (20191125)
Requirement already satisfied: pdfminer.six in /usr/local/lib/python3.10/dist-packages (20231228)
Requirement already satisfied: pdf2image in /usr/local/lib/python3.10/dist-packages (1.17.0)
Requirement already satisfied: pillow_heif in /usr/local/lib/python3.10/dist-packages (0.18.0)
Requirement already satisfied: opencv-python in /usr/local/lib/python3.10/dist-packages (4.10.0.84)
Requirement already satisfied: unstructured_inference in /usr/local/lib/python3.10/dist-packages (0.7.36)
Requirement already satisfied: pytesseract in /usr/local/lib/python3.10/dist-packages (0.3.13)
Requirement already satisfied: unstructured_pytesseract in /usr/local/lib/python3.10/dist-packages (0.3.13)
Requirement already satisfied: numpy<2.0,>=1.17.3 in /usr/local/lib/python3.10/dist-packages (from scikit-learn) (1.26.4)
Requirement already satisfied: scipy>=1.5.0 in /usr/local/lib/python3.10/dist-packages (from scikit-learn) (1.13.1)
Requirement already satisfied: joblib>=1.1.1 in /usr/local/lib/python3.10/dist-packages (from scikit-learn) (1.4.2)
Requirement already satisfied: threadpoolctl>=2.0.0 in /usr/local/lib/python3.10/dist-packages (from scikit-learn) (3.5.0)
Requirement already satisfied: PyYAML>=5.3 in /usr/local/lib/python3.10/dist-packages (from langchain_community) (6.0.2)
Requirement already satisfied: SQLAlchemy<3,>=1.4 in /usr/local/lib/python3.10/dist-packages (from langchain_community) (2.0.32)
Requirement already satisfied: aiohttp<4.0.0,>=3.8.3 in /usr/local/lib/python3.10/dist-packages (from langchain_community) (3.10.2)
Requirement already satisfied: dataclasses-json<0.7,>=0.5.7 in /usr/local/lib/python3.10/dist-packages (from langchain_community) (0.6.7)
Requirement already satisfied: langchain<0.3.0,>=0.2.13 in /usr/local/lib/python3.10/dist-packages (from langchain_community) (0.2.14)
Requirement already satisfied: langchain-core<0.3.0,>=0.2.30 in /usr/local/lib/python3.10/dist-packages (from langchain_community) (0.2.33)
Requirement already satisfied: langsmith<0.2.0,>=0.1.0 in /usr/local/lib/python3.10/dist-packages (from langchain_community) (0.1.99)
```

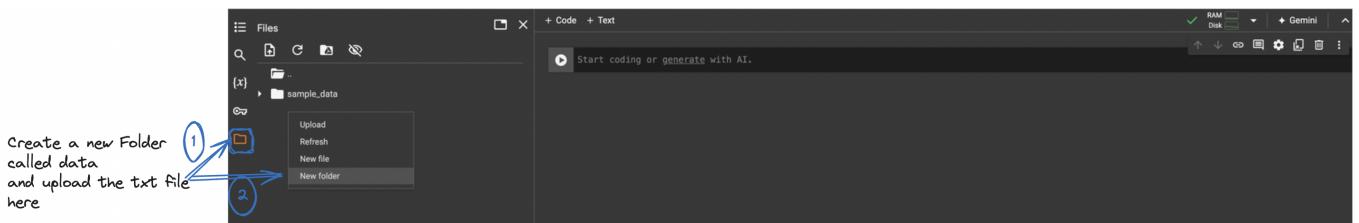
```
1  !apt-get install -y poppler-utils && apt-get install -y tesseract-ocr
```

```

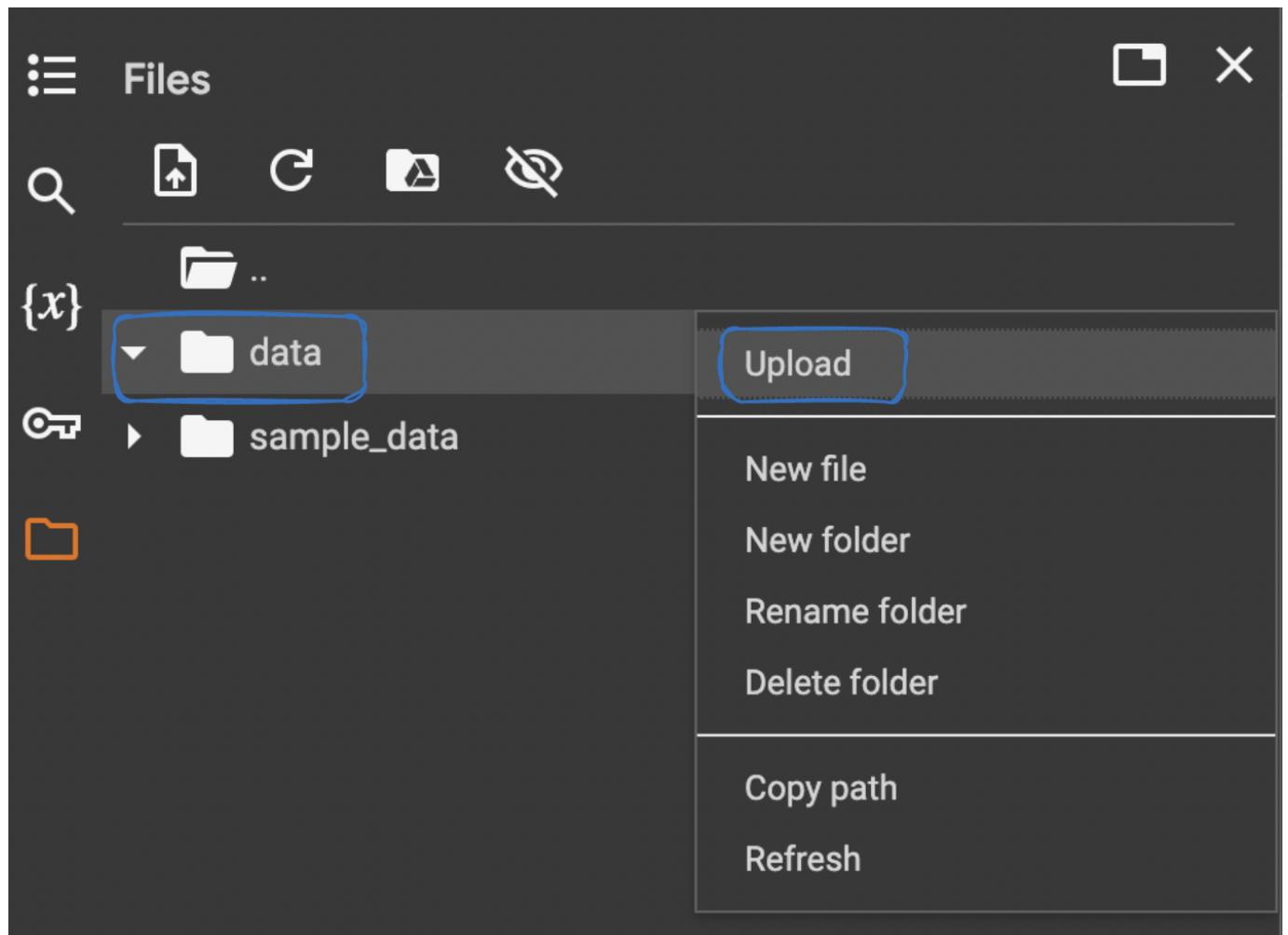
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following NEW packages will be installed:
  poppler-utils
0 upgraded, 1 newly installed, 0 to remove and 45 not upgraded.
Need to get 186 kB of archives.
After this operation, 696 kB of additional disk space will be used.
Get:1 http://archive.ubuntu.com/ubuntu jammy-updates/main amd64 poppler-utils amd64 22.02.0-2ubuntu0.5 [186 kB]
Fetched 186 kB in 1s (228 kB/s)
Selecting previously unselected package poppler-utils.
(Reading database ... 123594 files and directories currently installed.)
Preparing to unpack .../poppler-utils_22.02.0-2ubuntu0.5_amd64.deb ...
Unpacking poppler-utils (22.02.0-2ubuntu0.5) ...
Setting up poppler-utils (22.02.0-2ubuntu0.5) ...
Processing triggers for man-db (2.10.2-1) ...

```

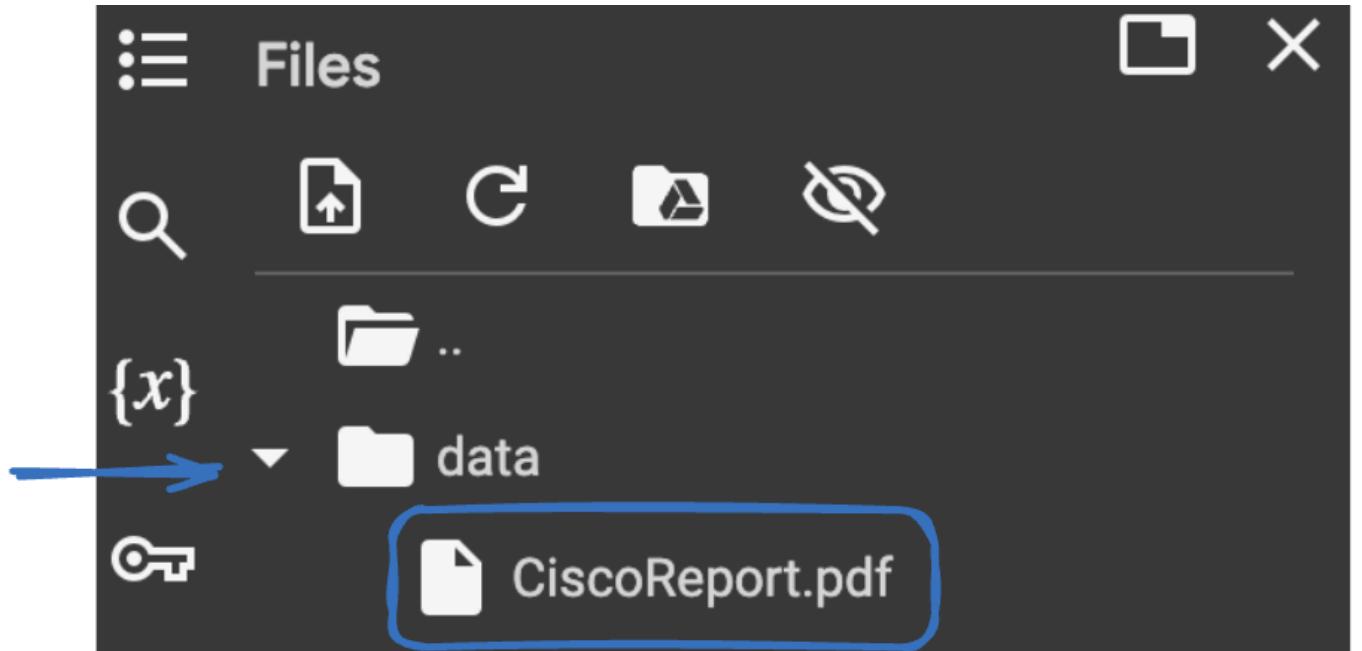
- Let's load our PDF files into Google Colab. For this example, we can use the Cisco Financial Results. Please Download the file here as we will be using in the next step.
- Within Google Colab, Click on Folder and create a new folder called "data"



- Click on [...], select Upload



- Choose your CiscoReport.pdf file and click Open



- We'll use the following code from the Unstructured library to demonstrate how tables can be extracted

```

1 import os
2 from unstructured.partition.pdf import partition_pdf
3 from unstructured.staging.base import elements_to_json
4
5 # Let's load up our PDF and then partition it.
6 filename = "/content/data/CiscoReport.pdf"# Use relative path since the file is in the same directory
7
8 # Extracts the elements from the PDF
9 elements = partition_pdf(
10     filename=filename,
11     extract_images_in_pdf=True,
12     strategy="hi_res",
13     infer_table_structure=True,
14     hi_res_model_name="yolox"
15 )
16
17 # Let's look at our elements
18 print(elements)

```

```
config.json: 100% [1.47k/1.47k [00:00<00:00, 76.5kB/s]
model.safetensors: 100% [115M/115M [00:01<00:00, 125MB/s]
model.safetensors: 100% [46.8M/46.8M [00:00<00:00, 117MB/s]

<unstructured.documents.elements.Title at 0x7ba52cc9c5e0>,
<unstructured.documents.elements.ListItem at 0x7ba52cc9d7b0>,
<unstructured.documents.elements.ListItem at 0x7ba52cc9d3f0>,
<unstructured.documents.elements.Title at 0x7ba52cc9caf0>,
<unstructured.documents.elements.ListItem at 0x7ba4c0997ca0>,
<unstructured.documents.elements.Text at 0x7ba4c09943a0>,
<unstructured.documents.elements.NarrativeText at 0x7ba52cc9ebc0>, ← Regular Text
<unstructured.documents.elements.NarrativeText at 0x7ba52cc9f370>,
<unstructured.documents.elements.NarrativeText at 0x7ba52cc9d720>,
<unstructured.documents.elements.Title at 0x7ba52cc9de10>,
<unstructured.documents.elements.Text at 0x7ba4c0997f10>,
<unstructured.documents.elements.Text at 0x7ba4c0994460>,
<unstructured.documents.elements.Text at 0x7ba4c0994520>,
<unstructured.documents.elements.Table at 0x7ba4c0994160>,
<unstructured.documents.elements.NarrativeText at 0x7ba4c09944f0>,
<unstructured.documents.elements.Title at 0x7ba4c09946a0>,
<unstructured.documents.elements.Text at 0x7ba4c09947f0>,
<unstructured.documents.elements.Text at 0x7ba4c0994730>,
<unstructured.documents.elements.Text at 0x7ba4c0994430>,
<unstructured.documents.elements.Table at 0x7ba4c0995150>, ← Table
<unstructured.documents.elements.Title at 0x7ba4c0994790>,
```

- Lets grab element 37 as its the one where our table is

```
1     elements[37].metadata.text_as_html
```

OUTPUT

```
'<table><thead><tr><th>Revenue</th><th>$8.51 - $8.53 Billion</th><th>$34.7 - $34.7 Billion</th></tr></thead><tbody><tr><td>Y/Y Growth</td><td>-10%</td><td>-10%</td></tr><tr><td>FX Impact</td><td>no impact</td><td>no impact</td></tr><tr><td>GAAP Operating Margin</td><td>-11.4%</td><td>-11.4%</td></tr><tr><td>Non-GAAP Operating Margin</td><td>-28.0%</td><td>-28.0%</td></tr><tr><td>GAAP Earnings per Share($)</td><td>$0.79 - $0.80</td><td>$0.67 - $2.69</td></tr><tr><td>Non-GAAP Earnings per Share($)</td><td>$1.89 - $1.90</td><td>$7.43</td></tr><tr><td>Operating Cash Flow Growth (Y/Y)%</td><td>-17%</td><td>-17%</td></tr><tr><td>Current Remaining Performance Obligation Growth (Y/Y)%</td><td>-10%</td><td>-10%</td></tr></tbody></table>'
```

- Tables are straightforward for humans to read, but they aren't as easy for language models to interpret. Language models are typically trained on HTML tables, so when you pass HTML-formatted tables to an LLM, it will better understand the structure and content. You can paste the HTML into an HTML viewer to see how it looks.

HTML Viewer

The screenshot shows the PDF.js interface with the following components:

- HTML Code:** A code editor containing the following HTML table structure:

```
<table><tbody><tr><td>Revenue</td><td>$13.6 billion</td><td>15.2 billion</td><td>(10)%</td></tr><tr><td>Net Income</td><td>$2.2 billion</td><td>$4.0 billion</td><td>(45)%</td></tr><tr><td>Diluted Earnings per Share (EPS)</td><td>$0.54</td><td>$0.97</td><td>(44)%</td></tr></tbody></table>
```
- Beautify HTML:** A blue button labeled "Beautify HTML".
- Output:** A panel displaying the table data in a clean format:

Revenue	\$ 13.6 billion	\$ 15.2 billion	(10)%
Net Income	\$ 2.2 billion	\$ 4.0 billion	(45)%
Diluted Earnings per Share (EPS)	\$ 0.54	\$ 0.97	(44)%

A large blue arrow points upwards from the "HTML Code" label towards the code editor. Another large blue arrow points upwards from the "Our First table in the pdf" label towards the output panel.

Note: That's how you handle tables within a PDF

Multi-Modal (Text + images)

What if there are images within a PDF or other documents? How can you extract them? We'll use the Unstructured library again to handle this.

EXTRACT IMAGES WITHIN PDF

- Install relevant libraries

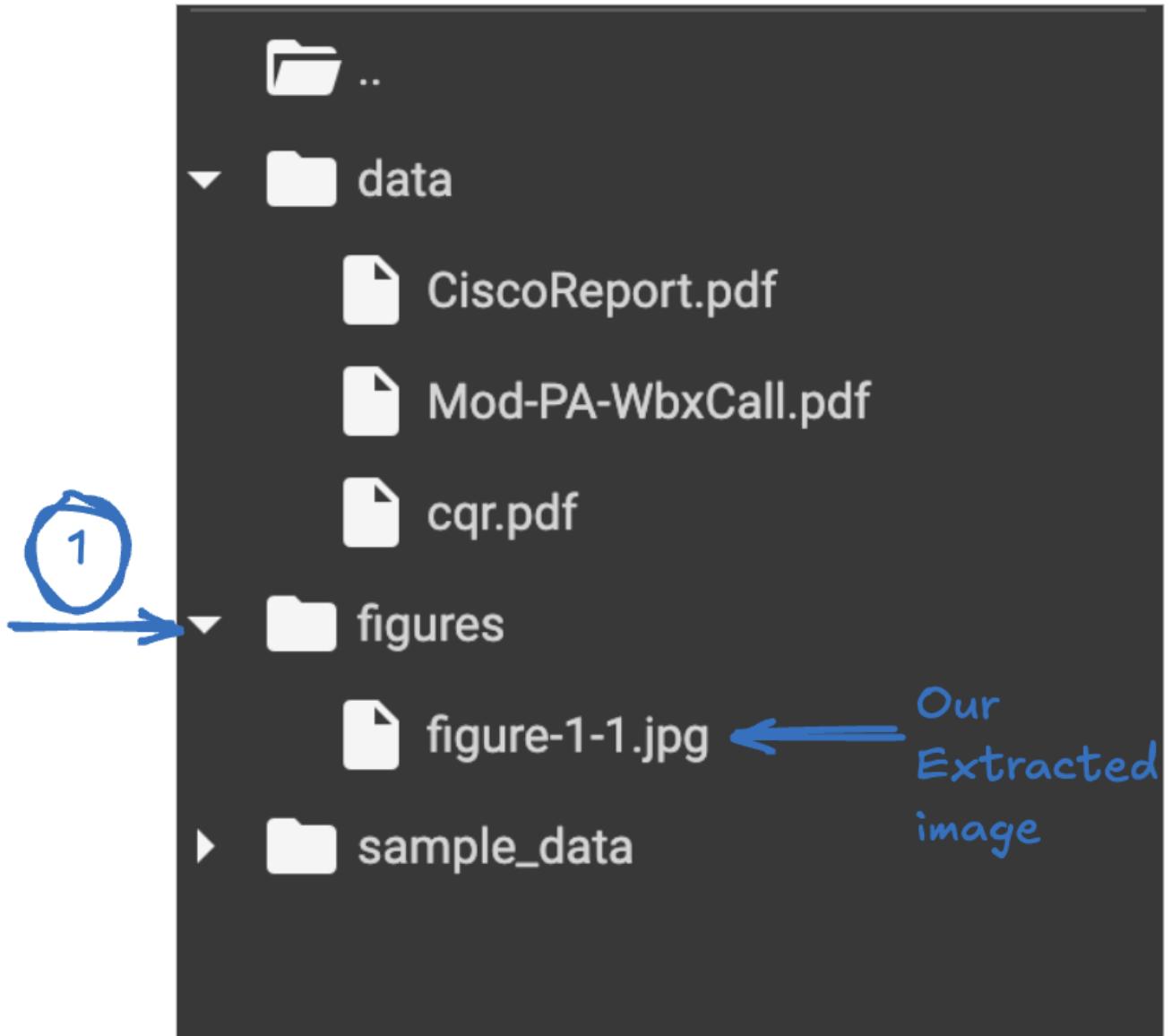
```

1  from typing import Any
2  from pydantic import BaseModel
3  from unstructured.partition.pdf import partition_pdf

1  filepath = "/content/data/CiscoReport.pdf"
2  # Get elements
3  raw_pdf_elements = partition_pdf(
4      filename=filepath,
5
6      # Using pdf format to find embedded image blocks
7      extract_images_in_pdf=True,
8
9      # Use layout model (YOLOX) to get bounding boxes (for tables) and find titles
10     infer_table_structure=True,
11
12     # Specifies the strategy to be used for chunking the text. In our case, it will chunk the text based on titles found in the document.
13     chunking_strategy="by_title",
14
15     # Sets a hard limit on the number of characters allowed in each chunk.
16     max_characters=4000,
17     # This function will attempt to create a new chunk after every 3,800 characters, allowing some flexibility while chunking.
18     new_after_n_chars=3800,
19     # If a chunk has fewer than 2,000 characters, the function will
20     # attempt to combine it with neighboring text blocks to create a larger, more meaningful chunk.
21     combine_text_under_n_chars=2000
22 )

```

Note: You'll notice that a folder called 'figures' is created, where all the extracted images are stored. To make these images more useful, we can generate embeddings for them, which can later be used for semantic search. Typically, embedding models are specialized—they either handle text or images, but not both. This means the vector lengths won't align, and using different models for text and image embeddings can complicate similarity searches. However, the CLIP model can generate embeddings for both images and text, making it a versatile option for this purpose. To simplify things, we can first create a text summary for each image, and then generate embeddings based on those summaries. This approach allows us to leverage text-based semantic search while still incorporating the visual information from the images.



Note: After extracting images from a PDF or any other source, you can use the below method to send them to a multimodal model like GPT-4o to understand and generate insights about the images. This approach allows you to combine textual and visual data for a richer understanding of the content.

SET OPENAI TOKEN - MULTIMODAL

Note: First, create an account from the [OpenAI official website](#).

<https://platform.openai.com/api-keys>

- Create a new project API key by browsing to API Keys web page. Select Create new secret key. The API key is automatically generated. Save the APi Key as we will be using it in the later steps .

Save your key

Please save this secret key somewhere safe and accessible. For security reasons, **you won't be able to view it again** through your OpenAI account. If you lose this secret key, you'll need to generate a new one.

JWZs13myG1i0gbLGkU9XHe6wEAhEsRwm8gxjX6eYwf8A

Copy

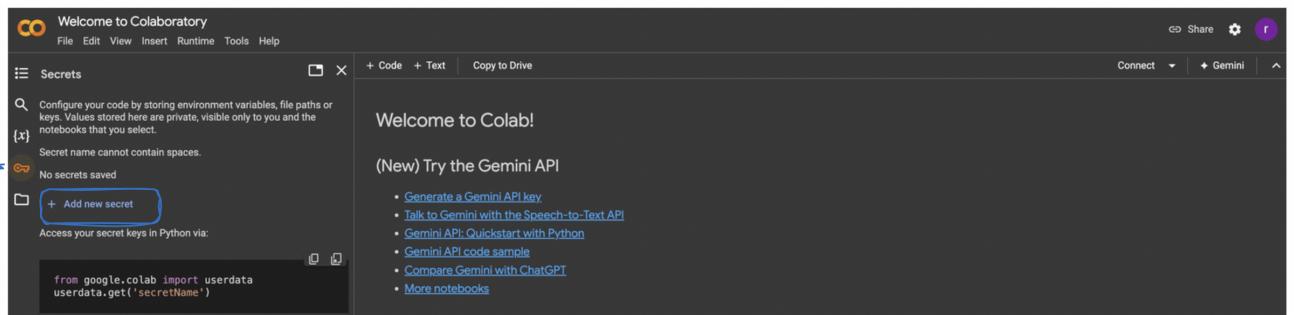
Permissions

Read and write API resources

Done

Prepare your python environment using Hugging Face

- Within your existing Google Colab notebook navigate to the new “Secrets” section in the sidebar.



- Click on “Add a new secret.” Enter the name example: OPENAI_API_KEY and value of the secret(the API key created above). Note: The name is permanent once set.
- The list of secrets is global across all your notebooks.
- Use the “Notebook access” toggle to grant or revoke access to a secret for each notebook.

Secrets

Configure your code by storing environment variables, file paths or keys. Values stored here are private, visible only to you and the notebooks that you select.

Secret name cannot contain spaces.

Notebook access	Name	Value	Actions		
<input type="button" value="X"/>	HF_TOKEN	<input type="button" value=""/>	<input type="button" value=""/>	<input type="button" value=""/>
<input checked="" type="button" value=""/>	OPENAI_API_KE	<input type="button" value=""/>	<input type="button" value=""/>	<input type="button" value=""/>

+ Add new secret

Create Gemini API key

Access your secret keys in Python via:

```
from google.colab import userdata
userdata.get('secretName')
```

Lets import and load the environment variables

```
1 from langchain.chat_models import ChatOpenAI
2 from langchain.schema.messages import HumanMessage
3 import os
4 from dotenv import load_dotenv
5 from PIL import Image
6 import base64
7 import io
8 load_dotenv()
```

Incorporating Secrets into Your Code - We will use it later in our lab

```
1 import os
2 from google.colab import userdata
3 os.environ['OPENAI_API_KEY'] = userdata.get('OPENAI_API_KEY')
4 llm = ChatOpenAI(model="gpt-4o")
```

Function to Convert Image to Base64:

```
1 # Function to convert image to base64
2 def image_to_base64(image_path):
3     with Image.open(image_path) as image:
4         buffered = io.BytesIO()
5         image.save(buffered, format=image.format)
6         img_str = base64.b64encode(buffered.getvalue())
7         return img_str.decode('utf-8')
8
9 image_str = image_to_base64("/content/figures/figure-1-1.jpg")
```

Note: This is the image that was extracted from our data and saved in the figures folder.

```
1     print(image_str)
```

Output: Our base64-encoded string

Initializing the Multimodal and sending the Image to GPT-4o for Analysis:

```

1  chat = ChatOpenAI(model="gpt-4o", max_tokens=1024)
2
3  msg = chat.invoke(
4      [
5          HumanMessage(
6              content=[
7                  {"type": "text", "text": "Please give a summary of the image provided. Be descriptive"},
8                  {"type": "image_url", "image_url": {
9                      "url": f"data:image/jpeg;base64,{image_str}"
10                 }
11             },
12         ]
13     ]
14 )
15 )

```

- Retrieving the Response:

```
1  msg.content
```

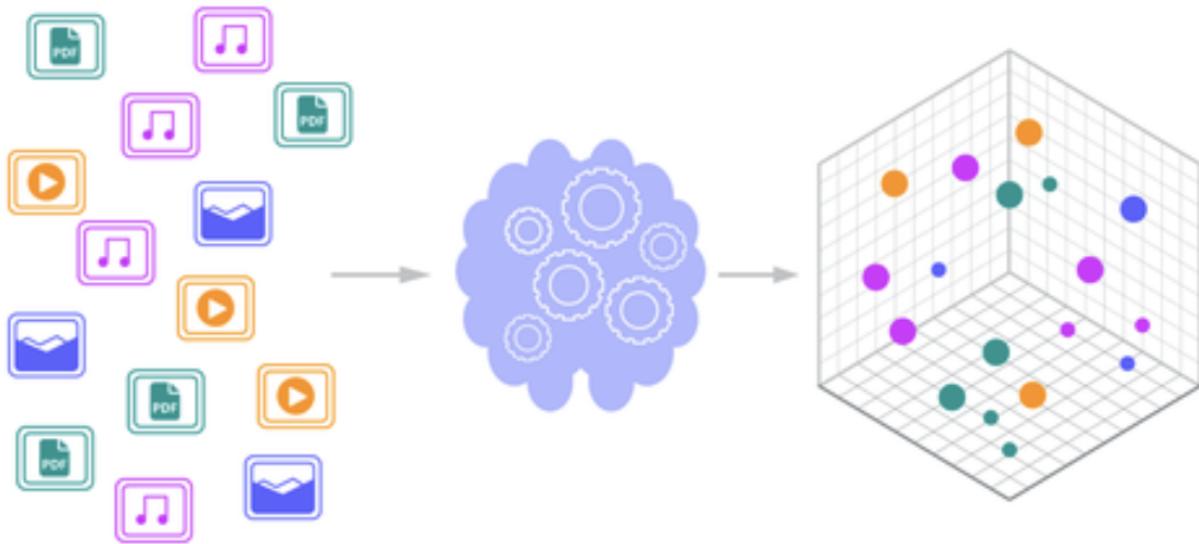
Output: The response from the model is stored in `msg.content`, which will contain the descriptive summary of the image.

The image is the logo of Cisco Systems, Inc., commonly known as Cisco. The logo features the company name "cisco" written in lowercase letters in a distinct, modern sans-serif font. Above the text, there is a series of vertical bars of varying heights, which is stylized to resemble the Golden Gate Bridge. The logo is rendered in a light blue color. The design is clean, professional, and easily recognizable, reflecting Cisco's identity as a major tech company specializing in networking hardware, telecommunications equipment, and other high-technology services and products.

Note: The description of your image may vary depending on the image retrieved.

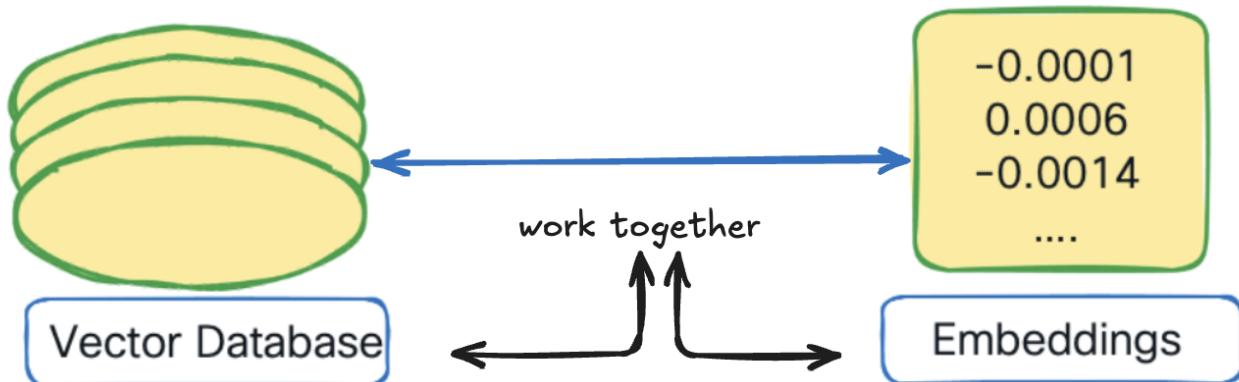
1.5 Task 4: Embeddings and Vector Database

- Vector databases are special databases meant to store content such as PDFs, Blogs, Word Documents, Images, Audio Files, and even Videos as embedding vectors, allowing for semantic based retrieval.



- In the process of a large language model, embedding is generally the first step to convert discrete tokens (like words or subwords) into dense vector representations, which will allow the rest of the network to do the math necessary to predict the next word. Embeddings and vector databases are essential if you are building any kind of AI product.

Embeddings and Vector Database



1.5.1 What are embeddings?

Embeddings are basically data.

Embeddings and Vector databases

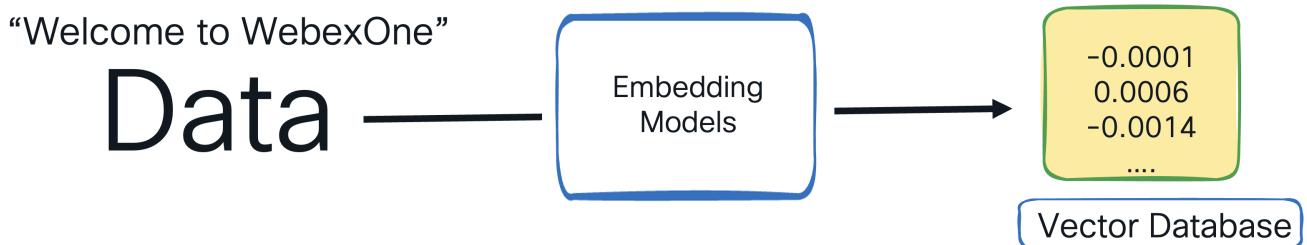
What are Embeddings?



That are converted into array of numbers called vectors that contain pattern of relationship, and can be used for similarity search.

Embeddings and Vector databases

What are Embeddings?



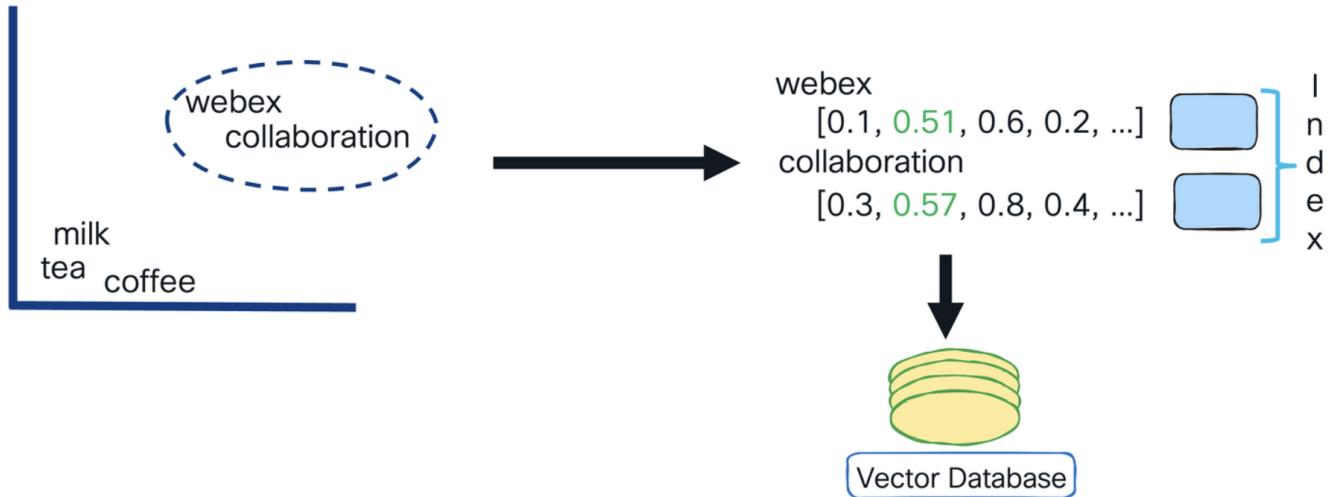
Let me explain this concept using a 2D graph. In this graph, words like "Webex" and "collaboration" are often used in similar context, so their embeddings essentially vector representations are positioned close to each other. These vectors are numerical arrays that computers can easily interpret.

The advantage of using vectors is that you can find similar items by calculating the distances between their vectors, a process known as nearest neighbor search. However, simply storing these embeddings isn't enough. Performing queries across thousands of vectors can be incredibly slow. To overcome this, these vectors also need to be indexed in the vector database.

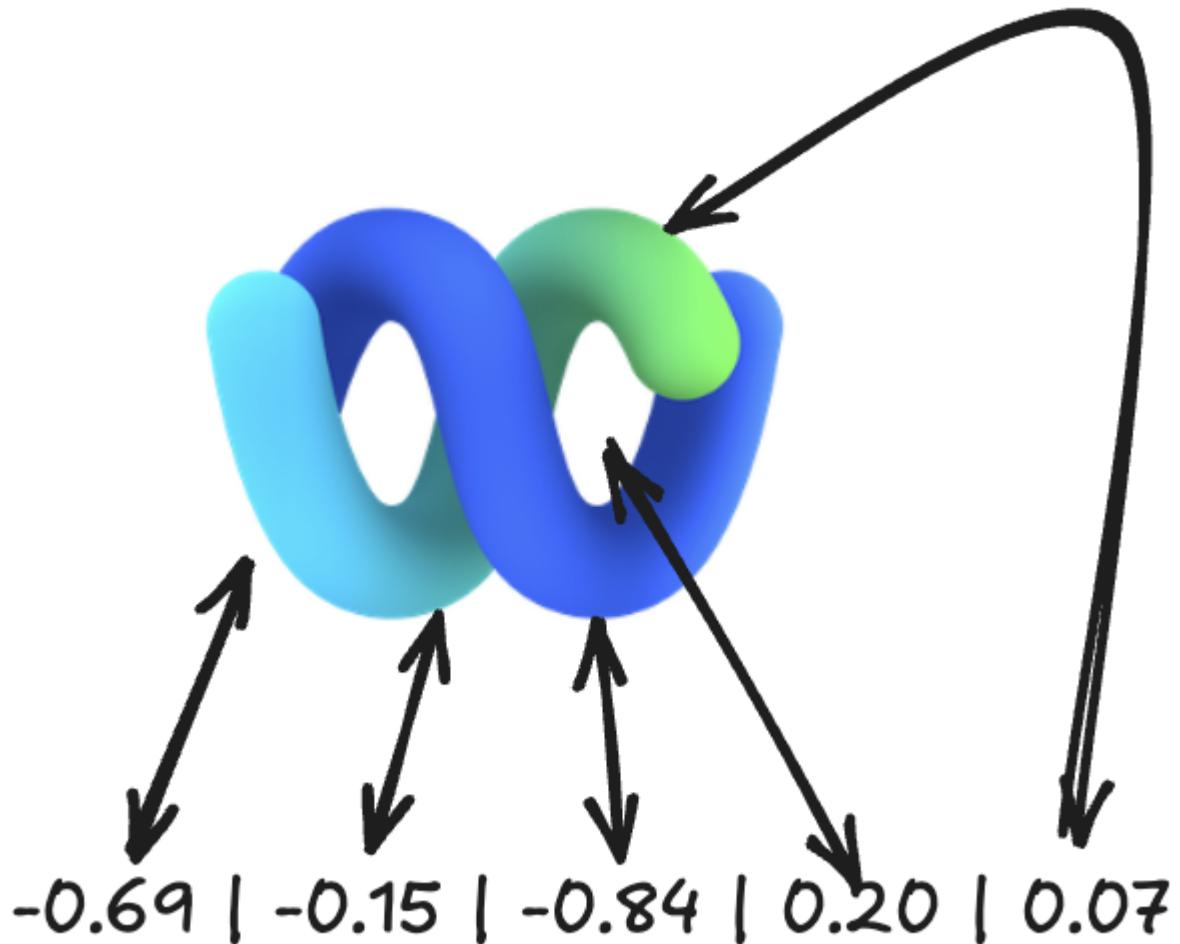
Note: In reality vectors can have hundred of dimensions

Embeddings and Vector databases

What are Embeddings?



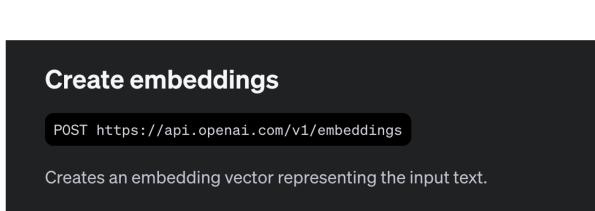
Similarly, images are also broken into vectors, which are arrays of numbers that machines can process. Once these embeddings (both for words and images) are generated, they are stored in a specialized database known as a vector database.



There are numerous embedding models available, such as Google's Word2Vec, CLIP (Contrastive Language-Image Pretraining), and even those provided by OpenAI, which offer excellent capabilities for generating embeddings. However, the challenge is that these models don't include tools for storing and managing those embeddings. That's where vector databases become essential.

Embeddings and Vector Database

OpenAI - Embeddings



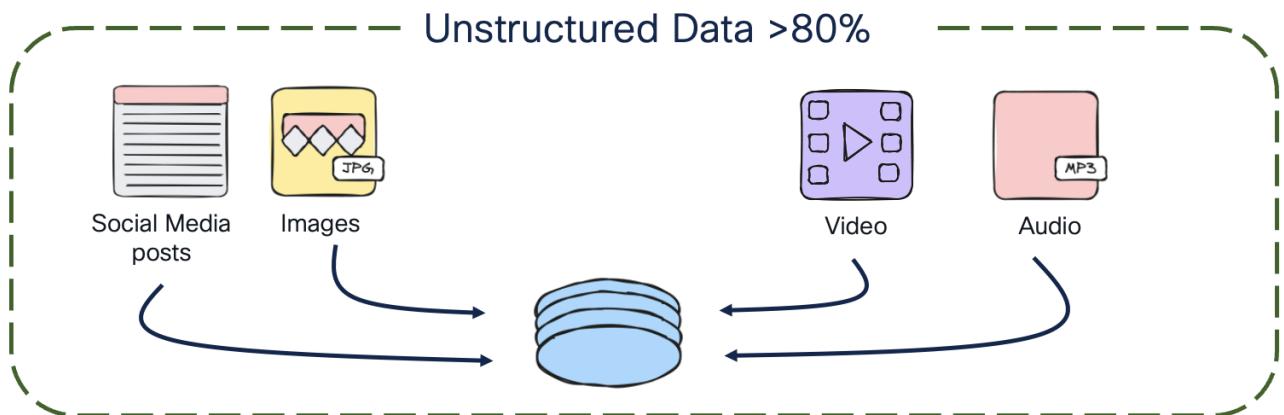
Note: In this lab session, we will be using OpenAI embeddings. More info can be found [here](#)

1.5.2 Why Vector Databases when we have relational DB's?

Around 80% of the data we encounter is unstructured, including social media posts, images, and videos. This type of data doesn't easily fit into traditional relational databases, which is where vector databases come into play.

Embeddings and Vector databases

Why Vector Database?

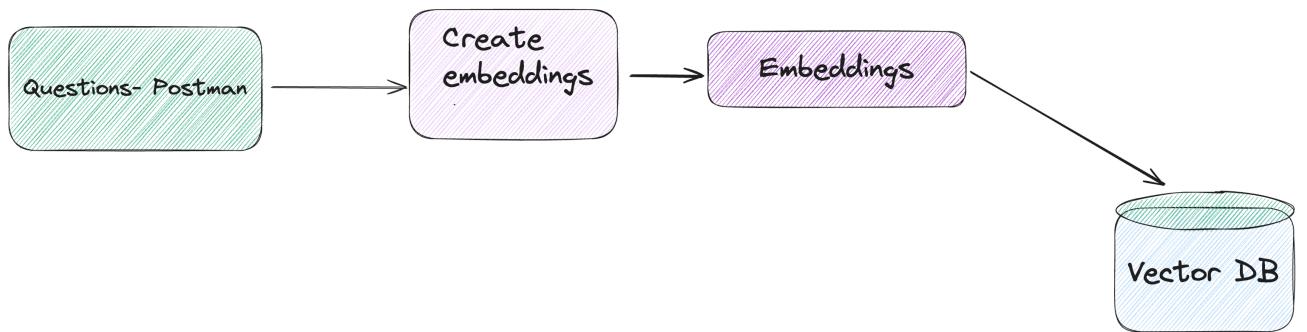


1.5.3 Understanding working of Embeddings and Vector Databases

Practical Example of Embedding Techniques Using Postman

Embeddings and Vector databases

High Level Overview



SET OPENAI TOKEN

Note: If you've already obtained the token in previous steps, you can skip this section

Note: First, create an account from the [OpenAI official website](#).

- Create a new project API key by browsing to API Keys web page. Select Create new secret key. The API key is automatically generated. Save the APi Key as we will be using it in the later steps .

Save your key

Please save this secret key somewhere safe and accessible. For security reasons, **you won't be able to view it again** through your OpenAI account. If you lose this secret key, you'll need to generate a new one.

JWZs13myG1i0gbLGkU9XHe6wEAhEsRwm8gxjX6eYwf8A

Copy

Permissions

Read and write API resources

Done

Note: We will use the OpenAI key in Postman to generate embeddings.

LOGIN TO POSTMAN

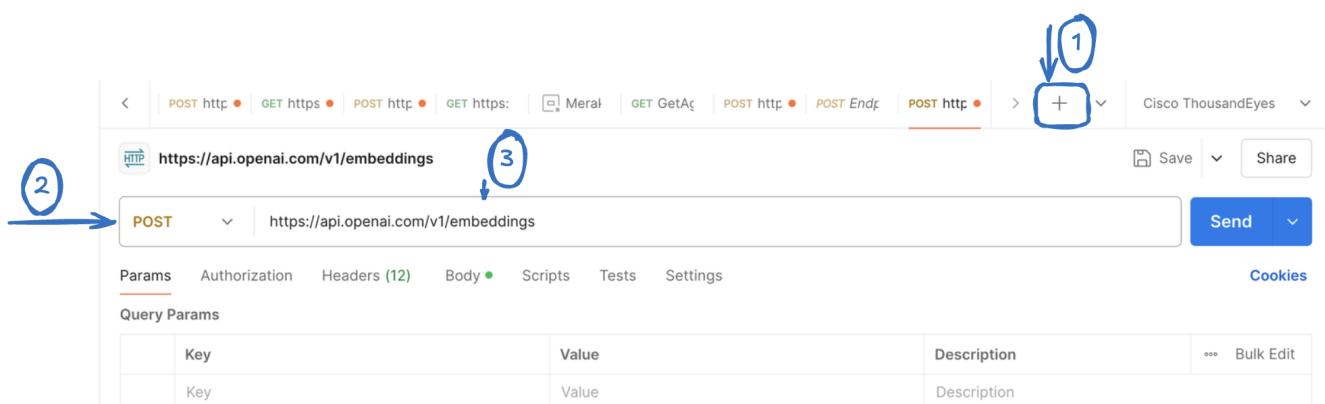
If you haven't installed Postman on your machine yet, you can download it from the following link

As we will be using embedding APIs to create an embedding vector that represents our input text, more information can be found at the following link

The POST request we will be using:

```
https://api.openai.com/v1/embeddings
```

* Open Postman, create a new request by pressing +. Select POST as your request and enter the request url



- Click on the Headers tab and enter the Following creds

```
Authorization: Bearer <replace with your openAi API key>
Content-Type : application/json
```

HTTP <https://api.openai.com/v1/embeddings>

POST https://api.openai.com/v1/embeddings

Params Authorization Headers (12) Body Scripts Tests Settings Cookies

Headers 10 hidden

Key	Value	Description	Bulk Edit	Presets
<input checked="" type="checkbox"/> Authorization	Bearer sk-Xm2PdTmYBhBCVf0dvpOhT3BlkFJht...			
<input checked="" type="checkbox"/> Content-Type	application/json			
Key	Value	Description		

Enter your API key and Content-Type

- In this lab, we will be using the text-embedding-ada-002 model, but feel free to use any other embedding model of your choice.

Embedding models

OpenAI offers two powerful third-generation embedding model (denoted by -3 in the model ID). You can read the embedding v3 [announcement blog post](#) for more details.

Usage is priced per input token, below is an example of pricing pages of text per US dollar (assuming ~800 tokens per page):

MODEL	~ PAGES PER DOLLAR	PERFORMANCE ON MTEB EVAL	MAX INPUT
text-embedding-3-small	62,500	62.3%	8191
text-embedding-3-large	9,615	64.6%	8191
text-embedding-ada-002	12,500	61.0%	8191

- In Postman, click on the "Body" tab, enter the following information, and then press "Send."

```
{
  "input": "What is WebexOne 2024? WebexOne is an annual in-person and virtual event that takes place over four days. It's an event focused on AI collaboration and customer experience, and it features a range of activities such as insightful breakout sessions, technical training courses, hands-on labs, inspiring keynotes, epic entertainment, a solutions showcase and expo, customer awards, meet the experts, 1:1 executive meetings, a partner program, and more!",
  "model": "text-embedding-ada-002"
}
```

HTTP <https://api.openai.com/v1/embeddings>

POST https://api.openai.com/v1/embeddings

Params Authorization Headers (12) **Body** Scripts Tests Settings Cookies

Body raw JSON

none form-data x-www-form-urlencoded raw binary GraphQL JSON

Beautify

Input

Model

1
2
3
4
5

- You will receive a 200 OK message, confirming success, and you'll notice that our text has been converted into embeddings (vectors). In the upcoming steps, I will show you on how to manually save this information in a vector database.

```

1 {
2   "object": "list",
3   "data": [
4     {
5       "object": "embedding",
6       "index": 0,
7       "embedding": [
8         -0.0042485874,
9         -0.02296605,
10        0.011213377,
11        0.01114761,
12        -0.027280405,
13        0.0023528363,
14        -0.004350527,
15        0.0043768343,
16        -0.00068562734,
17        -0.011660597,
18        0.010397859,
19        0.018730614,
20        -0.0158763,
21        -0.013469206,
22        0.0026997605,
      ]
    }
  ]
}

```

Embeddings

Note: Text embedding models, convert text into numerical data(embeddings) that represent text meaning.

INSERTING VALUES IN VECTOR DATABASE

We have a variety of databases available. In the upcoming step, I will demonstrate how to use SingleStore as a vector database and save our embeddings there.

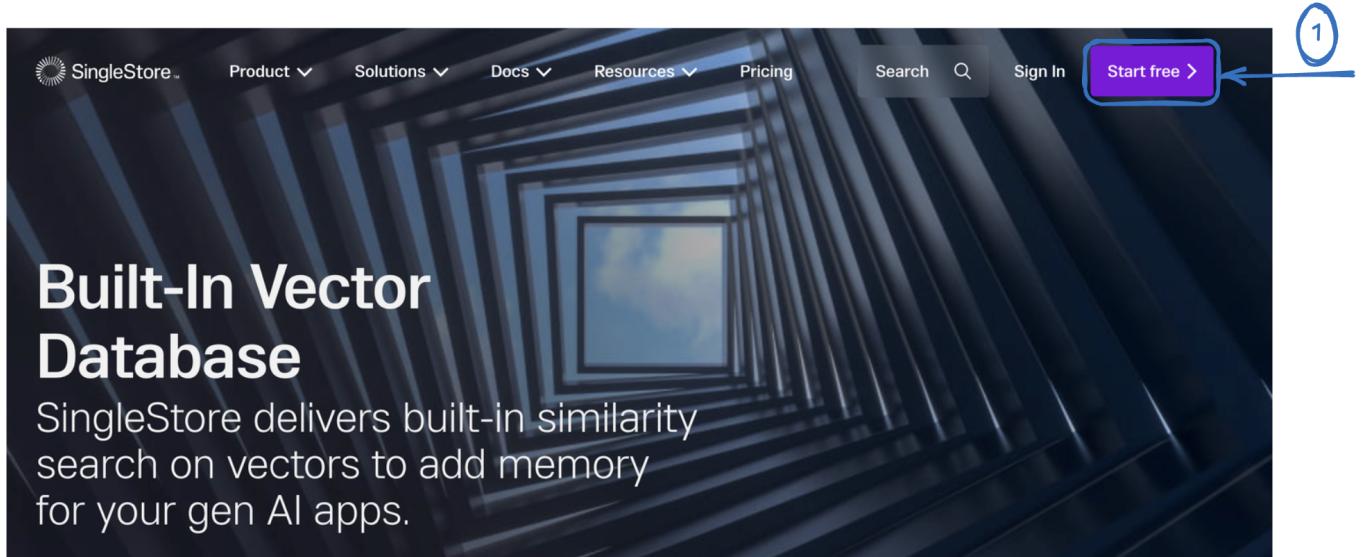
Note: This manual step is simply to illustrate how embeddings are stored in vector databases.

Embeddings and Vector Database

A few examples of Vector Database?



Note: We will create a Free account on SingleStore since it provides some credits to set up a trial account.



- I'm choosing to create an account using Google, but feel free to use any other method that works for you.



Try it free



 Sign up with Google

 Sign up with Microsoft

OR

First Name

Last Name

 Enter your first name

Company Email Address

This email address will be used as your username

Continue →

Already have an account? [Sign In](#)

- Select Continue

 Sign in with Google



Sign in to SingleStore

By continuing, Google will share your name, email address, language preference, and profile picture with SingleStore. See SingleStore's [Privacy Policy](#) and [Terms of Service](#).

You can manage Sign in with Google in your [Google Account](#).

[Cancel](#) [Continue](#)

1



Add your details

Please provide your details to receive a personalised experience and make the most of your trial.

JOB TITLE *

Developer / Software En... ▾

COUNTRY *

United Kingdom of Grea... ▾

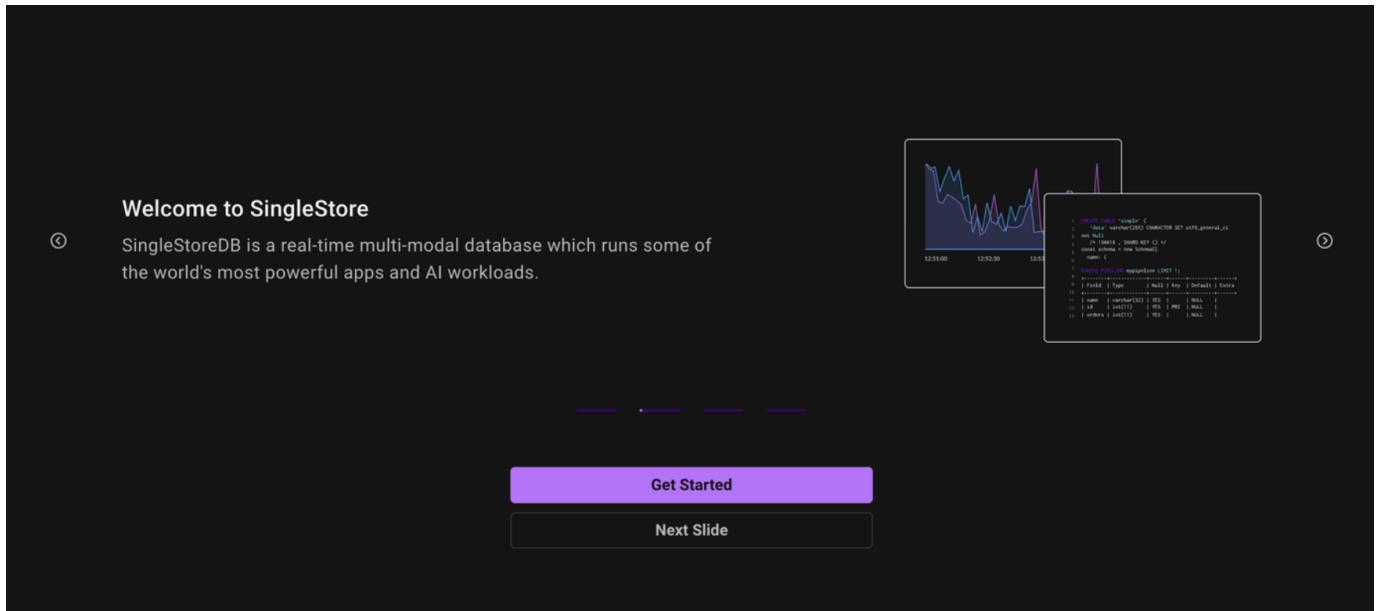
Company Name *

Omer

Company Email Address

omer.ilias@boldbetz.com

Continue >



- Click on New Deployment

1

+ New Deployment

Welcome, ilyas!

Suggested for you

- Load files from S3 into Shared Tier
- Hybrid Search
- Try SQrl, our AI assistant

Recents

starter-workspace Deployed

Help

Tutorials & Academy

- Give your workspace a name, keep all other settings at their default, and click "Next."

The screenshot shows the configuration page for creating a new workspace. A blue circle labeled '1' points to the 'Workspace Group Name' field, which contains 'OpenAI VectorDB'. A blue circle labeled '2' points to the 'Next' button at the bottom right.

Standard Workspace

Starter PREVIEW FREE
Ideal for starting projects and prototyping with our Shared Edition

Standard
Get full functionality and performance of SingleStore deployed on dedicated compute

1 Workspace Group Name
OpenAI VectorDB
A Workspace Group enables you to share data and access settings between multiple databases and workspaces. It is good practice to label a group as either: dev, prod or staging.

2 Cloud Provider
 AWS
 GCP
 Azure

3 Region
US East 1 (N. Virginia)
Want to try SingleStore in another region? Contact Us.

Advanced Settings Optional
Deploy In Two Availability Zones
Enable deployment across two Availability Zones, incurring a 1.3x increase in credit cost.

Summary
Workspace Group: OpenAI VectorDB
Provider and Region: AWS US East 1 (N. Virginia)
Workspace Name: —

Compute
Size: —
Credits per Hour: —
Credits Available: 60 CR
Free Trial Credits Applied: 60 Credits (US\$216). More will be added upon completion of the onboarding checklist.

Next

- Then, leave everything as is and click on "Create Workspace."

Create Workspace

Workspaces provide compute resources to run operations on a database. After you set up your first workspace you can create databases and attach them to this workspace to perform different types of operations.

1 Workspace Details

Workspace Name
workspace-1

Choose a name that will help identify what type of workloads will be connected to this workspace

2 Size

S-00			Edit
vCPUs	RAM	Credits per hour	
2	16 GB	0.25 CR	

Compute consumes a fixed number of credits per hour, and can be scaled up or down to increase or decrease performance. [Learn about credits](#) or [contact us to prepay for a discount](#).

Storage is separate from Compute
Storage is billed separately from compute charges and are based on the average total number of GB used per month.

Settings

SingleStore Kai Enable the MongoDB API and endpoint for your workspace.

Deployment Type
Updates roll out to non-production deployments prior to production.
Production

Auto Suspend

Summary

Workspace Group
OpenAI VectorDB

Provider and Region
AWS US East 1 (N. Virginia)

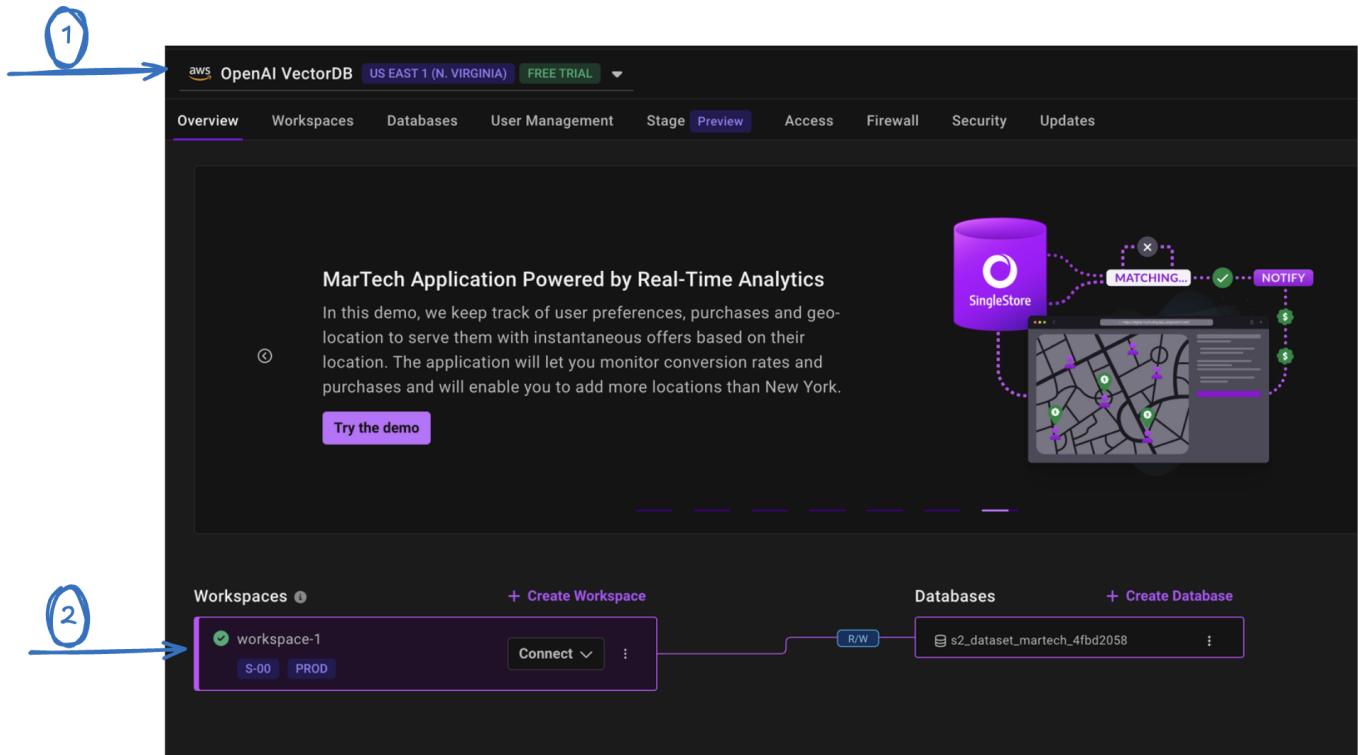
Workspace Name
workspace-1

Compute

Size	S-00
Credits per Hour	0.25 CR / hour
Credits Available	60 CR

Free Trial Credits Applied: 60 Credits (US\$216).
More will be added upon completion of the onboarding checklist.

Create Workspace



- Now we can create a database

aws OpenAI VectorDB US EAST 1 (N. VIRGINIA) FREE TRIAL

Overview Workspaces Databases User Management Stage Preview Access Firewall Security Updates

MarTech Application Powered by Real-Time Analytics
In this demo, we keep track of user preferences, purchases and geo-location to serve them with instantaneous offers based on their Ingest live and historical data from different sources into your workspace. We support AWS S3, GCS, Azure Blob and Kafka. Try the demo

Workspaces + Create Workspace
workspace-1 S-00 PROD Connect :

Databases + Create Database
s2_dataset_martech_4fdb2058

Create Database

Choose a new database name
WebexOneDB

Attach To
workspace-1

Attachment Type
Read & Write
Submit

Pricing (Average GB per Month)
\$0.023

Cancel Create Database

The screenshot shows the AWS OpenAI VectorDB console with the "Workspaces" tab selected. At the top, there's a navigation bar with "Overview", "Workspaces", "Databases", "User Management", "Stage", "Preview", "Access", "Firewall", "Security", and "Updates". Below the navigation bar, there's a "Welcome to Workspaces" section with a brief description of what workspaces are. To the right of this, there are two boxes: "Workspaces COMPUTE + RAM" showing "Acme Workspace S-0" with 4 vCPUs and 32 GB RAM, and "Databases STORAGE" showing three databases with sizes 52GB, 32GB, and 12GB. Below these sections, there are two main tabs: "Workspaces" and "Databases". The "Workspaces" tab is active, showing a list with "workspace-1" (S-0 PROD). A "Connect" button with a dropdown arrow and a three-dot menu icon are next to it. The "Databases" tab is also visible, showing two databases: "s2_dataset_martech_4fbcd2058" and "WebexOneDB", each with an R/W icon. A blue arrow points from the "Databases" tab towards the "WebexOneDB" entry.

- Click on Database tab

The screenshot shows the AWS OpenAI VectorDB console with the "Databases" tab selected. At the top, there's a navigation bar with "Overview", "Workspaces", "Databases", "User Management", "Stage", "Preview", "Access", "Firewall", "Security", and "Updates". Below the navigation bar, there's a "MarTech Application Powered by Real-Time Analytics" section with a brief description of the application's purpose. A "Try the demo" button is present. To the right of this, there's a diagram illustrating the application flow: data from a "SingleStore" database is processed through a "MATCHING..." step, which then triggers a "NOTIFY" action on a map showing user locations. Below this diagram, there are two tabs: "Workspaces" and "Databases". The "Databases" tab is active, showing the same two databases as the previous screenshot: "s2_dataset_martech_4fbcd2058" and "WebexOneDB", each with an R/W icon. A blue arrow points from the "Workspaces" tab towards the "WebexOneDB" entry.

No tables or data yet

Name	Status	Table Count	Memory Usage	Index Disk Usage
WebexOneDB	ATTACHED	0	0 B	0 B
s2_dataset_martech_4fbdb2058	ATTACHED	12	91 MB	20 MB
information_schema	ATTACHED	198	-	-

- Return to the home screen and navigate to Develop > Data Studio. Open the SQL Editor.

1

2

- Be sure to select your workspace and database.

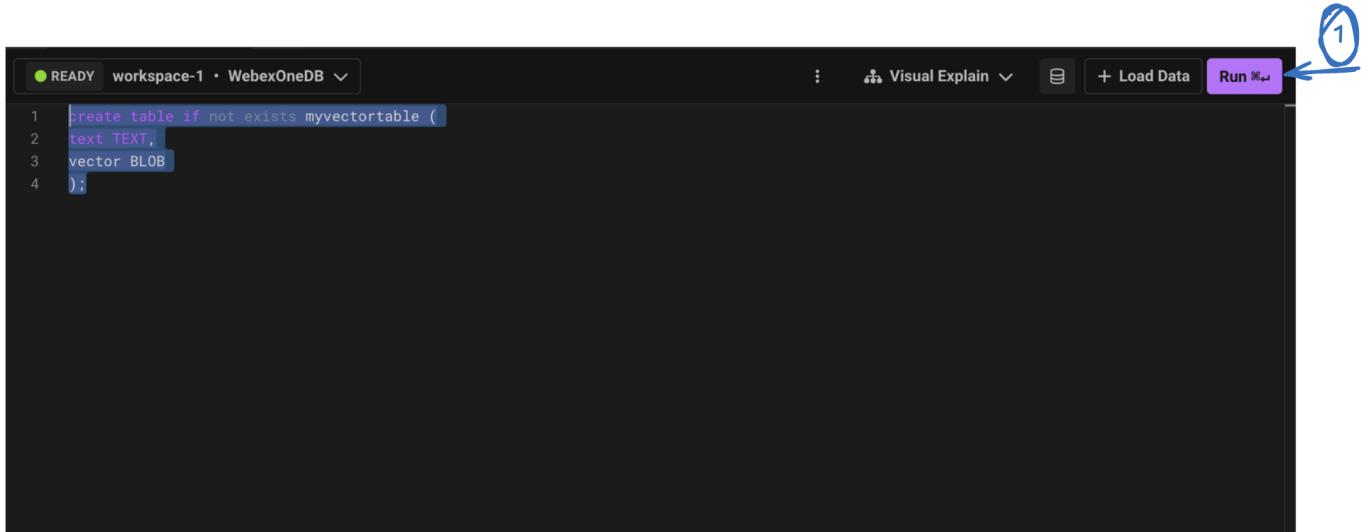
Your Workspace Your Database

READY workspace-1 • WebexOneDB

Connection: workspace-1 Database: WebexOneDB

- Run a SQL command to create a table - Press Run

```
create table if not exists myvectortable (
text TEXT,
vector BLOB
);
```

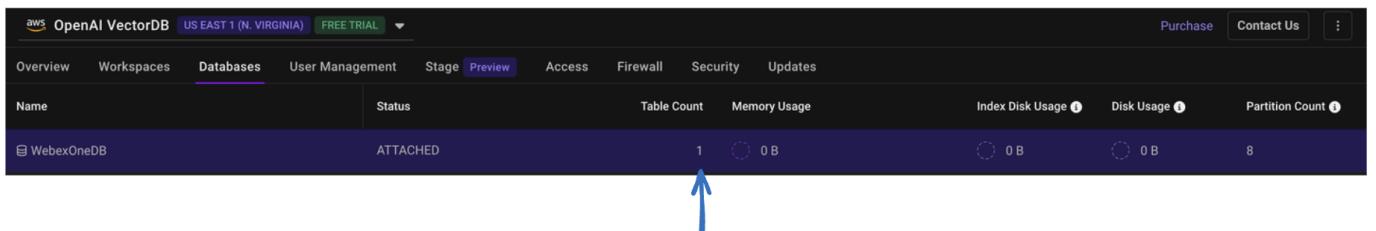


```

1 create table if not exists myvectortable (
2     text TEXT,
3     vector BLOB
4 );

```

- You can navigate to the Databases tab and verify that the table has been created.



Name	Status	Table Count	Memory Usage	Index Disk Usage	Disk Usage	Partition Count
WebexOneDB	ATTACHED	1	0 B	0 B	0 B	8

- Let's copy the embeddings we generated earlier using Postman so that we can insert them into our database.
- To proceed, copy the input text and all the embedding values, including the square brackets [], as shown in the image. These will be used for insertion into our database.

Copy Input

```

1 {
2   "input": "What is WebexOne 2024? WebexOne is an annual in-person and virtual event that takes place over four days. It's an event focused on AI collaboration and customer experience, and it features a range of activities such as insightful breakout sessions, technical training courses, hands-on labs, inspiring keynotes, epic entertainment, a solutions showcase and expo, customer awards, meet the experts, 1:1 executive meetings, a partner program, and more!",
3   "model": "text-embedding-ada-002"

```

Copy the vectors

```

1 {
2   "object": "list",
3   "data": [
4     {
5       "object": "embedding",
6       "index": 0,
7       "embedding": [
8         -0.0042485874,
9         -0.02296605,
10        0.011213377,
11        0.01114761,
12        -0.027280405,
13        0.0023528363,
14        -0.004350527,
15        0.0043768343,
16        -0.00068562734,
17        -0.011660597,
18        0.010397859,
19        0.018730614,
20        -0.0158763,
21        -0.013469206,
22        0.0026997605,

```

- Let's head back to our SQL editor and insert the values into our database. Once done press "Run"

```
insert into myvectortable (text ,vector) values ("your_input_text", JSON_ARRAY_PACK("your_embeddings"))
```

In this query:

```
* Replace "your_input_text" with the input values from Postman.
* Replace your_embeddings with the embeddings you copied earlier, including the square brackets [].

insert into myvectortable (text ,vector) values ("What is WebexOne 2024? WebexOne is an annual in-person and virtual event that takes place over four days. It's an event focused on AI collaboration and customer experience, and it features a range of activities such as insightful breakout sessions, technical training courses, hands-on labs, inspiring keynotes, epic entertainment, a solutions showcase and expo, customer awards, meet the experts, 1:1 executive meetings, a partner program, and more!", JSON_ARRAY_PACK([
-0.0042485874,
-0.02296605,
0.011213377,
0.01114761,
-0.027280405,
0.0023528363,
-0.004350527,
0.0043768343,
-0.00068562734,
-0.011660597,
0.010397859,
0.018730614,
-0.0158763,
-0.013469206,
0.0026997605,
-0.00003768792,
0.014850326,
-0.03183152,
0.0026175508,
-0.0459321,
-0.0055803815,
-0.008247258,
-0.0117313211,
0.0006169824,
-0.017112732,
-0.002885554,
-0.004113764,
-0.01348236,
```

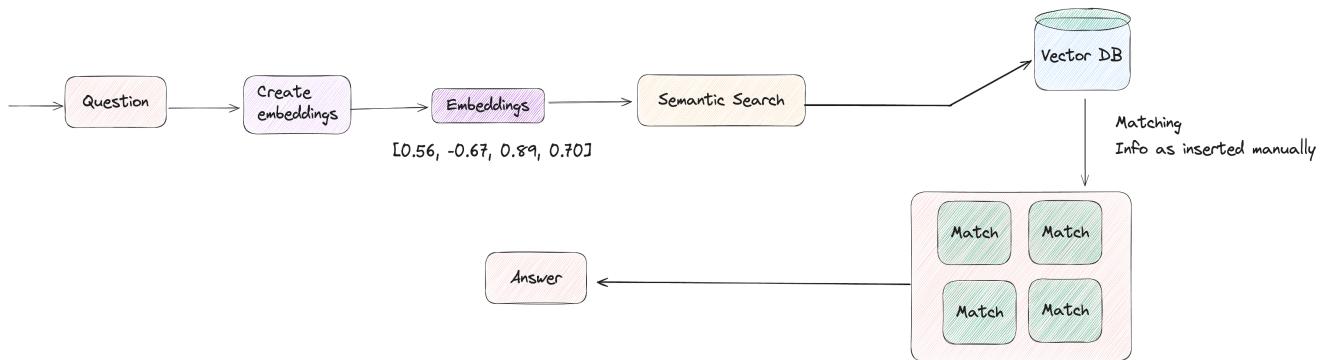
```
    .....
]) )
```

- After running the above command, you will see that our table now contains both the input text and the corresponding embeddings.

The screenshot shows the OpenAI VectorDB interface. At the top, there's a navigation bar with 'OpenAI VectorDB', 'WebexOneDB', and 'myvectortable'. Below the navigation are tabs for 'Columns' (2), 'Indexes' (0), 'Sample Data' (1), and 'SQL'. The main area displays a table with two columns: 'text' and 'vector'. The 'text' column contains the query 'What is WebexOne 2024? WebexOne is an annual in-person and virtual event that takes place over four days. It's an ...' and its corresponding vector representation. A blue circle labeled '1' points to the 'Sample Data' button.

RETRIEVING VALUES FROM THE VECTOR DATABASE

High Level Overview



- Searching the vector database is quite simple. Example: we want to find anything related to WebexOne. To do this, we'll create embeddings for our query, "Is WebexOne an annual event?" and then search the vector database to find matches against the existing embeddings.
- Let's open Postman and create an embedding for our question. Be sure to click on "Send."

The screenshot shows a POST request to <https://api.openai.com/v1/embeddings>. The request body is a JSON object:

```

1 {
2   "input": "Is WebexOne an annual event?",
3   "model": "text-embedding-ada-002"
4 }
5
    
```

The response is a JSON array of embeddings:

```

1 {
2   "object": "list",
3   "data": [
4     {
5       "object": "embedding",
6       "index": 0,
7       "embedding": [
8         -0.0015118581,
9         -0.021267666,
10        0.023597047,
11        -0.0072638104,
12        -0.02286653,
13        0.005909599,
14        -0.026119394,
15        -0.010268575,
16        0.0052169873,
17        -0.009551842,
18        0.024065679,
19        0.020178782,
20        0.015464887,
21        0.016815653,
22        -0.016333235
      ]
    }
  ]
}
    
```

A vertical blue line labeled "Embeddings" is drawn next to the "embedding" field in the response data.

- Let's head over to the SQL editor and run a query for our search.

```

select text,dot_product(vector,JSON_ARRAY_PACK("[your_embeddings]")) as score
from myvectortable
order by score desc
limit 5;
    
```

* In this query: * Replace your_embeddings with the embeddings you copied earlier, including the square brackets [] for the question.

```

select text,dot_product(vector,JSON_ARRAY_PACK([
  -0.0015118581,
  -0.021267666,
  0.023597047,
  -0.0072638104,
  -0.02286653,
  0.005909599,
  -0.026119394,
  -0.010268575,
  0.0052169873,
  -0.009551842,
  0.024065679,
  0.020178782,
  0.015464887,
  0.016815653,
  -0.016333235,
  -0.0006641838,
  0.018069934,
  -0.010564916,
  0.007911626,
  -0.026367495,
  -0.0019417254,
  -0.005547787,
  -0.012604848,
  0.0003715036,
  -0.013962504,
  0.011729606,
  -0.00078737224,
  -0.011881223,
  0.007815143,
  0.009992908,
])
    
```

```

0.020316616,
-0.0032580325,
-0.0066401153,
0.0097310245,
-0.0028652078,
-0.008145943,
-0.0074843434,
.....
]) as score
from myvectortable
order by score desc
limit 5;

```

The screenshot shows a SQL Editor interface with a dark theme. At the top, there's a toolbar with icons for 'SQL Editor', 'Visual Explain', 'Load Data', and a 'Run' button (circled with '1'). Below the toolbar is a code editor window containing a SELECT query. The results are displayed in a table below, with the first row showing the text 'What is WebexOne 2024? WebexOne is an annual in-person and virtual event that takes place over four days. It's an event focused on AI collaboration and customer ex...' and a score of '0.8964636921882629'. The bottom of the screen shows a message log and various export options like 'Save as CSV', 'Save to Stage', and 'Copy to Clipboard'.

text	score
What is WebexOne 2024? WebexOne is an annual in-person and virtual event that takes place over four days. It's an event focused on AI collaboration and customer ex...	0.8964636921882629

Note: You'll be able to see the success of the vector database—higher scores indicate better matches for the answer.

1.5.4 Conclusion

In summary, vector databases allow LLMs to have long-term memory. In this section, we explored how to use embeddings and vector databases by generating embeddings with the OpenAI API through Postman. After setting up a free SingleStore account, embeddings were stored in a vector database. The process included creating an embedding for a query, searching the database, and confirming that it effectively retrieved relevant information. This demonstrated how vector databases can efficiently manage and query embeddings, making it easier to find relevant information based on the data stored.

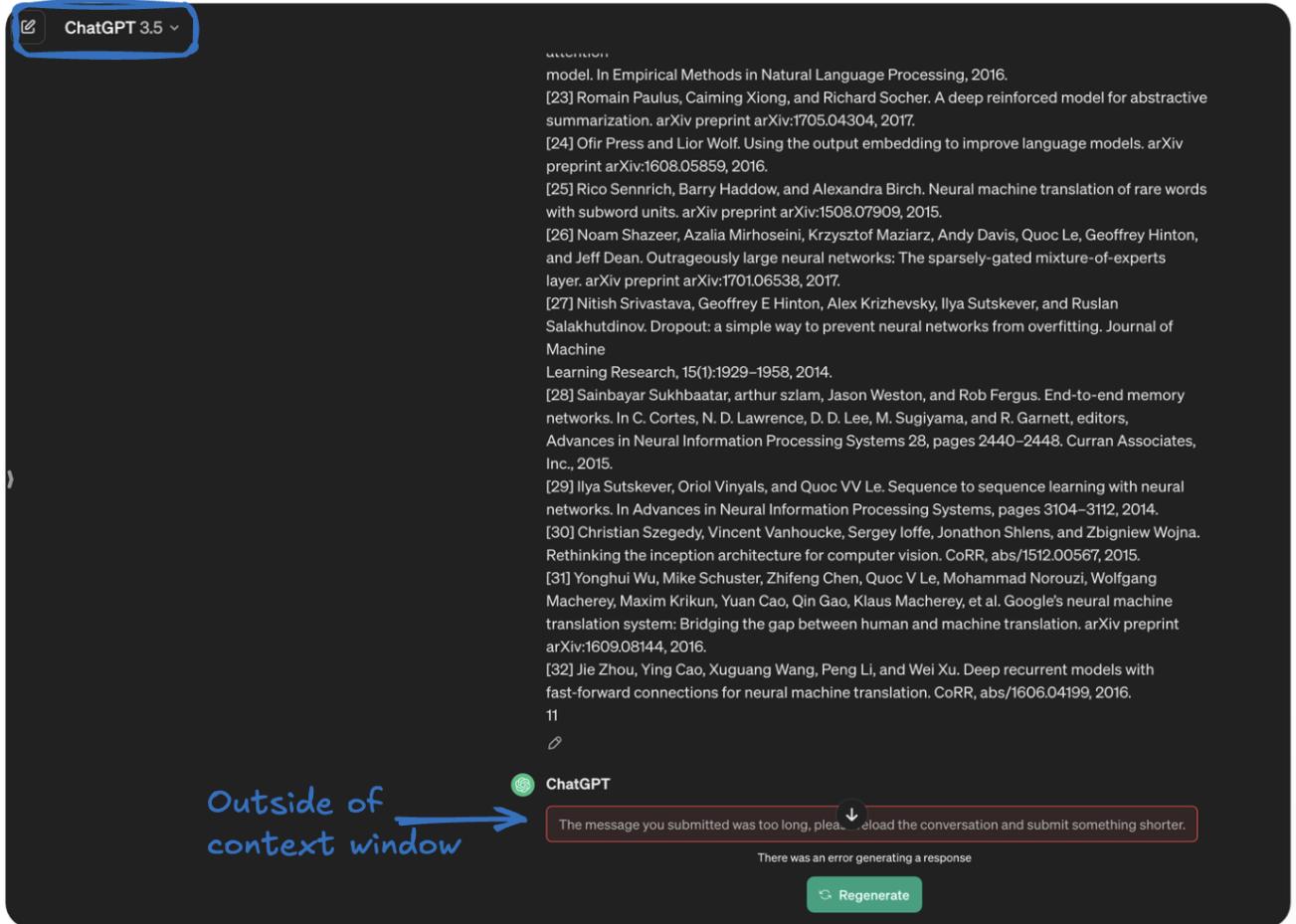
1.6 Task 5: Context Windows and Retrieval Augmented Generation (RAG)

What is Context window

- The context window is the limit on how much text the model can keep in mind and process at once. It includes your messages, the model's replies, and any other text in the conversation. If the conversation gets too long and goes beyond this limit, the model might start to forget what was said earlier.
- Example
- For GPT-3.5, the context window limit is approximately 8,000 tokens (one token typically represents about 4 characters on average.). If the input text along with previous interactions (if any) exceeds the limit of LLM, the model may not be able to see or remember parts of the text that fall outside this window. This can result in errors or the inability to refer back to earlier parts of the conversation or document.

Context Window

Model Context Window = 8K

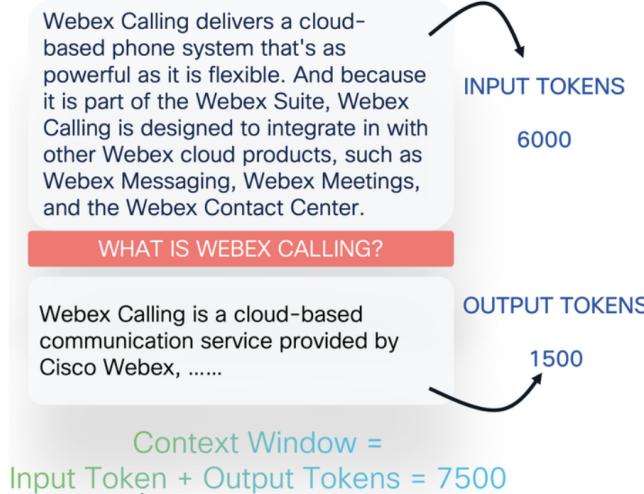


Note: A token can be as small as a piece of a word or as large as a word itself, depending on the language and complexity of the text

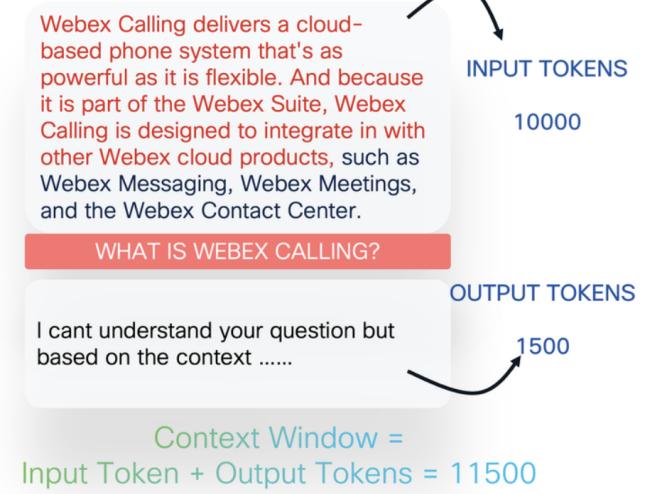
- Consider a model with a context window limit of 8,000 tokens. If we input 6,000 tokens and the model generates a response of 1,500 tokens, the total token count is 7,500. This is within the model's context window, so all information is processed correctly. Conversely, if the input increases to 10,000 tokens and the response remains at 1,500 tokens, the total becomes 11,500 tokens. This exceeds the model's 8,000-token limit. Consequently, the model might lose context or be unable to respond properly as the excess information falls outside its context window. This demonstrates the importance of managing input size to stay within the model's processing capabilities.

Context Window

Model Context Window = 8K



Model Context Window = 8K



1.6.1 How Context window work

- In the below example , multiple potential predicted words exist, but the model chooses the words based on the surrounding context and their probability. However, if the context window size increases and the word "Webex" moves out of the window, the model may lose critical context. As a result, the next predicted word might be incorrect, as the model can no longer reference "Webex" to inform its predictions accurately. This highlights the importance of keeping key information within the context window to maintain the accuracy of the model's predictions.
- Additionally, the temperature setting in the model can influence the type of content it generates. Lower temperatures are useful for summarizing or generating more deterministic and concise outputs, while higher temperatures encourage creativity, making the model more suitable for writing stories or poems. Adjusting the temperature allows you to control the balance between predictability and creativity in the model's output.

1	2	3	4	5	6	7	8	Predicted word
					Webex	Calling	is	cloud
				Webex	Calling	is	cloud	based
			Webex	Calling	is	cloud	based	telephony
		Webex	Calling	is	cloud	based	telephony	solution
	Webex	Calling	is	cloud	based	telephony	solution	that
Webex	Calling	is	cloud	based	telephony	solution	that	offers
Webex	Calling	is	cloud	based	telephony	solution	that	offers
Webex	Calling	is	cloud	based	telephony	solution	that	enterprise

How can we address the challenges related to the context window?

- That's where RAG comes into play

1.6.2 Retrieval Augmented Generation (RAG)

Lets try to understand what RAGs are all about before we can look how they can solve context window issues

RAG = Retrieval Augmented Generation

RAG = Retrieval Augmented Generation

Lets focus on the **Generation** part. Generation basically means responding to user query. For example, you might ask the model whether DX or Navigators are compatible with supporting ThousandEyes.

Retrieval Augmented Generation

Generation



Response to user
Query also known as
Prompt



Can have some
undesirable behavior

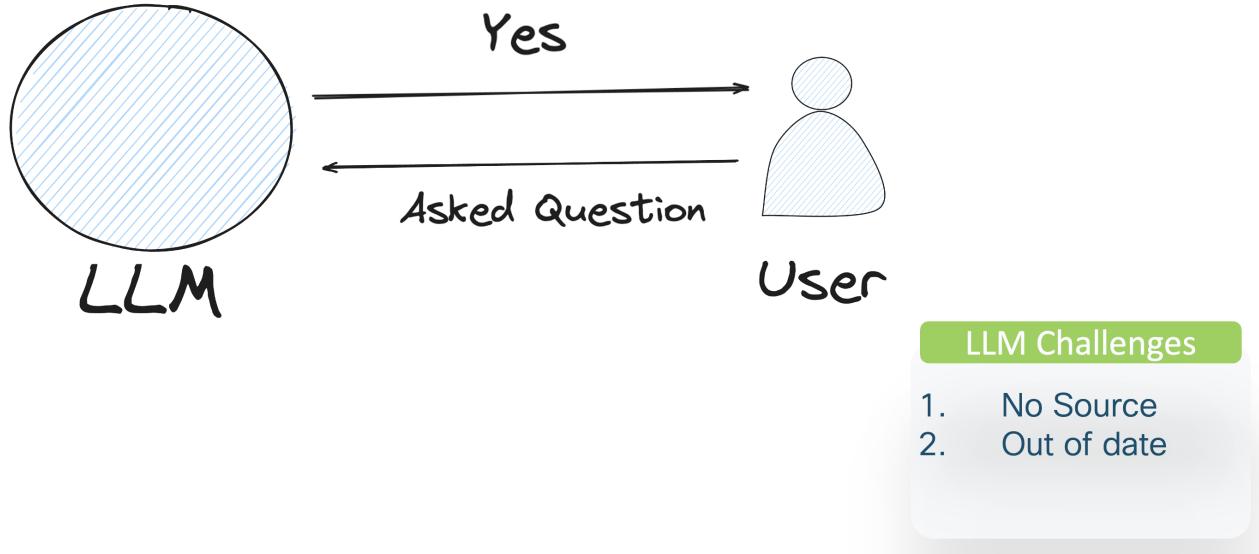


MTR?

ThousandEyes?

Source

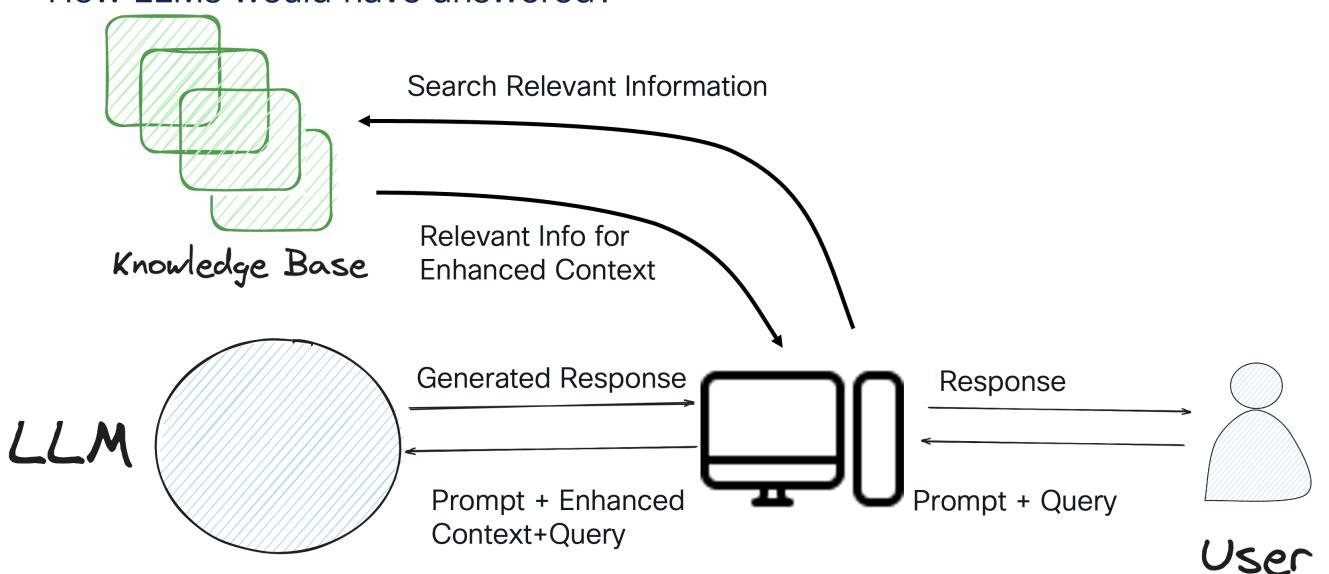
That's an excellent question. LLMs are trained on vast amounts of historical data, so they may recognize that DX and Navigators are well-regarded Cisco products. As a result, the LLM might confidently respond that you can install ThousandEyes on those devices. However, the issue is that the response may lack a verifiable source to back up that claim or the information can be out-of-date.



What if, instead of just asking the question, I had first consulted a reputable source and only then informed the user that ThousandEyes can't be installed on those devices? This is where Retrieval-Augmented Generation comes into play. By integrating a knowledge base—such as PDFs, CSVs, images, and other resources—into the model, we ensure that any time a question is asked, the response is based on the most current and accurate information available.

Retrieval Augmented Generation

How LLMs would have answered?



1.6.3 How can RAG address the limitations we've seen with the context window?

To address the context window limitation, we can use an embedding model to convert relevant text into embeddings and store them in a vector database. When a user submits a query, we transform it into an embedding and compare it to the stored embeddings to find the closest match.

Once we identify the match, we retrieve that specific chunk of text to respond to the query. This approach effectively overcomes the context window limitation by ensuring that our responses are accurate and contextually relevant.

RAG

Webex Calling delivers a cloud-based phone system that's as powerful as it is flexible. And because it is part of the Webex Suite, Webex Calling is designed to integrate in with other Webex cloud products, such as Webex Messaging, Webex Meetings, and the Webex Contact Center.

vector embeddings

```
[ -0.011466463, -0.02045446, -0.012214276, -0.005793769, -0.00562284, 0.012370961, -0.0033829627, 0.017833555, 0.0056691333, -0.033388063, 0.027633464, 0.011195826, 0.018887615, -0.010939433, -0.015568751, 0.009906739]
```



Compare embedded query to embedded text and find the closest match
Return closest chunk of text into prompt

User Query to embeddings

What is Webex Calling?

[-0.011466463, -0.02045446, -0.012214276, -0.005793769]



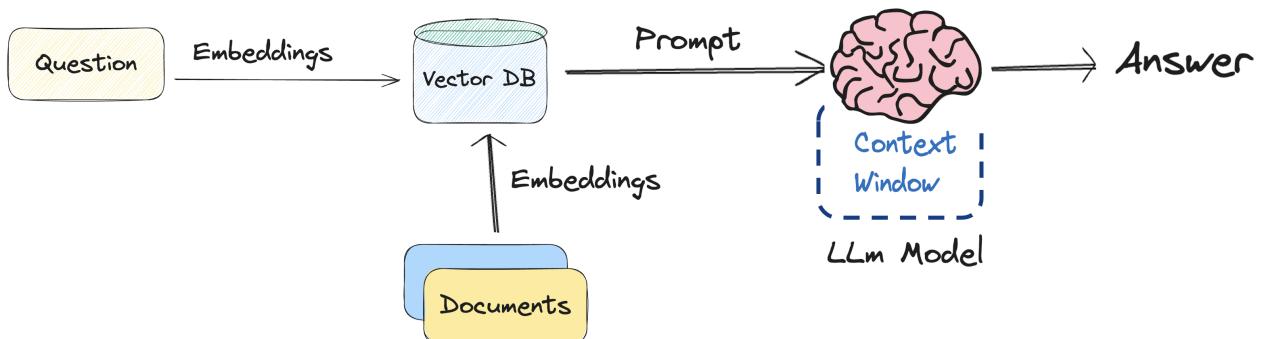
Webex Calling service provided by Cisco Webex,

What is Webex Calling?

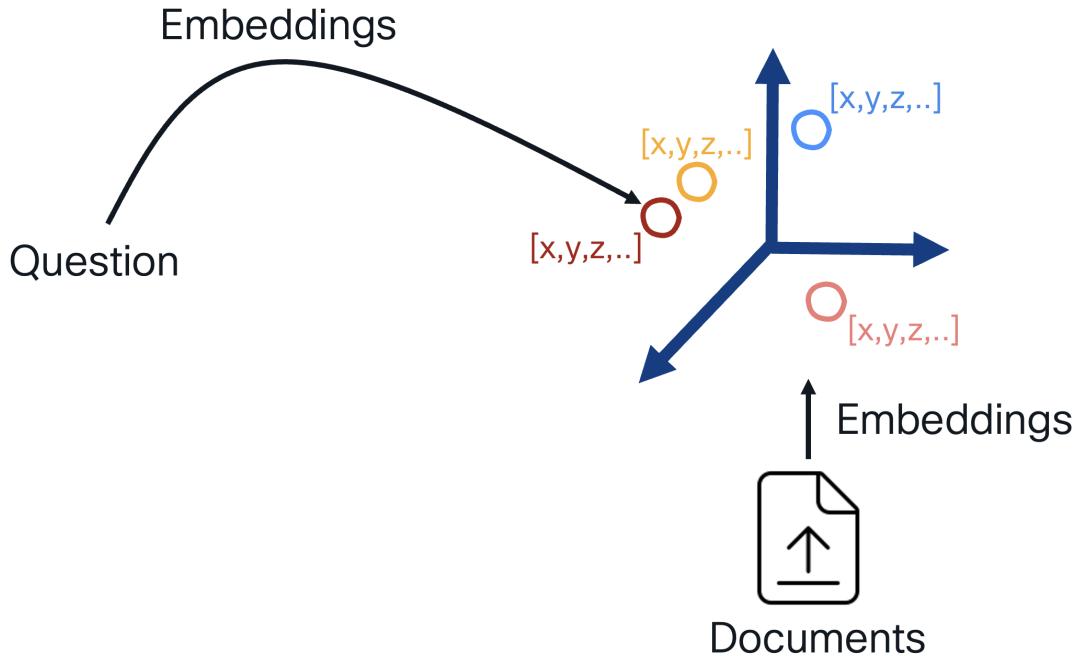
Webex Calling is a cloud-based communication service provided by Cisco Webex,

Answer the user query with fetched content

Now you can see how RAG can address some of the challenges with LLMs. Instead of re-training or fine-tuning the model with new information, we can augment our data sources to retrieve the most up-to-date information. Additionally, LLMs are now instructed to prioritize primary source data before generating responses, making them less likely to hallucinate and encouraging more accurate and reliable outputs.



1.6.4 What's Happening Under the Hood



We normally start by taking multiple documents and converting them into embeddings. Imagine these embeddings as points in a 3D space, where each document is projected based on its semantic meaning or content. Documents located near each other in this space contain similar semantic information, which is crucial when we search for or retrieve information.

Now, when we receive a query, we also convert it into an embedding (numerical value). We then perform a similarity search in this 3D space, looking for the documents that are closest to our query's embedding. These nearby documents are likely to contain the most relevant information.

Once we've identified these matching documents, we retrieve the relevant sections (or "splits") and feed all this information into the LLM's context window, allowing it to generate a well-informed response.

Large context window size. Do we still need RAG?

We've seen models like Gemini Pro with a context window size of millions, allowing them to retain more information. So, why use RAG? The answer is that RAG remains highly beneficial for sourcing real-time data, fact-checking, and accessing external knowledge that isn't contained within the context window.

1.7 Analyzing ThousandEyes EPA Data

Typically when a user experiences an issue they will either suffer in silence, wait for the issue to go away and when it crops up again get frustrated and call in for help. Sometimes while they are working with support or the helpdesk the issue resolves itself. This makes it really difficult to troubleshoot issues if they aren't persistent. ThousandEyes stores 30 days of data so you have the ability to go back in time and see what was occurring or see if this issue has been occurring and the user just didn't call into support.

1.7.1 Code Annotation

CODEBLOCKS

Some `Code` will be below:

A plain Code Block

```
Some Code here
print(A)
```

`omer.py`

```
Some Code here
print(A)
```

```
1 Some Code here
2
3 print(A)
```

```
Some Code here
print(A)
```

If you are troubleshooting an issue and want to quickly see all the test data for a user's system Agent Views provides the perfect visualization.

Navigate to VM session and browse to ThousandEyes portal. Open Endpoint Agents > Agent Views (be sure to use the incognito browser). Use the search box to find your agent which will be what you named your VM. Then you can begin to explore the metrics and tests. The top section contains your system's performance metrics.

- Use the Metrics pull down to view the different system metrics

The screenshot shows the ThousandEyes Endpoint Agents interface. On the left, a sidebar lists various sections: Cloud & Enterprise Agents, Endpoint Agents (selected), Overview, Views, Agent Views (highlighted with a red arrow), Monitoring Settings, Agent Settings, Devices, Internet Insights, Dashboards, Alerts (with 6 notifications), Integrations, Sharing, and Account Settings. The main content area is titled 'ROADSHOW1'. It displays basic agent information: Hostname (ROADSHOW1), Location (City of London, England, UK), Agent Version (1.169.3), Private IP Address (198.18.133.36), Public IP Address (64.103.46.1), Users (ROADSHOW1\Administrator), Labels (None), License (Advantage). A search bar at the top right contains the text 'road' with a red arrow pointing to it. To the right of the search bar is a 'Metrics' dropdown set to '8 of 17 selected'. Below this are several performance metrics: System (Memory 41.6%, CPU Load 18.8%), Network Access (Link Speed 1 Gbps), and Gateway (Gateway Latency < 1 ms, Gateway Loss 0%). A large sidebar on the right contains a 'Metrics' section with checkboxes for CPU Load, Memory, Signal Quality, Link Speed, Throughput, Retransmission Rate, Roaming Events, Channel Swap Events, Gateway Loss, Gateway Latency, VPN Loss, VPN Latency, and Proxy.

The middle section shows Automated Session Tests (ASTs) and Scheduled Tests (up to 10 max).

- Use the pull down to view the different web and network metrics.

The screenshot shows the 'Automated Session Tests' and 'Scheduled Tests' sections. The 'Automated Session Tests' section lists a single test for 'Webex' with a 'Loss' of 0.3%. The 'Scheduled Tests' section lists a single test for 'Bernie-O365' with a 'Loss' of 0.1%. Both sections have a 'Metric' dropdown menu with 'Loss' selected. A large sidebar on the right is titled 'Network' and includes 'Loss', 'Latency', 'Jitter', and 'TCP Connection Failures'. Another sidebar titled 'Web' includes 'Availability', 'Response Time', and 'Throughput'.

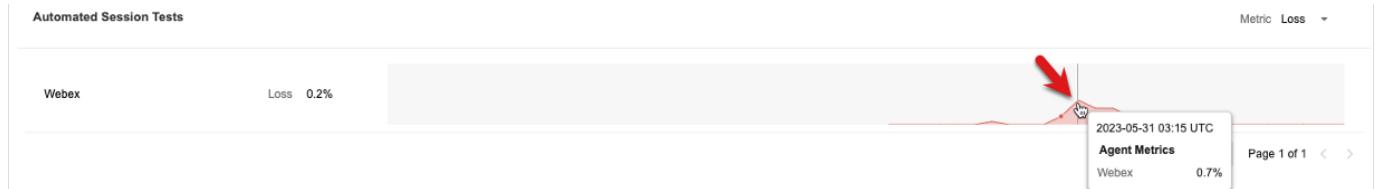
The bottom section shows browser sessions. If you don't see anything you might need to refresh the tabs in your VM's browser a few extra times or refresh the Agent Views page.

- Use the pull down to view the different browser and network metrics.

The screenshot shows the 'Browser Sessions' section. It lists three sessions: 'www.webex.com' (Experience Score 94.5%), 'www.cisco.com' (Experience Score 100%), and 'www.office.com' (Experience Score 100%). The 'Metric' dropdown menu is open, showing 'Experience Score' selected. A large sidebar on the right is titled 'Web' and includes 'Experience Score', 'Response Time', 'Page Load Time', and 'Browser Errors'. Another sidebar titled 'Network' includes 'Loss', 'Latency', 'Jitter', and 'Connection Failures'.

1.7.2 Drill into a Automated Session Test

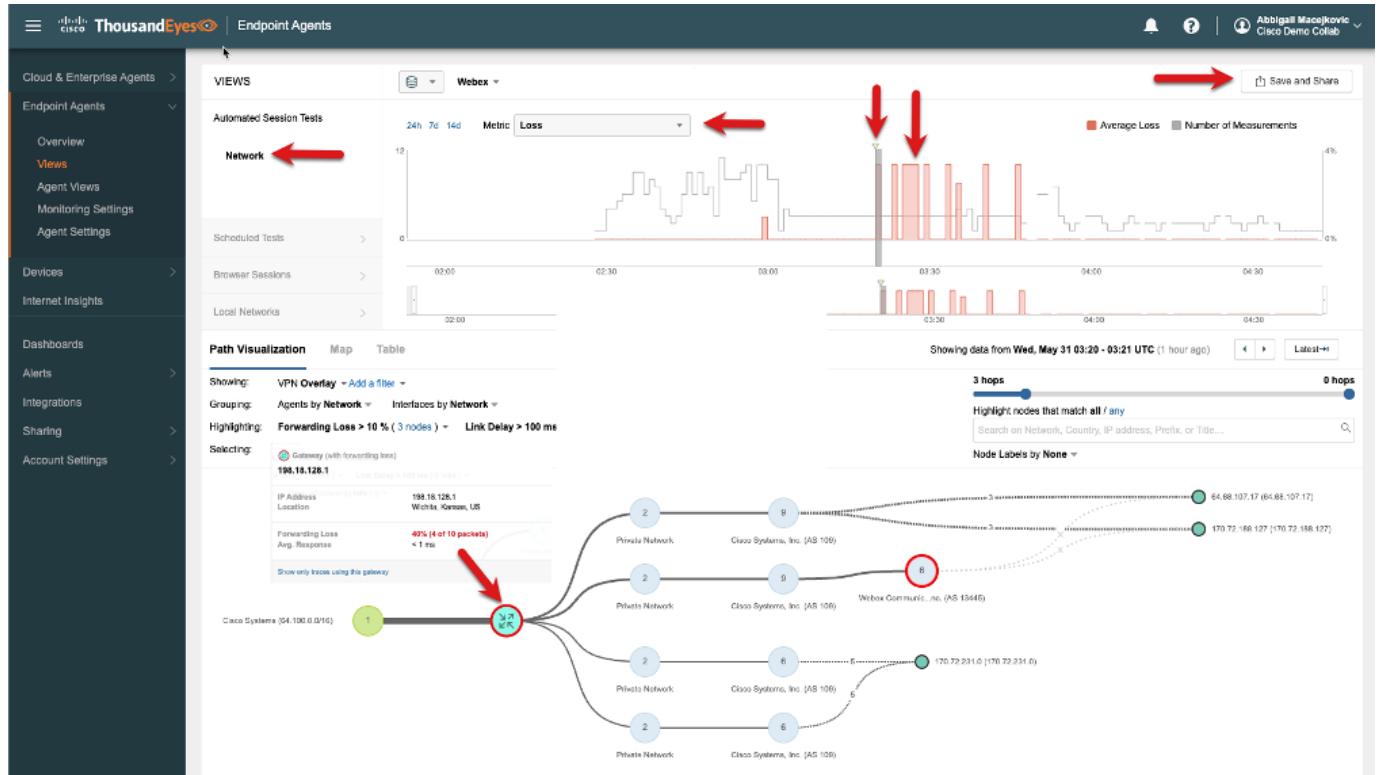
Now that you've explored the Agent Views page click on the AST metric to dive into the detailed view associated with the Webex Test. If you don't have any network loss try changing the metric to latency.



You will now see a filtered view for your agent showing the network path which is dynamically created based on the endpoints your Webex session is connected to while the Webex session is running.

Here's an example view. I used the upper right Save and Share feature to create a snapshot that anyone can view for a full year! This can be very helpful as it can be attached to tickets, shared with the end user, another team or a service provider. They will see what you see making it simple to collaborate and reduce any finger pointing. (think back to the last slide in the ThousandEyes Overview).

- Open the snapshot in another browser tab
- You can mouse over the nodes for extra information
- Try changing the metric and drag over and navigate the time bar to see how the test changes



From this same view you can change the grouping to show IP Addresses and adjust the number of hops to get more a granular hop by hop network path.

- You can mouse over the nodes and links for extra information.
- You can also adjust the link delay. In the below screenshot used 10ms. You can also click on the 8 links to see all links highlighted that are over 10ms.



Feel free to navigate back to your Automated Session Test view showing your agent and click around to experiment with how to see similar data.

1.7.3 Drill into Scheduled Tests

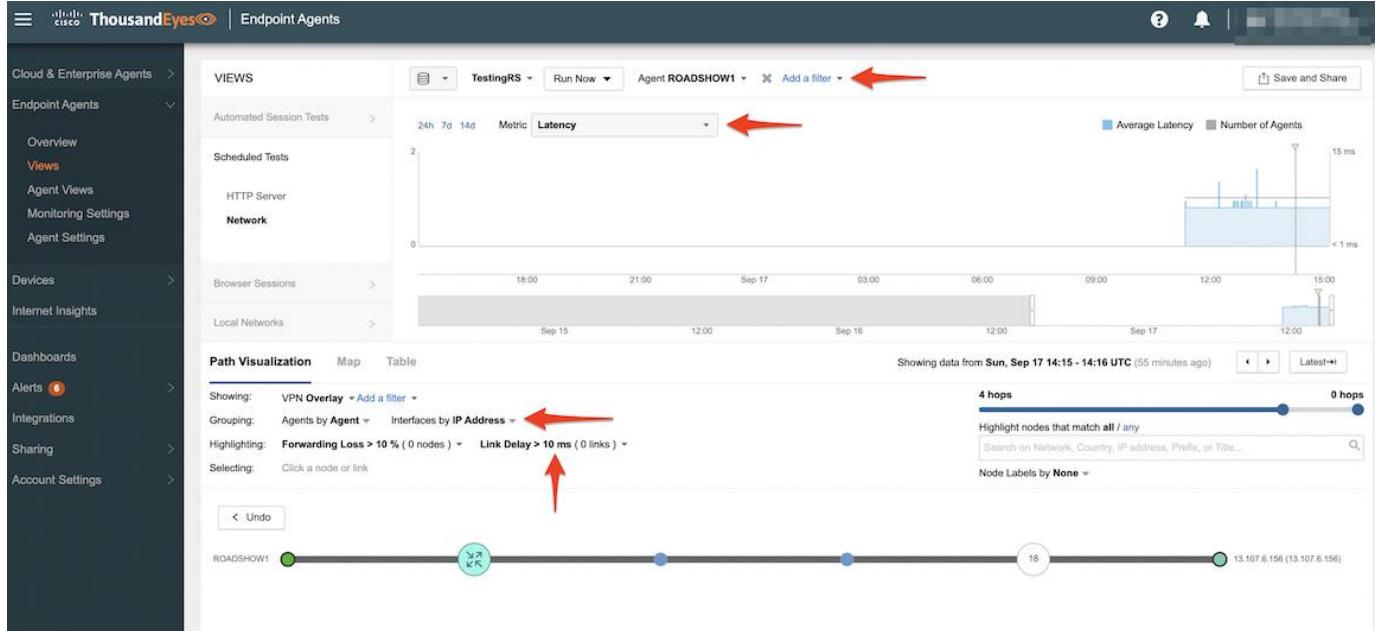
Scheduled tests provide a consistent baseline. Whenever an Endpoint Agent is online the test runs at the specified interval. Navigate back to Agent Views and search and select your agent.

- Click Agent Views and use the search box to find your agent.
- Click into the scheduled test to drill into a filtered view. (If you set up other scheduled tests feel free to use one of them instead)

The screenshot shows the ThousandEyes Agent Views interface. The left sidebar has a dark theme with navigation items: Cloud & Enterprise Agents, Endpoint Agents (selected), Overview, Views, Agent Views (highlighted with a red arrow), Monitoring Settings, Agent Settings, Devices, Internet Insights, Dashboards, Alerts (6 notifications), Integrations, Sharing, Account Settings, and a feedback box 'I wish this page would...'. The main content area shows details for 'ROADSHOW1' endpoint agent, including its location (City of London, England, UK), agent version (1.169.3), and private/public IP addresses. It also displays 'Local Networks' metrics (System: Memory 42%, CPU Load 19.9%; Network Access: Link Speed 1 Gbps) and 'Automated Session Tests' (Webex: Loss 0%). A red arrow points to the 'Scheduled Tests' section, which lists a single entry: 'TestingRS https://office.com Loss 0%'. At the bottom right, there are pagination controls 'Show 5' and 'Page 1 of 1'.

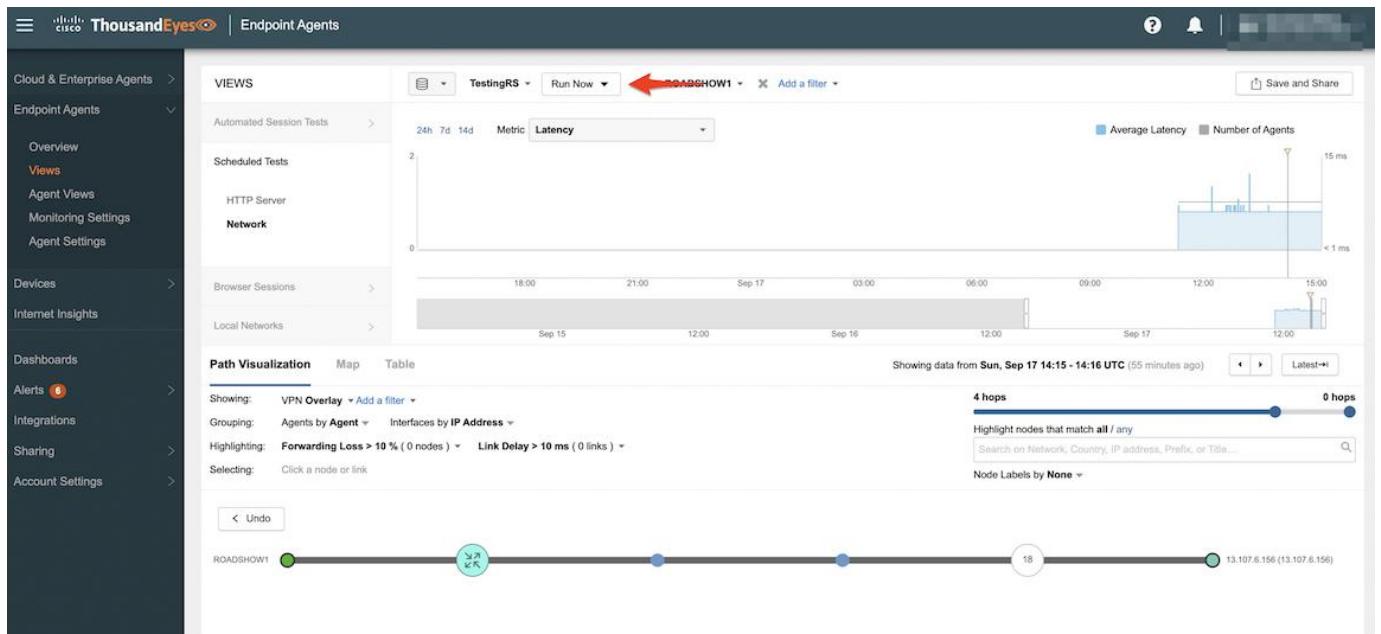
Now you're in a filtered test view. Note the filters at the top of the screen showing the test and your agent. See the below screen shot as an example.

- Adjust the metric to latency and change the grouping to IP Address
- Adjust the hops to max using the slider
- Set the link delay to 10ms and then click on the links to highlight them in the network path



If you wanted to see all agents that are running the scheduled test you could just remove the agent filter (this won't work in this lab as we have only one agent). This would be a great way to figure out if an issue is effecting multiple users or start to isolate the users that are having issues as they might all be in the same building or connecting through the same router that is causing congestion, latency or loss. You can also right click on the agent to pull up the traceroute which is run every time the test is executed.

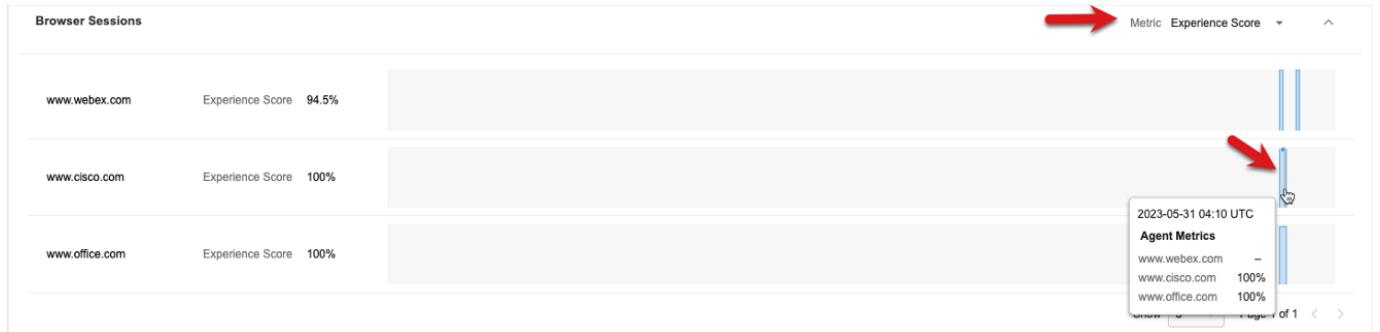
Another powerful feature is the Run Now option for when you are actively troubleshooting and need test results right away or you can make changes and run it. Feel free to test it out.



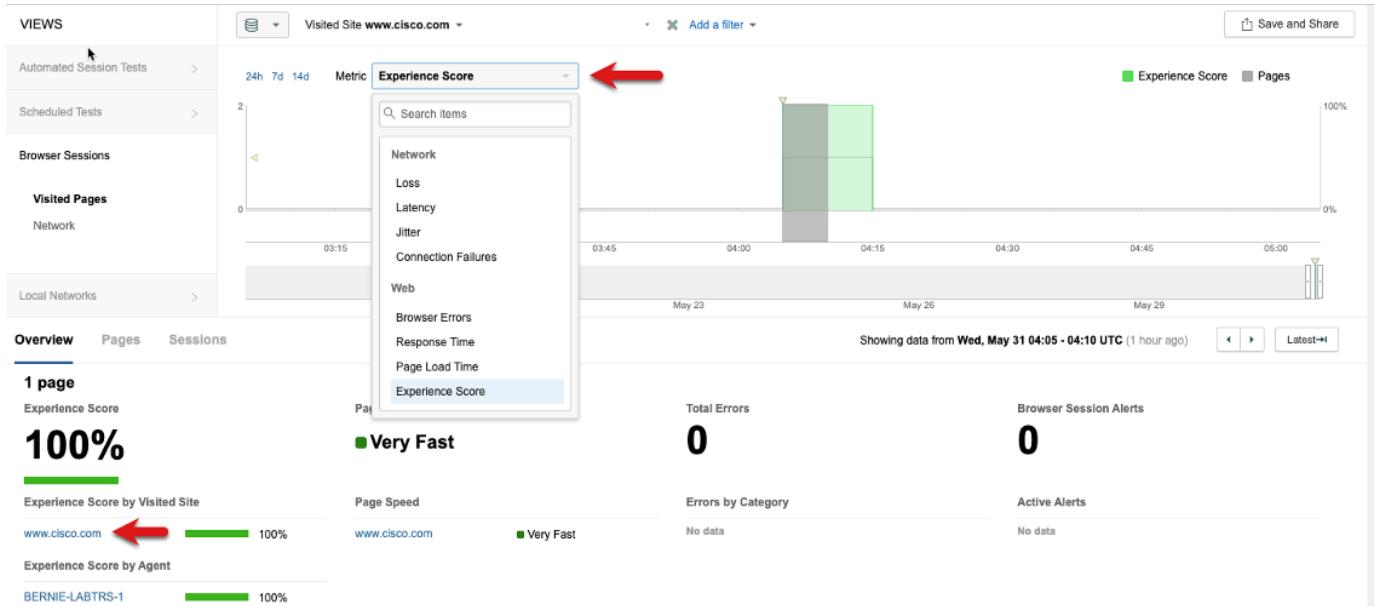
Note: It might take a few minutes for the test to complete and then be processed for visualization. So you may want to kick it off and then continue on with the lab and come back to the tab in a few minutes.

1.7.4 Drill into Browser Sessions

Browser Sessions are generated using the browser plugin and will monitor the user experience and network path for your monitored domain set. The experience score is a calculated metric to help gauge the end users web performance. You can change the metric based on what you are wanting to use for troubleshooting. Navigate back to Agent Views and search and select your agent. Click on the www.cisco.com session to drill into a filtered view. Note if your browser sessions aren't showing up, verify your agent is associated with the agent label and then make sure to go back to the browser and refresh the tabs.



The browser session view shows the visited pages and experience score. See the below screenshot for an example. Note the filter for the Visited Site and Agent which was automatically applied based on the browser session test you clicked from the Agent View.

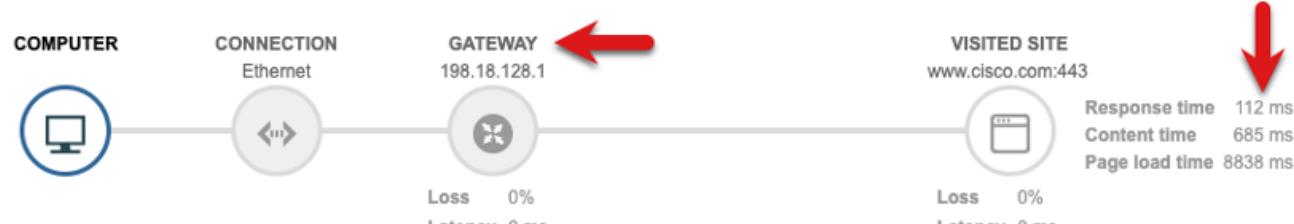


You can easily explore the other metrics from the session. Click on the web site to bring up the session, system and network details and review the tabs for the path trace and waterfall details. Additional metric will be pulled in based on how the Endpoint Agent is connected to the network like wireless, vpn or proxy.

(unknown)
https://www.cisco.com/

1 of 1 page 

Experience Score	Agent	Time	
100%	Visited Site www.cisco.com	2023-05-31 04:06:13 UTC	Page Speed Very Fast
	Session ID 1685505900:zTtBWRkp		Errors —

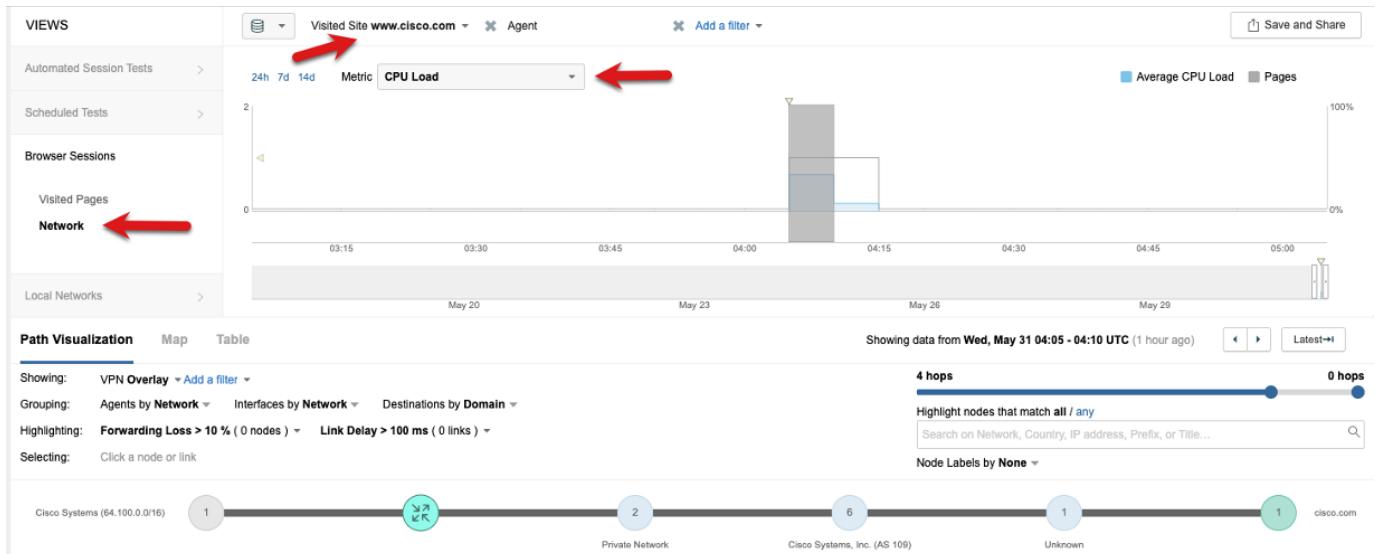


Computer Info	Path Trace	Waterfall
BERNIE-LABTRS-1		
Model VMware Virtual Platform	CPU 34%	
Manufacturer VMware, Inc.	Browser Google Chrome (113.0.0.0)	
OS Version Microsoft Windows 10 Pro	Public IP Address 64.100.12.5	
Kernel 10.0.19045	Private IP Address 198.18.133.36	
Memory 3278 MB / 8191 MB (40%)	Private Subnet Mask 255.255.255.0	
Endpoint Agent Version 1.158.1	DNS Servers 8.8.8.8	

Computer Info		Path Trace	Waterfall
Trace from Computer to 104.106.160.119			
Name (IP Address)			Delay
1	198.18.128.1 (198.18.128.1)		0 ms
2	10.255.0.3 (10.255.0.3)		0 ms
3	10.1.27.9 (10.1.27.9)		0 ms
4	64.100.12.36 (64.100.12.36)		1 ms
5	rtp5-dmzaas-gw1-ten1-3.cisco.com (64.102.244.193)		1 ms
6	rtp10-cd-dmznet-gw1-ten2-3.cisco.com (64.102.244.181)		1 ms
7	rtp1-md1-dmzbb-gw1-twe2-0-25.cisco.com (64.102.254.237)		1 ms
8	64.102.255.145 (64.102.255.145)		2 ms
9	128.107.6.22 (128.107.6.22)		1 ms
10	a104-106-160-119.deploy.static.akamaitechnologies.com (104.106.160.119)		8 ms

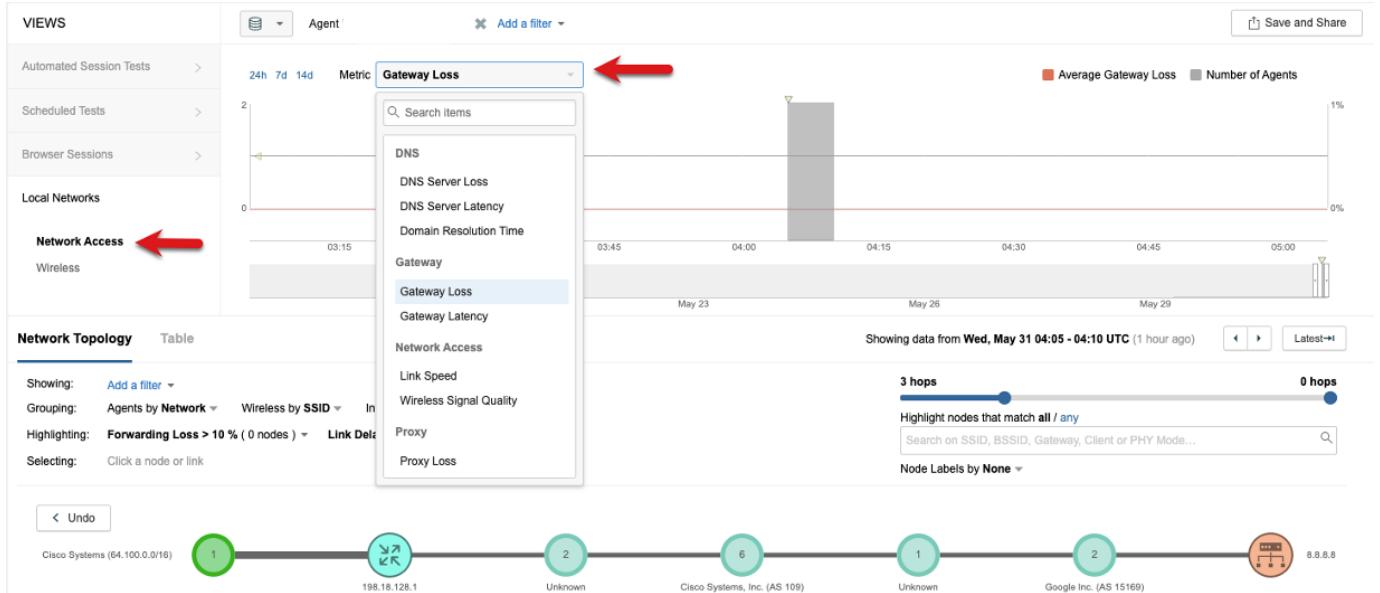
Computer Info		Path Trace	Waterfall				
https://www.cisco.com/							
https://www.cisco.com/							
Object	Response Code	Domain	Size (kB)	Waterfall ▲			
/	200 [Headers]	www.cisco.com	15	120 ms			
0	204 [Headers]	bat.bing.com		51 ms			
recording	200 [Headers]	k-aus1.clickta...	0	48 ms			
display	204 [Headers]	c.clicktale.net		62 ms			
events	204 [Headers]	c.clicktale.net		61 ms			
ctm-core.js	200 [Headers]	www.cisco.com	9.6	289 ms			
personal... plain.css	200 [Headers]	www.cisco.com	4.9	290 ms			
clientlib-d... clientlib-b...	200 [Headers]	www.cisco.com	4.9 (cached)	< 1 ms			
clientlib-h... ctm.js	200 [Headers]	www.cisco.com	9.9 (cached)	< 1 ms			
otSDKStu... segments...	200 [Headers]	cdn.cookielaw...	78.6 (cached)	< 1 ms			
UserProfile token.json	200 [Headers]	www.cisco.com	30.9	210 ms			
VisitorAPI.js	200 [Headers]	www.cisco.com	0.5	209 ms			
targeter-b... 01303338...	200 [Headers]	www.cisco.com	19	146 ms			
otBanner... root.html	200 [Headers]	cdn.cookielaw...	14.6	147 ms			
root.html	200 [Headers]	www.cisco.com	1.8 (cached)	< 1 ms			
root.html	200 [Headers]	www.cisco.com	99 (cached)	< 1 ms			
root.html	200 [Headers]	www.cisco.com	6.2	271 ms			
root.html	200 [Headers]	www.cisco.com	5	272 ms			
root.html	200 [Headers]	www.cisco.com	5.1	270 ms			

ThousandEyes takes a layered approach with visualizing the data to help you isolate and troubleshoot issues. The Network Layer will show the network path filter on the browser session. Click on the Network Layer that is associated with the Browser Session. Explore the different metrics that are captured in context in time like CPU Load.



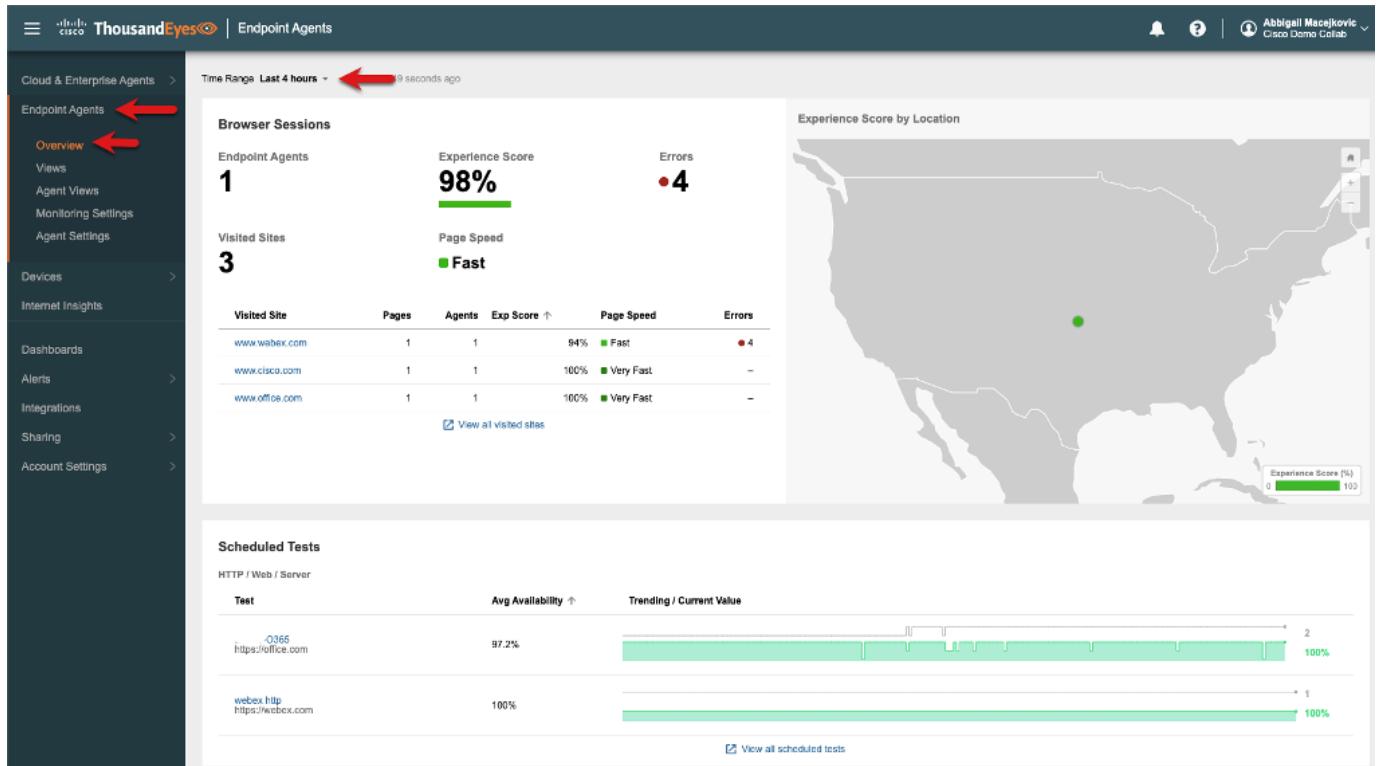
1.7.5 Local Network View

The Local Networks view will provide the local network and DNS information. This can be very helpful for finding bad DNS settings or Wi-Fi related issues. Click into the Local Networks > Network Access layer and make sure your agent is still in the agent filter or clear it out with the X to see all agents. Explore the different metrics as well using the pull down. If this was a production environment the filters could be used for isolation of issues.



1.7.6 Endpoint Agent Overview

This page provides a quick overall health view of your Endpoint Agents. Navigate to the Endpoint Agents > Overview to see all the stats from all the agents. Try adjusting the time range as well.



Now that you've learned about the metrics, tests and various way that ThousandEyes can help you troubleshoot, isolate and resolve end users issues you're one short step from the final stage of this learning journery. The last step will be learning about some of the ways a dashboard can be used to visualize the endpoint agent data and shared with other stakeholders. Check out Task 7 to learn about Endpoint Agent data in Dashboards.

1.8 Understanding Fine-Tuning - Large Language Models

1.8.1 Fine-Tuning LLMs

Goals: Equip audience with knowledge and practical skills in finetuning techniques accompanied by code examples. Discuss the approach to finetuning LLMs.

The section centers around fine-tuning LLMs, addressing their various aspects and methodologies. As the module progresses, the focus will be given to specialized instruction tuning techniques, namely LoRA. It will examine domain-specific applications (Webex Calling), ensuring a holistic understanding of fine-tuning techniques and their real-world implications.

- Techniques for Finetuning LLMs: The lesson highlights the challenges, particularly the resource intensity of traditional approaches. We will introduce instruction tuning methods like LoRA.
- Deep Dive into LoRA and SFT: This lesson offers an in-depth exploration of LoRA and SFT techniques. We will uncover the mechanics and underlying principles of these methods.
- Finetuning using LoRA : This lesson guides a practical application of LoRA and SFT to finetune an LLM to follow instructions, using data from the “HuggingFace Dataset” .

1.8.2 Techniques for Finetuning LLMs

Introduction

In this lesson, we will examine the main techniques for fine-tuning Large Language Models for superior performance on specific tasks. We explore why and how to fine-tune LLMs, the strategic importance of instruction fine-tuning, and several fine-tuning methods, such as Low-Rank Adaptation (LoRA), Supervised Finetuning (SFT). We also touch upon the benefits of the Parameter-Efficient Fine-tuning (PEFT) approach using Hugging Face's PEFT library, promising both efficiency and performance gains in fine-tuning.

Why We Finetune LLMs

While pretrained Large Language Models (LLMs) provide a broad understanding of language, it doesn't equip them with the specialized knowledge needed for complex tasks. For instance, a pre-trained LLM may excel at generating text but encounter difficulties when tasked with sentiment analysis or even providing information from your own Knowledge base. This is where fine-tuning comes into play.

Fine-tuning is the process of adapting a pretrained model to a specific task by further training it using task-specific data. For example, if we aim to make an LLM proficient in answering questions about Webex Calling or Webex CC, we would fine-tune it using a dataset comprising Webex question-answer pairs. This process enables the model to recalibrate its internal parameters and representations to align with the intended task, enhancing its capacity to address domain-specific challenges effectively.

However, fine-tuning LLMs conventionally can be resource-intensive and costly. It involves adjusting the parameters in the pretrained LLM models, which can number in the billions, necessitating significant computational power and time. Consequently, it's crucial to explore more efficient and cost-effective methods for fine-tuning, such as Low-Rank Adaptation (LoRA).

A Reminder On Instruction and Conversational Finetuning

In Conversational fine tuning the model engages in a dialogue with the user, maintaining context over multiple turns. The interaction mimics a natural conversation, with the model responding in a way that feels like a human interlocutor.

Instruction fine-tuning is a specific type of fine-tuning that grants precise control over a model's behavior. The objective is to train a Language Model (LLM) to interpret prompts as instructions rather than simply treating them as text to continue generating.

Introduction to Efficient Finetuning with Parameter-Efficient Fine-tuning (PEFT)

Parameter-Efficient Fine-tuning (PEFT) approaches address the need for computational and storage efficiency in fine-tuning LLMs. Hugging Face developed the PEFT library specifically for this purpose. PEFT leverages architectures that only fine-tune a small number of additional model parameters while freezing most parameters of the pretrained LLMs, significantly reducing computational and storage costs.

PEFT methods offer benefits beyond just efficiency. These methods have been proven to outperform standard fine-tuning methods, particularly in low-data situations, and provide improved generalization for out-of-domain scenarios. Furthermore, they contribute to the portability of models by generating tiny model checkpoints that require substantially less storage space compared to extensive full fine-tuning checkpoints.

The PEFT library supports popular methods such as Low-Rank Adaptation (LoRA) and Prompt Tuning.

A Reminder of the Techniques For Finetuning LLMs

There are several techniques to make the finetuning process more efficient and effective:

- Full Finetuning: This method involves adjusting all the parameters in the pretrained LLM models to adapt to a specific task. While effective, it is resource-intensive and requires extensive computational power, therefore it's rarely used. Not in our Scope
- Low-Rank Adaptation (LoRA): LoRA is a technique that aims to adapt LLMs to specific tasks and datasets while simultaneously reducing computational resources and costs. By applying low-rank approximations to the downstream layers of LLMs, LoRA significantly reduces the number of parameters to be trained, thereby lowering the GPU memory requirements and training costs. We'll also see QLoRA, a variant of LoRA that is more optimized and leverages quantization.

With a focus on the number of parameters involved in finetuning, there are multiple methods, such as:

- Supervised Finetuning (SFT): SFT involves doing standard supervised finetuning with a pretrained LLM on a small amount of demonstration data. This method is less resource-intensive than full finetuning but still requires significant computational power. Scope of this lab
- Reinforcement Learning from Human Feedback (RLHF): RLHF is a training methodology where models are trained to follow human feedback over multiple iterations. This method can be more effective than SFT, as it allows for continuous improvement based on human feedback. We'll also see some alternatives to RLHF, such as Direct Preference Optimization (DPO), and Reinforcement Learning from AI Feedback (RLAIF).Not in our Scope

Conclusion

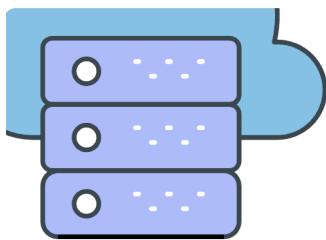
In this lesson, we've learned that while pretraining equips LLMs with a broad understanding of language, fine-tuning is necessary to specialize these models for complex tasks. We've looked into various fine-tuning techniques, including Full Finetuning, Low-Rank Adaptation (LoRA), Supervised Finetuning (SFT), and Reinforcement Learning from Human Feedback (RLHF).

1.9 Deep Dive into Quantization, LoRA and SFT

1.9.1 Introduction

In this lesson, we will dive deeper into Quantization and the mechanics of LoRA, a powerful method for optimizing the fine-tuning process of Large Language Models, its practical uses in various fine-tuning tasks, and the open-source resources that simplify its implementation. We will also introduce QLoRA, a highly efficient version of LoRA. By the end of this lesson, you will have an in-depth understanding of quantization and how LoRA and QLoRA can enhance the efficiency and accessibility of fine-tuning LLMs.

1.9.2 Understanding Quantization

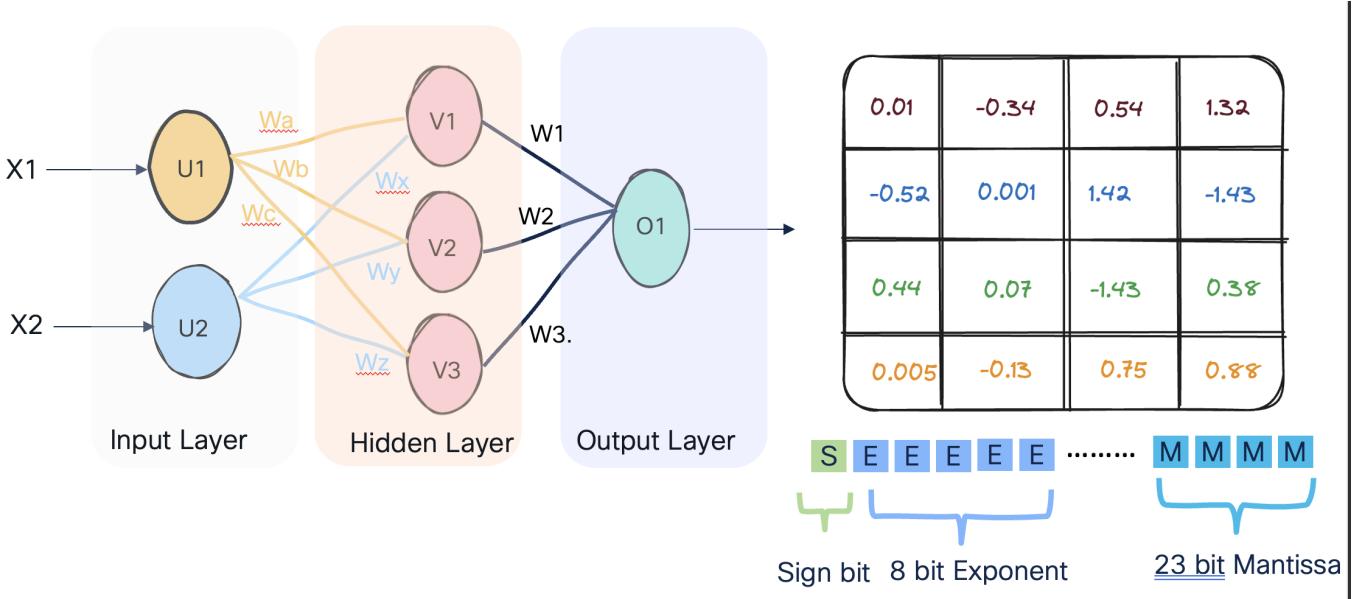


“Conversion from higher bit memory format to lower bit memory format”

As large language models grow more advanced, their increasing number of parameters leads to significant memory usage. This, in turn, increases the costs of hosting and deploying these tools.

In this section, we will quickly learn about quantization. Quantization is a process that converts data that's in your memory from high bit to low bit rate.

Quantization in Machine Learning



Neural Networks have millions or billions of parameters (floating point values) representing their weights and biases, a characteristic known as precision. The precision of these data types directly impacts the amount of memory required by the model. Using higher precision types, such as Float32 or Float64, offers greater accuracy but demands more memory. By default when you train your neural network usually the parameters are stored in FP32, which means each parameter takes 32 bits. The idea of Quantization is to reduce parameters that takes less space

example 16 bit floating points(FP16) or Int8. If we store all the parameters in INT8 , it means everything be saved between 0 - 255, which will result in saving space 4 times comapred to FP32

Quantization can significantly reduce this memory requirement by representing parameters with fewer bits. For example, converting from 32-bit to smaller bit representations allows us to perform inference with the models more efficiently. Once models are quantized, they can be deployed on edge devices like phones, smartwatches, etc. However, using lower precision types, such as Float16 or BFloat16 or Int8, while saving memory, may result in a slight loss of accuracy. To overcome this loss in accuracy, techniques like Lora and QLoRA can be utilized.

Benefits of Quantization:

FP16 QUANTIZATION:

- Reduces the model size by half.
- Maintains higher precision than INT8.
- Suitable for GPUs and TPUs that support FP16 arithmetic.

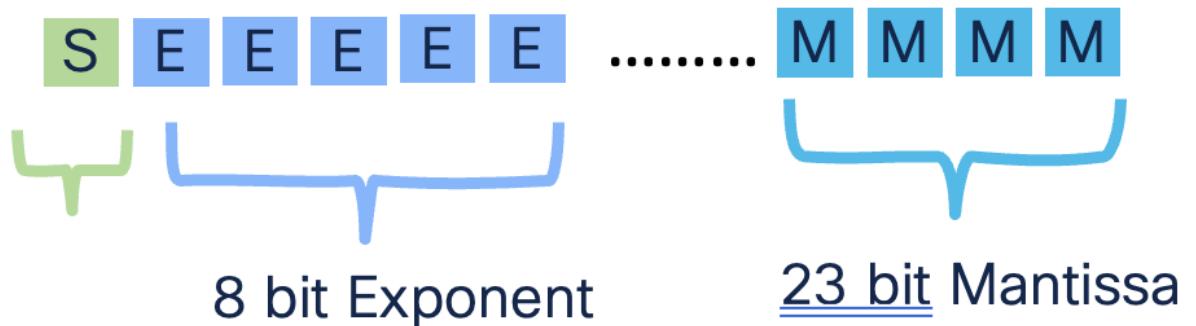
INT8 QUANTIZATION:

- Significantly reduces model size and improves latency.
- Suitable for CPUs and hardware accelerators that support INT8 arithmetic.

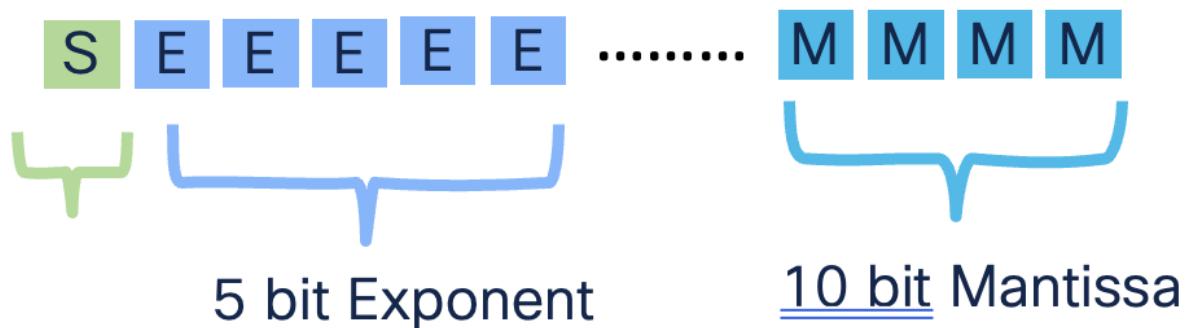
By applying quantization, we can make large models more efficient and capable of running on a variety of devices, ensuring a balance between performance and resource utilization.

In the figure below, you can see the main floating point data types.

FP32

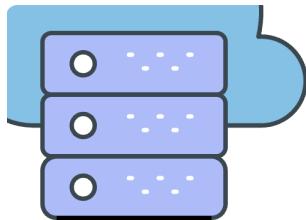


FP16



Why Quantization uses less memory

As mentioned above, the primary advantage of using quantization (4-bit) is the reduction in model size and memory usage. Here's a simple explanation:



“Imagine we have a FP32. Do you know how this number is stored in memory ”

- A float32 number takes up 32 bits of memory.
- A 4-bit quantized number takes up only 4 bits of memory. So, theoretically, you can fit 8 times more 4-bit quantized numbers into the same memory space as float32 numbers. This allows you to load larger models into the GPU memory or use smaller GPUs that might not have been able to handle the model otherwise.

Note: The amount of memory used by an integer in a computer system is directly related to the number of bits used to represent that integer.

Memory Usage for 4-bit Integer A 4-bit integer uses 4 bits of memory

Memory Usage for 32-bit Integer A 32-bit integer uses 32 bits of memory.

Conversion to Bytes To convert these to bytes (since memory is often measured in bytes):

```
1 byte = 8 bits
A 4-bit integer would use ( 4/8 = 0.5 ) bytes.
A 16-bit integer would use ( 16/8 = 2 ) bytes.
```

Lets Understand by taking Llama 2 as an example:

For example, you may come across config like this in Llama 2 model:

Note: The bitsandbytes library will be used for quantization and fine-tuning of models. It provides tools to reduce memory footprint and improve computational efficiency.

```
bnn_config = transformers.BitsAndBytesConfig(
    load_in_4bit=True,
    bnb_4bit_quant_type='nf4',
    bnb_4bit_use_double_quant=True,
    bnb_4bit_compute_dtype=bfloat16
)
load_in_4bit=True //Enables 4-bit quantization.
bnb_4bit_quant_type='nf4' //Specifies the type of 4-bit quantization.
bnb_4bit_use_double_quant=True //Enables double quantization for better accuracy.
bnb_4bit_compute_dtype=bfloat16 //Specifies the data type for computation, which is bfloat16 here.
```

By using 4-bit quantization, you can load the Llama 2 model with significantly less GPU memory, making it more accessible for devices with limited resources.

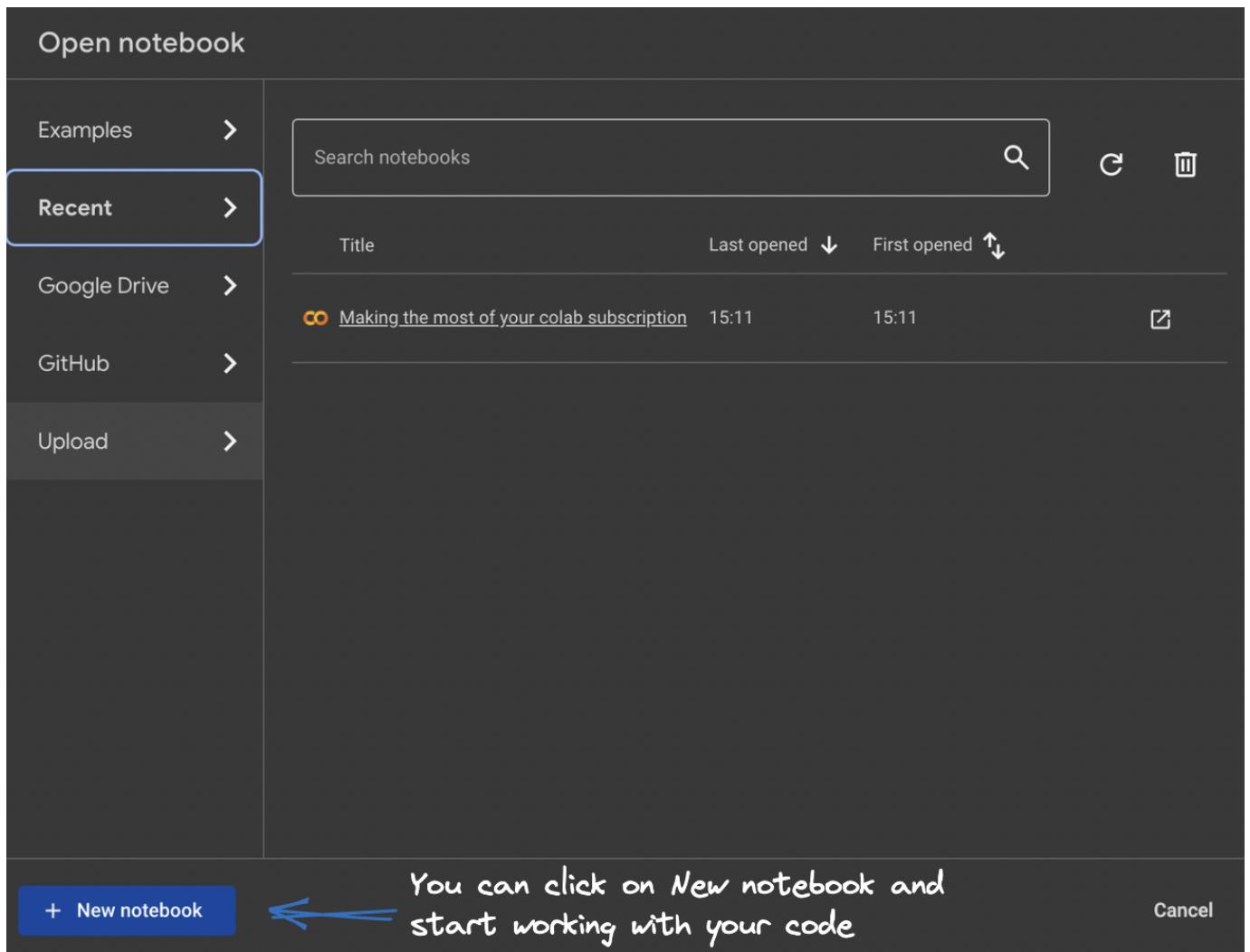
1.9.3 Understanding Quantization - Lab Example

We will be using and running the code in Google Colab. This code demonstrates how much memory can be saved by using 4-bit quantization compared to 32-bit floating-point precision for a hypothetical model with 1000 weights. It calculates the total memory required for both 32-bit and 4-bit representations, determines the memory savings, and converts the result to bytes for easier understanding.

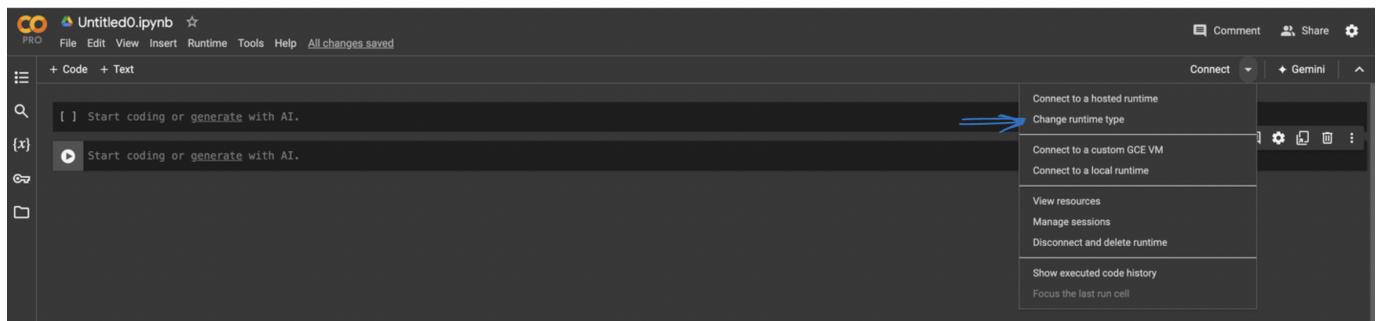
By quantizing the model weights from 32-bit to 4-bit, significant memory savings are achieved, making it more efficient to deploy models on resource-constrained devices like phones and smartwatches.

If you have not yet created a Google Colab account, please refer to the following section

- Browse to Google Colab and create a new Notebook



- Select your runtime environment



NOTE: As we will be running a hypothetical model in Google Colab, you can use the CPU to run the code below. Our goal is to understand how quantization can help us use less memory.

Reminder: Whenever you want to copy the below code into Google Colab and run it on individual line, be sure to click on + Code to add a new code cell. Also make sure to press the Play button to execute the code



press the play button
or use the keyboard
shortcut "Command/Ctrl+Enter"

- Let's break down the code step by step and explain what it does:

Step 1: Define Memory Weight for Float32:

```
1 float32_memory = 32 # in bits
2 num_weights = 1000 # hypothetical number of weights
```

- float32_memory is set to 32 bits, which is the memory required for each weight when using 32-bit floating-point precision (Float32).
- num_weights is set to 1000, representing a hypothetical number of weights in the model.

Step 2: Calculate Total Memory for Float32 Weights:

```
1 float32_total_memory = float32_memory * num_weights # in bits
```

- float32_total_memory calculates the total memory required for all the weights when using Float32. This is done by multiplying the memory per weight (float32_memory) by the number of weights (num_weights).

Step 3: Define Memory Per Weight for 4-bit Quantization:

```
1 bit4_memory = 4 # in bits
```

- bit4_memory is set to 4 bits, which is the memory required for each weight when using 4-bit quantization.

Step 4: Calculate Total Memory for 4-bit Quantized Weights:

```
1 bit4_total_memory = bit4_memory * num_weights # in bits
```

- bit4_total_memory calculates the total memory required for all the weights when using 4-bit quantization. This is done by multiplying the memory per weight (bit4_memory) by the number of weights (num_weights).

Step 5: Calculate Memory Saved:

```
1 memory_saved = float32_total_memory - bit4_total_memory # in bits
2 memory_saved_in_bytes = memory_saved / 8 # convert bits to bytes
```

- memory_saved calculates the total amount of memory saved by using 4-bit quantization instead of Float32. This is done by subtracting the total memory required for 4-bit weights (bit4_total_memory) from the total memory required for Float32 weights (float32_total_memory).
- memory_saved_in_bytes converts the saved memory from bits to bytes by dividing by 8 (since there are 8 bits in a byte).

Step 5: Print the Result:

```
1 print(f"Memory saved by using 4-bit quantization: {memory_saved_in_bytes} bytes")
```

OUTPUT

```
Memory saved by using 4-bit quantization: 3500.0 bytes
```

- Will changing the model from 32-bit to 4-bit reduce the model's accuracy? Yes

NOTE: Let's sets up the context of the code, indicating that reducing the bit-width from 32-bit to 4-bit quantization can introduce a loss of accuracy. Please us ethe same notebook as above to continue with your code.

Step 1: Import Python Library NumPY

```
1 import numpy as np
```

* Import the NumPy library, which is used for numerical operations on arrays.

Step 2: Simulate Original float32 Weights:

```
1 original_weights = np.random.rand(1000).astype(np.float32)
```

* Generate an array of 1000 random floating-point numbers between 0 and 1, simulating the original model weights with 32-bit precision (float32).

Step 3: Simulate 4-bit Quantized Weights:

```
1 quantized_weights = np.round(original_weights * 15).astype(np.uint8)
```

* Normalize the original weights to a range of 0 to 15 because 4-bit integers can represent 16 values (0 to 15). * Use np.round() to round the normalized weights to the nearest integer. * Convert the rounded values to 8-bit unsigned integers (np.uint8), which effectively simulates the 4-bit quantized weights.

Step 4: De-normalize to Get the Approximated Original Weights:

```
1 approximated_weights = quantized_weights / 15.0
```

* De-normalize the quantized weights back to the original range by dividing by 15.0. This gives an approximation of the original weights after quantization.

Step 5: Calculate the Error

```
1 error = np.abs(original_weights - approximated_weights).mean()
```

* Calculate the absolute difference between the original weights and the approximated weights. * Compute the mean of these differences to get the average quantization error.

Step 6: Print the Result:

```
1 print(f"Average Quantization Error: {error}")
```

* Print the average quantization error to show the potential loss of accuracy introduced by reducing the bit-width from 32-bit to 4-bit.

OUTPUT

```
Average Quantization Error: 0.016423573800129817
```

Optional Step Lets see the original generated randon floating point weights

```
original_weights

0.9047668 , 0.81753767, 0.15176044, 0.82469964, 0.26190883,
0.25543845, 0.91670144, 0.44082588, 0.8387138 , 0.5301147 ,
0.597587 , 0.28645265, 0.31866026, 0.8412814 , 0.87486124,
0.14635882, 0.00842665, 0.40396062, 0.7154632 , 0.92024857,
```

```

0.58492464, 0.369594 , 0.3199917 , 0.01947883, 0.5505348 ,
0.09161595, 0.75082487, 0.9818501 , 0.800069 , 0.55406606,
0.87903994, 0.6254798 , 0.8290317 , 0.73905164, 0.42614278,
0.6899402 , 0.03985518, 0.3132978 , 0.97768533, 0.42348453,
0.60850483, 0.80669063, 0.18966372, 0.79855096, 0.73046243,
0.3299927 , 0.79309106, 0.01327937, 0.01851322, 0.9269705 ,
0.8274083 , 0.45073316, 0.51967096, 0.7210426 , 0.41424063,
0.24505323, 0.90779364, 0.64724004, 0.65470797, 0.14317876,
0.91511154, 0.57397956, 0.83594376, 0.09163648, 0.3037902 ,
0.01362608, 0.92493033, 0.6721083 , 0.5777128 , 0.98029524,
0.29148692, 0.5840548 , 0.805403 , 0.17140059, 0.9880919 ,
0.01023464, 0.8627494 , 0.89710003, 0.76805186, 0.9278038 ,
0.9041322 , 0.19642776, 0.78401035, 0.16597563, 0.16709319,
0.41655192, 0.6231926 , 0.95298046, 0.6890473 , 0.79470664,
0.7721617 , 0.2800831 , 0.32485998, 0.83056533, 0.84955883,
0.9345336 , 0.20025656, 0.31240928, 0.0943019 , 0.92295516,
0.07533643, 0.2782531 , 0.34772104, 0.20622289, 0.6932999 ,
0.01451688, 0.05406734, 0.69295406, 0.56339973, 0.74457246,
0.8313878 , 0.51609147, 0.4473876 , 0.29768255, 0.9012652 ,
0.7957006 , 0.5588959 , 0.54948616, 0.41233262, 0.04758253,
0.09688636, 0.19657251, 0.60875326, 0.6305743 , 0.62236613,
0.53727114, 0.60597694, 0.27432016, 0.5396689 , 0.35570994,
0.7779527 , 0.62765694, 0.5046841 , 0.95913595, 0.37506285,
0.06490876, 0.5690066 , 0.5926366 , 0.8343261 , 0.00439525,
0.75573623, 0.89352626, 0.45979103, 0.16634902, 0.2184745,
0.39872894, 0.68869066, 0.08809475, 0.8567979 , 0.2187864,
0.21870795, 0.52296937, 0.16986693, 0.6928503 , 0.41990125,
0.5135413 , 0.45609972, 0.38769007, 0.00987561, 0.15869647,
0.80649745, 0.8347802 , 0.7420551 , 0.07340008, 0.6842359 ,
0.33549812, 0.6722021 , 0.3444218 , 0.7548876 , 0.67199737,
0.04117178, 0.24504112, 0.39629644, 0.6458071 , 0.4220033 ,
0.44719183, 0.00884627, 0.8429784 , 0.5802084 , 0.07012173,
0.2552524 , 0.23461716, 0.08043662, 0.17875864, 0.35063305,
0.99479717, 0.3229556 , 0.7217851 , 0.09103325, 0.00775268,
0.3796975 , 0.24862018, 0.32581842, 0.530216 , 0.71165705,
0.5170614 , 0.06309021, 0.6869316 , 0.9265822 , 0.04267561,
0.44075203, 0.28398433, 0.46985227, 0.5829586 , 0.23592392,
0.95805424, 0.52453846, 0.8329666 , 0.14974083, 0.48412293,
0.04571012, 0.8544567 , 0.89506125, 0.10473824, 0.29405326,
0.29569003, 0.59090966, 0.66576856, 0.35182896, 0.12008226,
0.49510583, 0.8308251 , 0.14861059, 0.5513083 , 0.45685318,
0.8333833 , 0.9543581 , 0.42451006, 0.6116371 , 0.76232755,
0.81956553, 0.89240205, 0.2516925 , 0.92773354, 0.9864879 ,
0.04117434, 0.78647643, 0.21779937, 0.3965154 , 0.6441789 ,
0.6636646 , 0.547766 , 0.8827409 , 0.24284157, 0.1969708 ,
0.5668731 , 0.78194785, 0.6832345 , 0.10209882, 0.97626406,
0.5010292 , 0.12451114, 0.32503366, 0.17754969, 0.53745407],
dtype=float32)

```

Optional Step Lets see the quantized_weights (4- bit)

```

quantized_weights

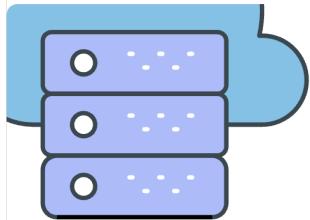
6,  1,  2,  1,  8, 14,  3,  9, 13,  1, 10,  5, 14, 12, 13,  9,  2,
4,  1,  6,  7,  5, 15,  3,  0,  9,  9, 12,  2,  6, 11,  4,  3, 13,
8,  7, 14,  4,  9, 10,  0,  8,  9,  6,  9, 10,  8,  4,  1, 13,  1,
7,  3, 13,  5,  3, 11,  6, 12,  8,  8,  2,  8, 13, 10,  6,  6,  0,

```

```

0, 3, 7, 10, 11, 4, 11, 13, 3, 8, 8, 1, 5, 5, 7, 2, 13,
5, 12, 4, 11, 6, 9, 6, 8, 13, 4, 13, 6, 6, 15, 13, 2, 0,
4, 15, 7, 7, 5, 5, 2, 4, 8, 8, 15, 6, 11, 9, 12, 1, 13,
3, 14, 8, 1, 5, 15, 1, 9, 7, 1, 1, 13, 11, 12, 4, 5, 9,
14, 11, 14, 0, 12, 12, 2, 4, 9, 10, 8, 7, 3, 7, 9, 1, 13,
7, 9, 7, 12, 14, 11, 3, 11, 9, 11, 1, 13, 9, 4, 5, 2, 15,
8, 13, 1, 9, 3, 15, 3, 3, 12, 8, 14, 7, 11, 13, 3, 0, 10,
15, 11, 14, 4, 11, 7, 6, 13, 1, 4, 10, 1, 11, 9, 2, 10, 11,
8, 1, 5, 7, 8, 11, 13, 3, 3, 14, 4, 10, 10, 1, 6, 12, 13,
12, 7, 6, 8, 9, 10, 7, 12, 11, 1, 7, 14, 13, 8, 14, 1, 13,
7, 9, 12, 14, 14, 1, 14, 14, 11, 13, 2, 9, 11, 1, 13, 1, 13,
2, 8, 12, 1, 6, 7, 1, 12, 7, 4, 6, 8, 9, 8, 4, 1, 7,
13, 7, 12, 0, 6, 7, 11, 15, 7, 6, 13, 12, 12, 10, 8, 9, 12,
0, 10, 1, 13, 9, 3, 2, 14, 9, 6, 9, 6, 4, 15, 5, 5, 5,
12, 5, 15, 4, 7, 0, 3, 3, 15, 15, 3, 10, 2, 15, 7, 8, 2,
2, 10, 11, 14, 4, 0, 8, 1, 8, 5, 1, 12, 5, 6, 14, 11, 10,
2, 6, 11, 12, 8, 12, 0, 14, 12, 4, 9, 11, 7, 14, 2, 4, 4,
10, 4, 13, 1, 14, 0, 15, 8, 6, 5, 12, 10, 6, 1, 4, 7, 7,
7, 10, 10, 5, 8, 12, 8, 6, 12, 2, 1, 14, 15, 13, 9, 3, 3,
8, 7, 3, 10, 7, 1, 10, 7, 6, 4, 8, 2, 11, 2, 15, 2, 11,
12, 10, 5, 7, 13, 2, 6, 15, 13, 6, 1, 10, 12, 10, 11, 4, 1,
8, 3, 4, 12, 15, 12, 9, 8, 1, 12, 2, 2, 14, 2, 9, 7, 4,
4, 3, 2, 11, 0, 5, 9, 12, 15, 14, 13, 1, 15, 3, 12, 10, 3,
13, 5, 8, 9, 15, 2, 14, 8, 8, 6, 7, 1, 6, 3, 15, 3, 0,
13, 9, 6, 12, 5, 5, 9, 9, 2, 13, 15, 3, 9, 11, 1, 8,
13, 9, 12, 14, 5, 9, 2, 8, 14, 10, 4, 12, 8, 9, 10, 13, 14,
3, 1, 14, 9, 9, 12, 2, 1, 7, 4, 2, 0, 4, 10, 2, 6, 3,
10, 10, 9, 11, 9, 8, 15, 12, 3, 2, 5, 9, 6, 11, 1, 9, 5,
14, 7, 14, 4, 13, 14, 5, 8, 14, 10, 12, 3, 1, 12, 9, 14, 1,
2, 11, 11, 2, 2, 13, 12, 3, 4, 12, 7, 1, 9, 3, 4, 4, 14,
14, 8, 0, 12, 8, 6, 12, 13, 6, 11, 6, 10, 9, 6, 2, 14, 10,
4, 10, 2, 15, 9, 14, 0, 4, 0, 10, 1, 15, 5, 7, 10, 6, 12,
10, 11, 6, 7, 2, 14, 8, 12, 7, 14, 9, 13, 0, 5, 7, 2, 11,
9, 11, 13, 10, 14, 8, 8, 14, 4, 10, 1, 11, 15, 10, 0, 6, 0,
13, 14, 9, 4, 4, 12, 5, 9, 4, 6, 2, 10, 3, 4, 1, 14, 11,
6, 7, 1, 4, 11, 1, 7, 12, 8, 14, 2, 8, 8, 3, 11, 15, 10,
2, 14, 12, 2, 12, 4, 4, 14, 7, 13, 8, 9, 4, 5, 13, 13, 2,
0, 6, 11, 14, 9, 6, 5, 0, 8, 1, 11, 15, 12, 8, 13, 9, 12,
11, 6, 10, 1, 5, 15, 6, 9, 12, 3, 12, 11, 5, 12, 0, 0, 14,
12, 7, 8, 11, 6, 4, 14, 10, 10, 2, 14, 9, 13, 1, 5, 0, 14,
10, 9, 15, 4, 9, 12, 3, 15, 0, 13, 13, 12, 14, 14, 3, 12, 2,
3, 6, 9, 14, 10, 12, 12, 4, 5, 12, 13, 14, 3, 5, 1, 14, 1,
4, 5, 3, 10, 0, 1, 10, 8, 11, 12, 8, 7, 4, 14, 12, 8, 8,
6, 1, 1, 3, 9, 9, 9, 8, 9, 4, 8, 5, 12, 9, 8, 14, 6,
1, 9, 9, 13, 0, 11, 13, 7, 2, 3, 3, 14, 12, 11, 4, 13, 15,
4, 7, 13, 13, 0, 8, 13, 13, 15, 8, 15, 3, 11, 4, 8, 9, 2,
2, 11, 3, 7, 4, 7, 11, 6, 10, 1, 13, 3, 8, 3, 10, 6, 8,
7, 6, 0, 2, 12, 13, 11, 1, 10, 5, 10, 5, 11, 10, 1, 4, 6,
10, 6, 7, 0, 13, 9, 1, 4, 4, 1, 3, 5, 15, 5, 11, 1, 0,
6, 4, 5, 8, 11, 8, 1, 10, 14, 1, 7, 4, 7, 9, 4, 14, 8,
12, 2, 7, 1, 13, 13, 2, 4, 4, 9, 10, 5, 2, 7, 12, 2, 8,
7, 13, 14, 6, 9, 11, 12, 13, 4, 14, 15, 1, 12, 3, 6, 10, 10,
8, 13, 4, 3, 9, 12, 10, 2, 15, 8, 2, 5, 3,
dtype=uint8)

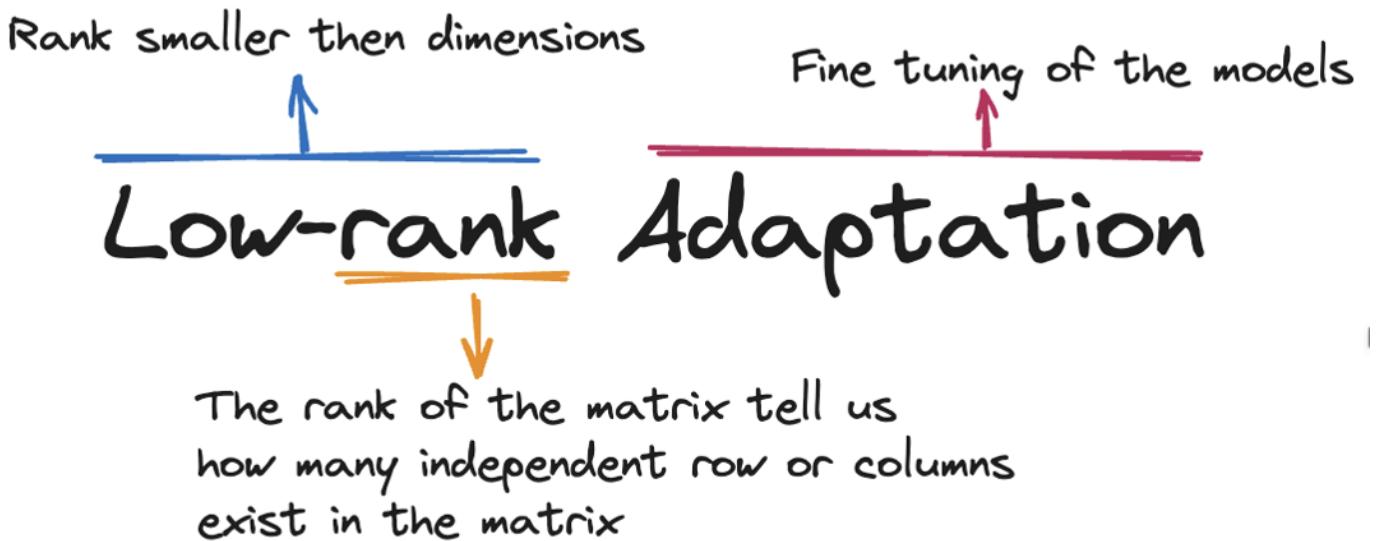
```



“Why fine-tune when we can Quantize”

- After quantizing LLMs (Large Language Models) from 32-bit to 8-bit precision, we achieve significant reductions in memory usage and computational demands. However, this process can also result in some loss of information and a decrease in accuracy. This brings up an important question: why should we fine-tune a quantized model?
- Fine-tuning becomes crucial in this context because it helps to restore or even enhance the model's performance after quantization. By fine-tuning, we can adapt the model to specific tasks or datasets, which can mitigate the accuracy loss that occurs during quantization.

1.9.4 The Functioning of LoRA in Fine-tuning LLMs

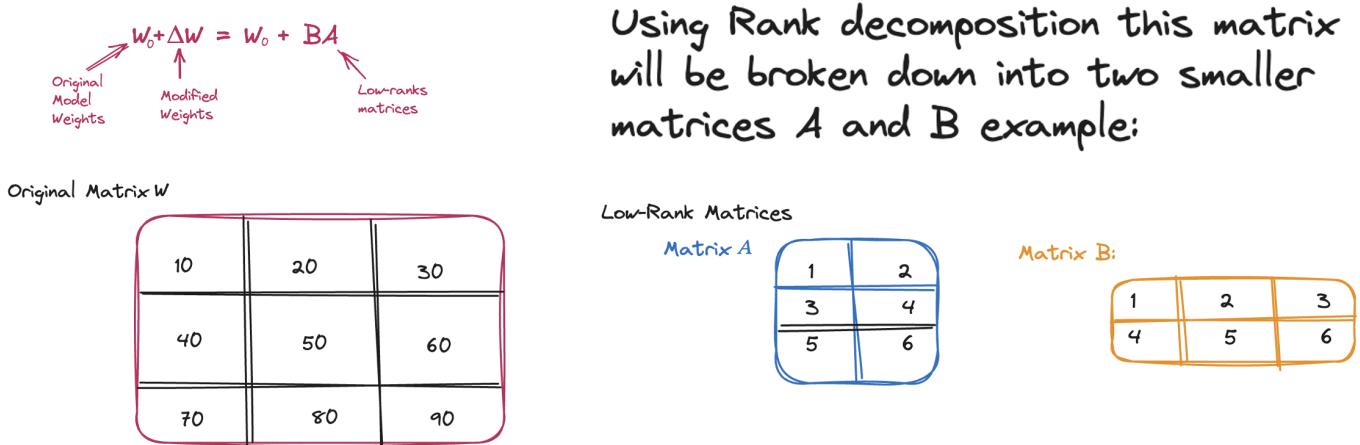


LoRA, or Low-Rank Adaptation, is a method developed by Microsoft researchers to optimize the fine-tuning of Large Language Models. This technique tackles the issues related to the fine-tuning process, such as extensive memory demands and computational inefficiency. LoRA introduces a compact set of parameters, referred to as low-rank matrices, to store the necessary changes in the model instead of altering all parameters.

Here are the key features of how LoRA operates:

- Maintaining Pretrained Weights: LoRA adopts a unique strategy by preserving the pretrained weights of the model. This approach reduces the risk of catastrophic forgetting, ensuring the model maintains the valuable knowledge it gained during pretraining.
- Efficient Rank-Decomposition: LoRA incorporates rank-decomposition weight matrices, known as update matrices, to the existing weights. Rank decomposition is a way to simplify a large, complex matrix by breaking it down into the product of smaller matrices. These update matrices have significantly fewer parameters than the original model, making them highly memory-efficient. By training only these newly added weights, LoRA achieves a faster training process with reduced memory demands. These LoRA matrices are typically integrated into the attention layers of the original model.

By using the low-rank decomposition approach, the memory demands for training large language models are significantly reduced. This allows running fine-tuning tasks on consumer-grade GPUs, making the benefits of LoRA available to a broader range of researchers and developers.



Open-source Resources for LoRA

The following libraries offer a mix of tools that enhance the efficiency of fine-tuning large language models. They provide optimizations, compatibility with different data types, resource efficiency, and user-friendly interfaces that accommodate various tasks and hardware configurations.

- PEFT Library: Parameter-efficient fine-tuning (PEFT) methods facilitate efficient adaptation of pre-trained language models to various downstream applications without fine-tuning all the model's parameters. By fine-tuning only a portion of the model's parameters, PEFT methods like LoRA, Prefix Tuning, and P-Tuning, including QLoRA, significantly reduce computational and storage costs.
- Lit-GPT: Lit-GPT from LightningAI is an open-source resource designed to simplify the fine-tuning process, making it easier to apply LoRA's techniques without manually altering the core model architecture. Models available for this purpose include Vicuna, Pythia, and Falcon. Specific configurations can be applied to different weight matrices, and precision settings can be adjusted to manage memory consumption.

In our lab, we'll focus on the PEFT library.

QLoRA: An Efficient Variant of LoRA

QLoRA, or Quantized Low-Rank Adaptation, is a popular variant of LoRA that makes fine-tuning large language models even more efficient. QLoRA introduces several innovations to save memory without sacrificing performance.

The technique involves backpropagating gradients through a frozen, 4-bit quantized pretrained language model into Low-Rank Adapters. This approach significantly reduces memory usage, enabling the fine-tuning of even larger models on consumer-grade GPUs. For instance, QLoRA can fine-tune a 65 billion parameter model on a single 48GB GPU while preserving full 16-bit fine-tuning task performance.

QLoRA uses a new data type known as 4-bit NormalFloat (NF4), which is optimal for normally distributed weights. It also employs double quantization to reduce the average memory footprint by quantizing the quantization constants and paged optimizers to manage memory spikes.

The practical implementation of QLoRA for fine-tuning LLMs is very accessible, thanks to open-source libraries and tools. For instance, the BitsAndBytes library offers functionalities for 4-bit quantization. In the coming sections we will see a code example showing how to use QLoRA with PEFT.

1.9.5 Conclusion

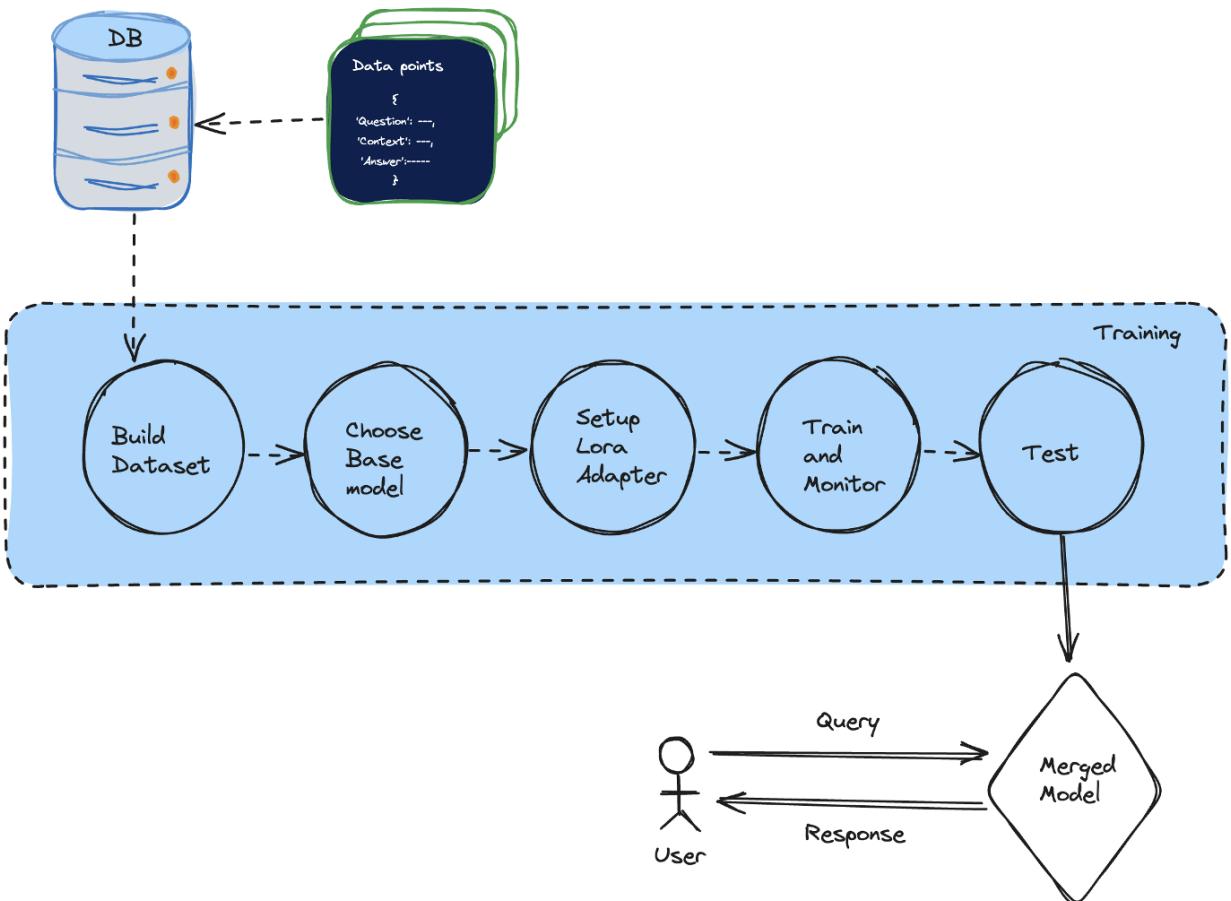
In this lesson, we explored LoRA and QLoRA, two powerful techniques for fine-tuning Large Language Models (LLMs). We examined how LoRA operates by preserving pretrained weights and introducing low-rank matrices, thereby enhancing the memory and computational efficiency of the fine-tuning process. We also highlighted open-source libraries like PEFT and Lit-GPT that facilitate the implementation of LoRA.

Additionally, we discussed QLoRA, an efficient variant of LoRA that employs 4-bit NormalFloat and double quantization to further reduce memory usage. This technique allows for significant memory savings but can introduce quantization errors, impacting model accuracy.

By quantizing weights to lower precision, we achieve reduced memory usage and potentially increased processing speed, at the cost of a possible minor loss in accuracy. The provided code example illustrates this trade-off by converting a set of 32-bit floating-point weights into 4-bit integer weights, demonstrating the balance between memory efficiency and model accuracy.

Overall, you have successfully learned how to use quantization to optimize LLMs, understanding the benefits and potential drawbacks of this approach.

1.10 Task 8a - Configuring and Fine Tuning - Using llama2

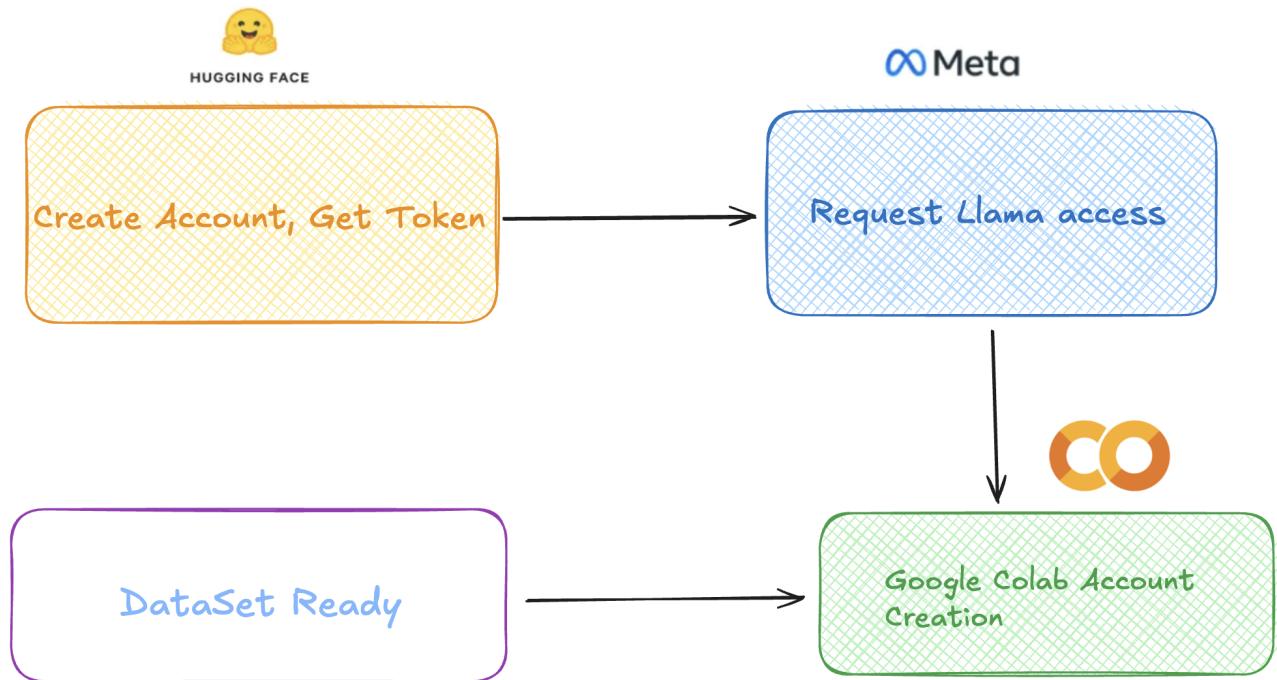


1.10.1 Introduction

This diagram represents the workflow of training and using a LLM. Here's a step-by-step explanation:

- Build Dataset: Once we have determined that fine-tuning is the right solution, we need to create a dataset to fine-tune our model. Datasets are collections of structured or unstructured data used to train machine learning models.
- Choose Base Model: A base model is selected. This could be a pre-trained model that will be fine-tuned with the new dataset e.g Llama2 or Llama3 (in our lab)
- Setup Lora Adapter: The LoRA (Low-Rank Adaptation) adapter is set up a technique used to fine-tune the model efficiently.
- Train and Monitor: The model is trained using the dataset. During training, the process is monitored to ensure it's proceeding correctly and to make adjustments if necessary.
- Merged Model: Once the model has been tested and validated, it is considered a merged model. This model is ready to handle user queries.
- User Interaction: Users can now interact with the model by sending queries. The model processes these queries and returns appropriate responses.

1.10.2 Pre-requisites



1.10.3 UseCase 1 - Fine Tuning Using Llama2

Datasets Creation

Datasets

Once you have determined that fine-tuning is the right solution, we need to create a dataset to fine-tune our model.

Datasets are collections of structured or unstructured data used to train machine learning models.

The quality of our datasets is **essential**. It is important that our datasets are trained on our tasks

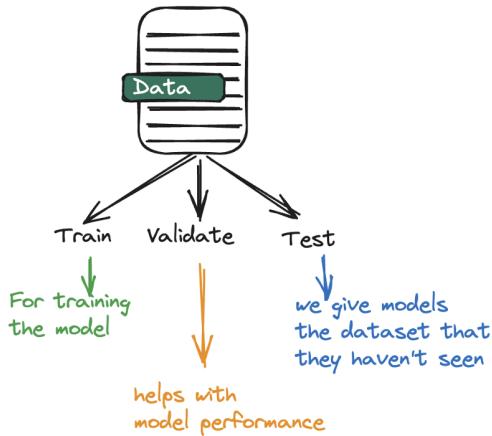


There are several ways to create datasets:

- Using existing open source datasets e.g The Pile, Common Crawl , Wiki or even HuggingFace
- Using LLM to create synthetic datasets faster and can be less expensive
- Using your own custom datasets e.t.c <-- Focus for our lab

Datasets

Before training your model you normally split your data into 3 categories



- Training set is utilized during training process to optimize model's parameters.
- Validation used to assess model's performance
- Testing data only used for predictions (models should never know anything about Test data)

Custom datasets

As we will be using and fine tuning Llam2 we need to convert them(datasets) into a uniform format compatible with training regimens More Info

EXAMPLE FORMAT FOR LLAMA2

Sample Format - Dont Copy - reference ONLY

```

<s>[INST] <<SYS>>
{{ system_prompt }}
<</SYS>>
{{ user_message }} [/INST] Model answer </s>
  
```

Datasets Creation

After defining the problem and confirming that LLMs are a suitable solution, the next step is to prepare a dataset for fine-tuning. If you already have a clean, high-quality dataset, that's great. However, let's assume you don't have one readily available.

In my scenario, I was able to generate synthetic dataset from the Cisco Preferorange Architecture for Webex Calling.

CREATING A DATASET FROM A PDF - SUMMARY

To delve into the intricacies of dataset creation, let's explore a scenario where we aim to train an AI to replicate our preferorange architecture. In this lab, we will transform a PDF document into a dataset. Specifically, we will use a PDF about Webex Calling as our source material.

Our goal is to convert the PDF into a dynamic and engaging dialogue, similar to an interview format. This will allow us to create a more interactive and informative dataset.

Steps Involved

Converting the PDF to Text

- Begin by extracting the text content from the PDF document.

Paragraph Analysis and Segmentation:

- Once the PDF is in text form, analyze and segment it into individual paragraphs. Combine short paragraphs and split longer ones to ensure each segment is coherent and self-contained, while still contributing to the overall narrative.

Generating Interview Questions:

- For each paragraph, create an artificial interview scenario where a language model (LLM) acts as the interviewer.
- Develop questions that naturally elicit responses based on the content of the paragraph.
- The objective is to simulate an insightful dialogue, making it seem as though the PDF is responding to questions about its content and experiences.

By following these steps, we will convert the static information in the PDF into a dynamic and interactive dataset, facilitating a deeper understanding and more engaging learning experience. You can find the complete code that worked for me here.

Here's an example of the resulting file. Please note that for this lab demonstration, only a few data points were converted from the PDF.

Note: We will use the following file to create and eventually train downstream tasks for our LLM, specifically the Llama2 model.

[Download example.txt](#)

SAMPLE FILE CONTENT

example.txt

```
human_text: What is the solution provided by Webex Calling? assistant_text: Webex Calling provides enterprise-grade PBX features, functionality, and performance.
human_text: What options are available for PSTN access in Webex Calling? assistant_text: The options are Cisco Calling Plans, Cloud Connected PSTN, and Premises-based PSTN.
human_text: What type of devices does Webex Calling support? assistant_text: Webex Calling supports Cisco MPP Series IP Phones, Webex Devices, and a variety of third-party devices.
human_text: How are Webex Calling data centers distributed? assistant_text: Webex Calling data centers are globally distributed and geo-redundant, located in six regions: US, Canada, Europe, UK, Australia, and Japan.
human_text: What is the function of the Local Gateway in Webex Calling? assistant_text: The Local Gateway connects to the PSTN and registers with Webex Calling to handle all calls between the PSTN and Webex Calling.
human_text: What are the key benefits of Webex Calling for large enterprises? assistant_text: Webex Calling offers scalability, advanced call management, and integration with collaboration tools for large enterprises.
human_text: How does Webex Calling handle call encryption? assistant_text: SRTP and TLS are used in Webex Calling to encrypt media and signaling for secure communications.
human_text: What features are included in the Webex Calling analytics suite? assistant_text: Call quality, usage patterns, and performance metrics are part of the Webex Calling analytics suite.
human_text: How does Webex Calling integrate with third-party devices? assistant_text: Integration with third-party SIP phones provides flexibility in device choices for Webex Calling users.
human_text: What disaster recovery mechanisms are available in Webex Calling? assistant_text: Geo-redundant data centers and automatic failover mechanisms support disaster recovery in Webex Calling.
human_text: How does Webex Calling manage international call routing? assistant_text: A global network of data centers manages international call routing in Webex Calling.
human_text: What role does the Webex Control Hub play in Webex Calling? assistant_text: Provisioning, configuring, and managing Webex Calling services are handled through the Webex Control Hub.
human_text: What are the main security protocols used in Webex Calling? assistant_text: SRTP and TLS are the main security protocols used in Webex Calling.
human_text: How does Webex Calling support mobile users? assistant_text: The Webex App enables mobile users to make and receive calls on smartphones and tablets in Webex Calling.
human_text: What are the benefits of Webex Calling for remote teams? assistant_text: Remote teams benefit from enterprise-grade calling features, mobility, and collaboration tool integration in Webex Calling.
human_text: How does Webex Calling ensure compliance with regulatory requirements? assistant_text: Compliance with regulatory requirements in Webex Calling is ensured through call recording, monitoring, and secure data handling.
human_text: What redundancy features are built into Webex Calling? assistant_text: Geo-redundant data centers and automatic failover ensure continuous service in Webex Calling.
human_text: What user management capabilities are available in Webex Calling? assistant_text: User settings and permissions are managed through the Webex Control Hub in Webex Calling.
human_text: How does Webex Calling handle call routing? assistant_text: Dial plans and customizable routing rules manage call routing in Webex Calling.
human_text: What kind of customer support does Cisco provide for Webex Calling? assistant_text: Technical assistance, training, and deployment resources are part of Cisco's support for Webex Calling.
human_text: How does Webex Calling support hybrid deployments? assistant_text: Integration with cloud and on-premises PBX systems supports hybrid deployments in Webex Calling.
human_text: What are the call management features in Webex Calling? assistant_text: Call management features in Webex Calling include call forwarding, call transfer, call hold, and voicemail.
human_text: How does Webex Calling support video calls? assistant_text: Video calls are supported between video-capable devices in Webex Calling.
human_text: What compliance features are available in Webex Calling? assistant_text: Compliance features in Webex Calling include call recording, logging, and monitoring.
human_text: What is the function of SRTP in Webex Calling? assistant_text: SRTP encrypts media streams in Webex Calling to ensure secure communication.
human_text: How does Webex Calling support call analytics? assistant_text: Webex Calling provides detailed call analytics on call quality, usage patterns, and performance.
human_text: What kind of redundancy features are available in Webex Calling? assistant_text: Redundancy features in Webex Calling include geo-redundant data centers and automatic failover.
human_text: How does Webex Calling ensure secure voice communications? assistant_text: Encryption protocols like SRTP for media and TLS for signaling ensure secure voice communications in Webex Calling.
human_text: What integrations does Webex Calling offer with contact center solutions? assistant_text: Integration with Webex Contact Center offers advanced features like call routing, IVR, and analytics.
human_text: How does Webex Calling handle call monitoring and recording? assistant_text: Call monitoring and recording features in Webex Calling support compliance, training, and quality assurance.
human_text: What options are available for user authentication in Webex Calling? assistant_text: Webex Calling supports user authentication through secure protocols and integration with identity providers for SSO.
human_text: What is the role of the Webex App in Webex Calling? assistant_text: The Webex App unifies messaging, video conferencing, and calling for a seamless communication experience.
human_text: How does Webex Calling support international business operations? assistant_text: Global data centers support reliable and high-quality voice communication for international operations in Webex Calling.
human_text: What management capabilities does Webex Calling provide? assistant_text: The Webex Control Hub provides centralized provisioning, configuration, and monitoring for Webex Calling services.
human_text: How does Webex Calling handle voice and video integration? assistant_text: Integration with Webex Meetings and Webex Teams allows for seamless
```

voice and video communication in Webex Calling.

human_text: What are the benefits of Webex Calling for large enterprises? assistant_text: Scalability, advanced call management, and integration with collaboration tools are benefits for large enterprises using Webex Calling.

human_text: What disaster recovery options are available with Webex Calling? assistant_text: Geo-redundant data centers and automatic failover provide disaster recovery options in Webex Calling.

human_text: How does Webex Calling manage call quality? assistant_text: Call quality in Webex Calling is managed through network optimization, advanced codecs, and QoS policies.

human_text: What user profile management features are available in Webex Calling? assistant_text: User profile management features in Webex Calling include settings and permissions configuration through the Webex Control Hub.

human_text: How does Webex Calling integrate with CRM systems? assistant_text: Integration with CRM systems enhances customer interactions and business processes in Webex Calling.

human_text: What compliance features does Webex Calling offer? assistant_text: Compliance features in Webex Calling include call recording, monitoring, and secure data handling.

human_text: How does Webex Calling support remote teams? assistant_text: Remote teams benefit from enterprise-grade calling features, mobility, and collaboration tool integration in Webex Calling.

human_text: What are the main security features of Webex Calling? assistant_text: Security features in Webex Calling include encryption, secure voice, and industry-standard compliance.

human_text: How does Webex Calling handle emergency call routing? assistant_text: Predefined routing rules manage emergency call routing in Webex Calling.

human_text: What kind of support does Cisco provide for Webex Calling? assistant_text: Technical assistance, training, and deployment resources are part of Cisco's support for Webex Calling.

human_text: How does Webex Calling support hybrid work environments? assistant_text: Webex Calling supports hybrid work environments by integrating with cloud and on-premises PBX systems.

human_text: What management tools are available in Webex Calling? assistant_text: Management tools in Webex Calling include the Webex Control Hub for provisioning, configuring, and managing services.

human_text: How does Webex Calling handle call quality issues? assistant_text: Tools for monitoring and diagnosing issues ensure high-quality voice communication in Webex Calling.

human_text: What kind of analytics does Webex Calling offer? assistant_text: Analytics on call quality, usage patterns, and performance metrics are available in Webex Calling.

human_text: What devices are compatible with Webex Calling? assistant_text: Compatible devices for Webex Calling include Cisco IP Phones, Webex Room Devices, and third-party SIP phones.

human_text: How does Webex Calling integrate with Webex Meetings? assistant_text: Integration with Webex Meetings provides a seamless experience for scheduling and joining video meetings in Webex Calling.

human_text: What role does Webex Control Hub play in Webex Calling? assistant_text: Webex Control Hub provides a centralized interface for provisioning, configuring, and managing Webex Calling services.

human_text: How does Webex Calling ensure security for voice communications? assistant_text: Encryption protocols like SRTP for media and TLS for signaling ensure security in Webex Calling.

human_text: What features does Webex Calling offer for team collaboration? assistant_text: Group call management, conferencing, and Webex Teams integration are features of Webex Calling for team collaboration.

human_text: How is call recording handled in Webex Calling? assistant_text: Built-in call recording features in Webex Calling support compliance, training, and quality assurance.

human_text: What scalability options are available in Webex Calling? assistant_text: Scalability in Webex Calling supports small businesses to large enterprises, accommodating thousands of users.

human_text: How does Webex Calling support international operations? assistant_text: Global data centers ensure reliable and high-quality voice communication for international operations in Webex Calling.

human_text: What redundancy features are built into Webex Calling? assistant_text: Redundancy features in Webex Calling include geo-redundant data centers and automatic failover for continuous service.

human_text: How does Webex Calling handle user authentication? assistant_text: Secure protocols and integration with identity providers for single sign-on (SSO) manage user authentication in Webex Calling.

human_text: What reporting capabilities does Webex Calling provide? assistant_text: Detailed call analytics, usage patterns, and performance metrics are part of Webex Calling's reporting capabilities.

human_text: What integrations does Webex Calling support with CRM systems? assistant_text: Integrations with CRM systems in Webex Calling enhance customer interactions and streamline business processes.

human_text: How does Webex Calling support compliance with regulations? assistant_text: Compliance features in Webex Calling include call recording, monitoring, and secure data handling.

human_text: What is the role of SRTP in Webex Calling? assistant_text: SRTP encrypts voice media streams to ensure secure communication in Webex Calling.

human_text: How does Webex Calling manage user profiles? assistant_text: Webex Control Hub manages user profiles in Webex Calling, enabling configuration of settings and permissions by administrators.

human_text: How does Webex Calling support video conferencing? assistant_text: Webex Calling supports video conferencing via integration with Webex Meetings.

human_text: What is the role of a Local Gateway in Webex Calling? assistant_text: The Local Gateway connects to the PSTN and handles call routing between Webex Calling and on-premises PBX systems.

human_text: How does Webex Calling handle number porting? assistant_text: Webex Calling allows users to port existing phone numbers from another provider.

human_text: How does Webex Calling support disaster recovery? assistant_text: Geo-redundant data centers and automatic failover mechanisms support disaster recovery in Webex Calling.

human_text: What is the Webex Calling architecture? assistant_text: Webex Calling has a cloud-based architecture with distributed data centers for high availability and resilience.

human_text: How does Webex Calling support call encryption? assistant_text: SRTP and TLS are used to encrypt call signaling and media in Webex Calling.

human_text: What are the benefits of Webex Calling for small businesses? assistant_text: Small businesses benefit from enterprise-grade calling features, scalability, and collaboration tool integration at a lower cost with Webex Calling.

human_text: How does Webex Calling integrate with existing collaboration tools? assistant_text: Integration with Webex Teams and Microsoft Teams offers a seamless collaboration experience in Webex Calling.

human_text: What analytics features are available in Webex Calling? assistant_text: Analytics features in Webex Calling include call quality monitoring, usage reports, and performance dashboards.

human_text: How does Webex Calling support mobile devices? assistant_text: The Webex App enables Webex Calling users to make and receive calls on mobile devices.

human_text: What is the role of the Webex App in Webex Calling? assistant_text: The Webex App offers a unified interface for messaging, video conferencing, and calling in Webex Calling.

human_text: How does Webex Calling handle multi-location deployments? assistant_text: Centralized management and consistent calling features support multi-location deployments in Webex Calling.

human_text: What troubleshooting tools are available in Webex Calling? assistant_text: Troubleshooting tools in Webex Calling include diagnostic reports, call logs, and real-time call quality monitoring.

human_text: How does Webex Calling ensure voice quality? assistant_text: Voice quality in Webex Calling is ensured by network optimization, advanced codecs, and QoS policies.

human_text: What integrations does Webex Calling support? assistant_text: Integrations with CRM systems, collaboration tools, and contact center solutions are supported by Webex Calling.

human_text: How does Webex Calling handle call recording? assistant_text: Call recording in Webex Calling is available for compliance, training, and quality assurance.

human_text: What role do data centers play in Webex Calling? assistant_text: Data centers provide redundancy, high availability, and disaster recovery in Webex Calling.

human_text: What are the benefits of Webex Calling for remote teams? assistant_text: Enterprise-grade calling features, mobility, and collaboration tool integration benefit remote teams using Webex Calling.

human_text: How does Webex Calling handle call forwarding? assistant_text: Flexible call forwarding options in Webex Calling allow users to forward calls to other numbers or devices.

human_text: What is Webex Calling? assistant_text: Webex Calling is a cloud-based phone system offering enterprise-grade calling features.

human_text: What types of phones are supported by Webex Calling? assistant_text: Webex Calling supports Cisco MPP Series IP Phones, Webex Devices, and

various third-party devices.

human_text: How does Webex Calling integrate with other Webex services? assistant_text: Webex Calling integrates with Webex Meetings, Webex Teams, and Webex Contact Center.

human_text: What are the deployment options for Webex Calling? assistant_text: Webex Calling offers cloud-only and hybrid deployment options.

human_text: How is call quality maintained in Webex Calling? assistant_text: Advanced codecs, network optimization, and QoS policies maintain call quality in Webex Calling.

human_text: What is the purpose of the Webex Control Hub? assistant_text: The Webex Control Hub centralizes management for provisioning, configuring, and managing Webex Calling services.

human_text: How does Webex Calling support remote work? assistant_text: Webex Calling supports remote work by offering enterprise-grade calling features and connectivity from any location with internet access.

human_text: What security measures are implemented in Webex Calling? assistant_text: Encryption, secure voice, and industry-standard compliance are key security measures in Webex Calling.

human_text: How does Webex Calling handle emergency calls? assistant_text: Emergency calls in Webex Calling are handled by comparing the dial string with defined emergency numbers.

human_text: What are the benefits of Webex Calling for enterprises? assistant_text: Lower maintenance costs, scalability, remote work support, and collaboration tool integration are benefits for enterprises using Webex Calling.

human_text: How does Webex Calling manage call routing? assistant_text: Dial plans and route groups manage call routing in Webex Calling.

human_text: What customer support is available for Webex Calling? assistant_text: Webex Calling customer support includes technical assistance, training, and deployment resources from Cisco.

human_text: How does Webex Calling ensure high availability? assistant_text: Geo-redundant data centers and a redundant global backbone network ensure high availability in Webex Calling.

human_text: What are the international calling capabilities of Webex Calling? assistant_text: International calling is supported by Webex Calling's global data center network for high-quality voice communication.

human_text: What is the role of SIP in Webex Calling? assistant_text: SIP manages signaling and multimedia communication sessions in Webex Calling.

human_text: How does Webex Calling handle compliance with local regulations? assistant_text: Compliance with local regulations is ensured by routing calls through regional data centers and supporting lawful intercept.

human_text: What integration options are available for Webex Calling? assistant_text: Webex Calling integrates with Webex Meetings, Webex Teams, and third-party tools for enhanced productivity.

human_text: How does Webex Calling support contact centers? assistant_text: Advanced contact center features like call routing, IVR, and analytics are provided through Webex Contact Center integration.

human_text: What is the function of SRTP in Webex Calling? assistant_text: SRTP encrypts media streams to ensure secure communication in Webex Calling.

human_text: How does Webex Calling handle video calls? assistant_text: Video calls are supported between video-capable MPP phones, Webex Devices, and the Webex App in Webex Calling.

human_text: What are the benefits of Webex Calling for remote workers? assistant_text: Webex Calling offers remote workers enterprise-grade calling features and connectivity from any location with internet access.

human_text: What is the role of the Webex Control Hub in managing Webex Calling? assistant_text: The Webex Control Hub centralizes management for provisioning, configuring, and managing Webex Calling services.

human_text: How does Webex Calling integrate with existing on-premises PBX systems? assistant_text: Webex Calling integrates with on-premises PBX systems via Local Gateways for hybrid deployment.

human_text: What is the significance of SRTP in Webex Calling? assistant_text: SRTP encrypts media streams in Webex Calling to ensure secure communication.

human_text: How does Webex Calling support compliance with local regulations? assistant_text: Webex Calling complies with local regulations by using regional data centers and supporting lawful intercept.

human_text: What features are available for call management in Webex Calling? assistant_text: Webex Calling offers call forwarding, call transfer, call hold, and voicemail features.

human_text: What options are available for integrating Webex Calling with other collaboration tools? assistant_text: Webex Calling integrates with Webex Meetings, Webex Teams, and third-party tools for enhanced productivity.

human_text: How does Webex Calling handle call routing? assistant_text: Dial plans and route groups manage call routing in Webex Calling.

human_text: What security features are built into Webex Calling? assistant_text: Webex Calling features encryption, secure voice, and industry-standard compliance for communication security.

human_text: How does Webex Calling support mobile users? assistant_text: The Webex App enables mobile users to make and receive calls on their devices with Webex Calling.

human_text: How does Webex Calling integrate with Cisco devices? assistant_text: Cisco MPP phones and Webex Room devices integrate with Webex Calling for unified communication.

human_text: What are the deployment models available for Webex Calling? assistant_text: Webex Calling supports cloud-only and hybrid deployment models.

human_text: How does Webex Calling handle international calling? assistant_text: International calling is supported by Webex Calling's global data center network for high-quality voice communication.

human_text: What features does Webex Calling offer for contact centers? assistant_text: Advanced contact center features like call routing, IVR, and analytics are offered through Webex Contact Center integration.

human_text: How does Webex Calling support scalability? assistant_text: The cloud-based architecture of Webex Calling supports scalability for businesses of all sizes.

human_text: What type of customer support is available for Webex Calling? assistant_text: Webex Calling customer support includes technical assistance, training, and deployment resources from Cisco.

human_text: How does Webex Calling handle voice quality? assistant_text: High voice quality in Webex Calling is ensured by advanced codecs, network optimization, and QoS policies.

human_text: What are the benefits of using Webex Calling over traditional PBX systems? assistant_text: Benefits of Webex Calling include lower maintenance costs, scalability, remote work support, and collaboration tool integration.

human_text: What signaling and media protocols does Webex Calling use? assistant_text: Webex Calling uses SIP for signaling and SRTP for media.

human_text: What is the role of the Webex Control Hub in Webex Calling? assistant_text: The Webex Control Hub provides connection parameters and digest credentials for SIP authentication during Local Gateway registration.

human_text: What group features does Webex Calling provide? assistant_text: Webex Calling provides group features like unlimited subscriptions of auto-attendants, hunt groups, and call queues.

human_text: What are the regional platforms for Webex Calling? assistant_text: Webex Calling operates regional platforms in the US, Canada, UK, Europe, APAC, Japan, and APJC Australia.

human_text: What role do load balancers play in Webex Calling datacenters? assistant_text: Load balancers are used to build a scalable, redundant datacenter architecture.

human_text: What are the capabilities of the Webex App in Webex Calling? assistant_text: The Webex App supports mid-call features, rich presence, and control of the user's Cisco MPP phone.

human_text: What is the role of the Webex Control Hub in Local Gateway registration? assistant_text: The Webex Control Hub provides connection parameters and digest credentials for SIP authentication during Local Gateway registration.

human_text: What deployment options are available for Webex Calling? assistant_text: Webex Calling can be deployed as a cloud-only solution or as part of a hybrid cloud.

human_text: What are the data center locations for Webex Calling in the US? assistant_text: Webex Calling data centers in the US are located in Dallas, Chicago, and New York.

human_text: What capabilities does the Webex App offer when integrated with Webex Calling? assistant_text: The Webex App offers messaging, screen sharing, audio and video conferencing, and integrated calling with mid-call features or control of a user's desk phone.

human_text: What are the considerations for video calls in Webex Calling? assistant_text: Video calls can be made within a single Webex Calling org between video-capable devices or Webex App, while PSTN only supports voice calls.

human_text: What are the main functions hosted in each Webex Calling datacenter? assistant_text: Webex Calling datacenters host call routing functions, provide provisioning interface access, and host access and peering SBCs.

human_text: How does Webex Calling ensure firewall traversal for calls? assistant_text: Webex Calling ensures firewall traversal by using TLS connections initiated by phones and Local Gateways, and sending traffic back through the same connection.

human_text: How are trunks used in Webex Calling? assistant_text: Trunks connect Webex Calling with Local Gateways or Dedicated Instances.

human_text: What is the Private Network Connect (PNC) solution? assistant_text: The Private Network Connect (PNC) solution extends private networks to the cloud for high quality of service and low latency.

```

human_text: What type of subscription is Webex Calling based on? assistant_text: Webex Calling uses a subscription-based licensing model managed with the Cisco Collaboration Flex Plan.
human_text: Where are Webex Calling data centers located? assistant_text: Webex Calling data centers are located in the US, Canada, Europe, UK, Australia, and Japan.
human_text: How does Webex Calling handle emergency calls? assistant_text: Emergency calls are handled by comparing the dial string with emergency numbers defined in the national numbering plan.
human_text: How are unknown numbers handled in Webex Calling? assistant_text: Unknown numbers are handled based on the 'Unknown Number Handling' and 'Calls to On-Premises Extension' settings.
human_text: What is the purpose of dial plans in Webex Calling? assistant_text: Dial plans enable call routing to premises-based call control instances based on dial patterns.
human_text: What are the benefits of using route groups in Webex Calling? assistant_text: Route groups provide orangeundancy and increased capacity by grouping multiple trunks together.
human_text: What happens if ICE negotiation fails in Webex Calling? assistant_text: If ICE negotiation fails, media is anchored on the Webex Calling Access SBC, resulting in media flowing through the customer's Internet edge to the SBC and back to the destination endpoint.
human_text: What are route groups in Webex Calling? assistant_text: Route groups provide orangeundancy or increased capacity by grouping multiple trunks together.
human_text: What is requiorange for Webex Calling endpoints to connect to the datacenters? assistant_text: Webex Calling endpoints use the public Internet to connect to datacenters and establish over-the-top TLS connections.
human_text: What features does the Dedicated Instance option provide in Webex Calling? assistant_text: The Dedicated Instance option provides a Cisco Unified Communications Manager based stack of applications in a private cloud dedicated to a single customer.
human_text: How does Webex Calling handle PSTN access? assistant_text: Webex Calling handles PSTN access through Cisco Calling Plans, Cloud Connected PSTN, and Premises-based PSTN.
human_text: What are the benefits of the Webex Calling global backbone? assistant_text: The global backbone optimizes media round-trip times and ensures high availability with multi-gigabit, fully orangeundant network.
human_text: What is the purpose of a Local Gateway in Webex Calling? assistant_text: The Local Gateway connects to the PSTN and registers with Webex Calling to handle all calls between the PSTN and Webex Calling.
human_text: What is the significance of media path optimization in Webex Calling? assistant_text: Media path optimization establishes a direct media path between entities to orangeuce bandwidth usage and improve call quality.
human_text: What is Webex Edge Connect? assistant_text: Webex Edge Connect peers Webex meetings and Webex Calling traffic with an Equinix Cloud Exchange (ECX) location to improve user experience with guaranteed bandwidth and QoS.
human_text: What is the Webex Calling solution overview? assistant_text: Webex Calling provides enterprise-grade PBX features, functionality, and performance.
human_text: What connectivity options are available for Webex Calling? assistant_text: Connectivity options for Webex Calling include Over-the-top (OTT) Internet, Webex Edge Connect, and Private Network Connect.
human_text: What type of patterns can be included in a Webex Calling dial plan? assistant_text: Dial plans can include numeric patterns and domain patterns for routing SIP URIs.
human_text: What is the role of Local Gateways in Webex Calling? assistant_text: Local Gateways provide PSTN access and connect Webex Calling to existing on-premises call control services.
human_text: How does Webex Calling handle calls between different customers? assistant_text: Calls between different Webex Calling customers are routed through the PSTN.
human_text: What types of phones are supported by Webex Calling? assistant_text: Webex Calling supports all models of Cisco Multiplatform Phones (MPP).
human_text: What is Webex Edge Connect? assistant_text: Webex Edge Connect peers Webex meetings and Webex Calling traffic with an Equinix Cloud Exchange location, improving user experience with guaranteed bandwidth and QoS.
human_text: How can video calls be made in Webex Calling? assistant_text: Video calls in Webex Calling can be made between video capable MPP phones, Webex Devices, and Webex App.
human_text: What is the Dedicated Instance option in Webex Calling? assistant_text: The Dedicated Instance option provides a Cisco Unified Communications Manager based stack of applications in a private cloud dedicated to a single customer.
human_text: How many participants can Webex Meetings support when added to Webex Calling? assistant_text: Webex Meetings can support up to 1000 meeting participants.

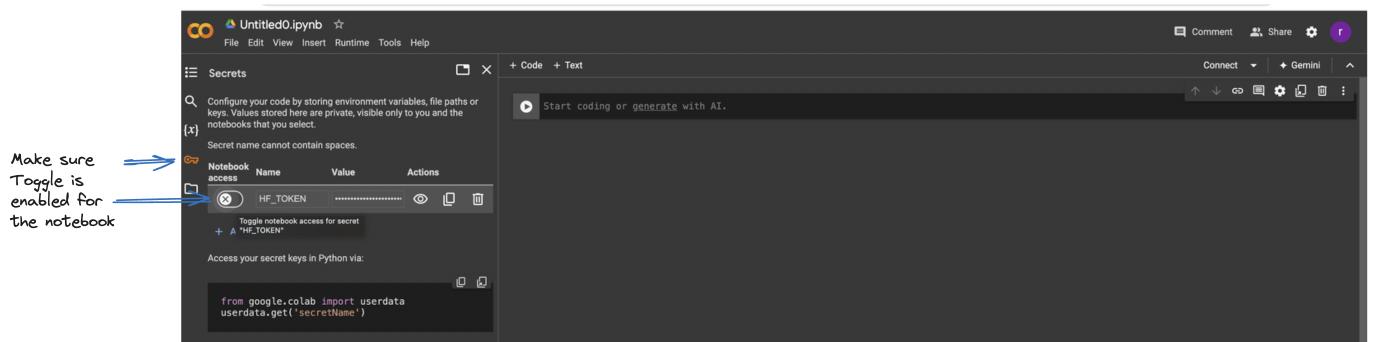
```

Save the file as example.txt as we will be using it in the next step

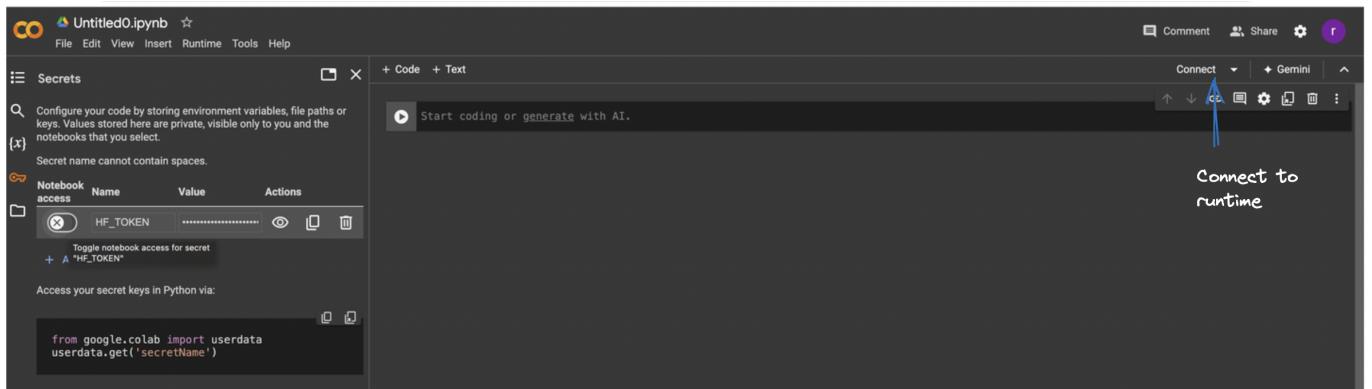
To finalize, we again convert the above dataset (example.txt) into Llama2 format. Lets look into those steps

CONVERT DATASET INTO LLAMA2 FORMAT AND UPLOAD ON HUGGING FACE

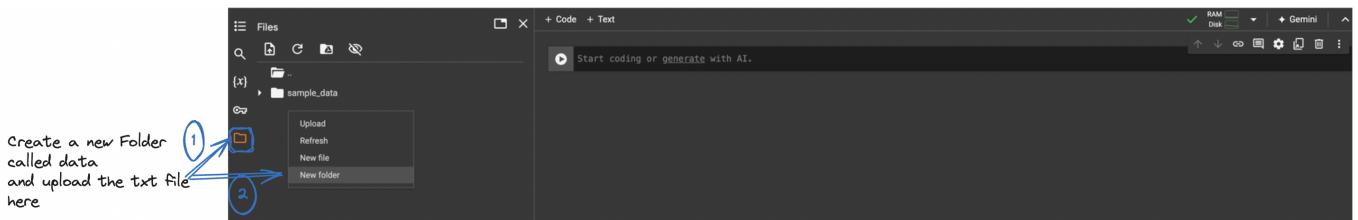
- Open Google Colab and create a new notebook. Click on "File" > "New notebook". Go to the "Secrets" section in the sidebar and ensure the Hugging Face toggle is enabled. Please refer to the following section to create Google Colab account.



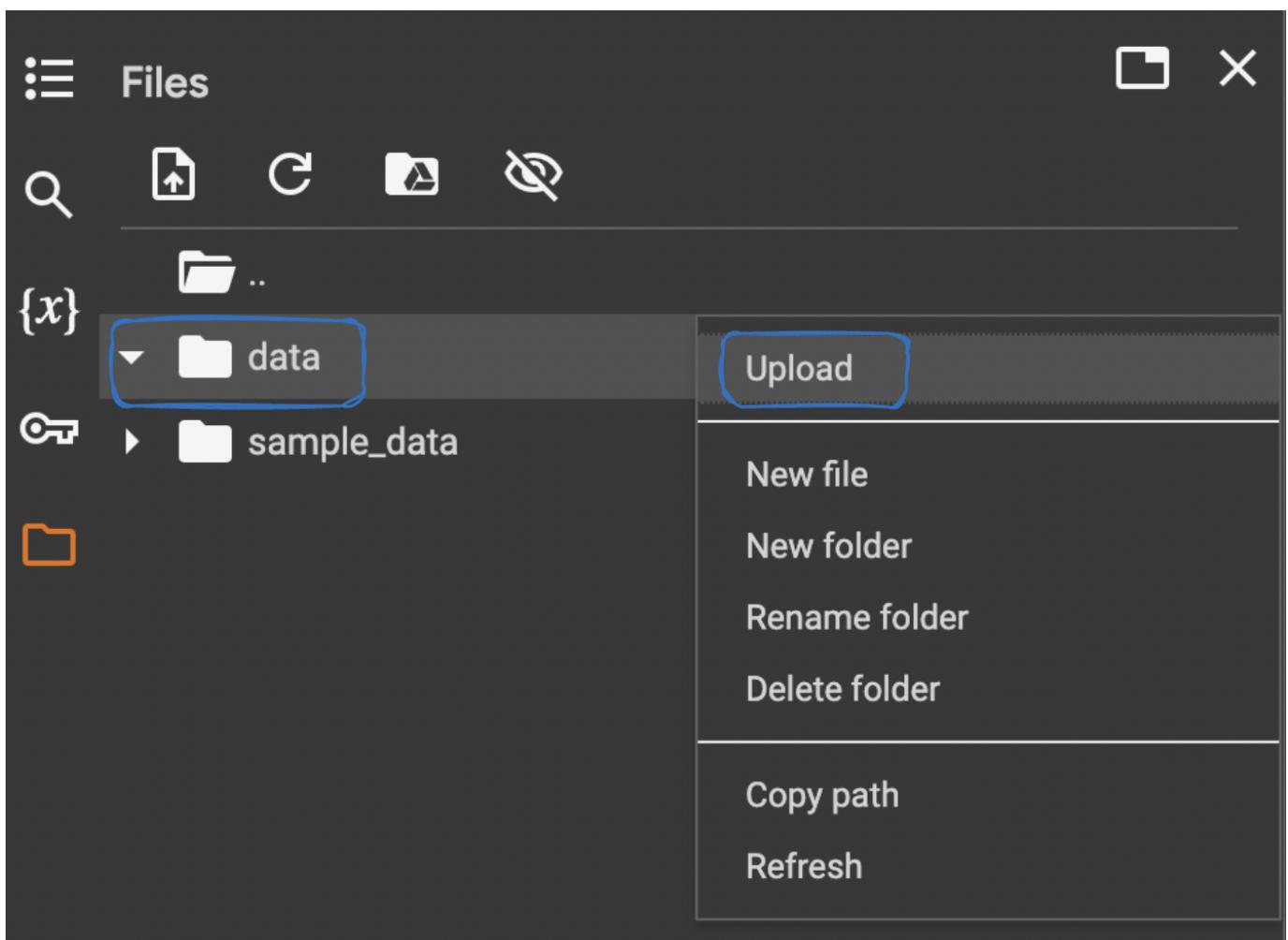
- Make sure you are connected to a runtime. For this task, you can use the CPU as the runtime environment.



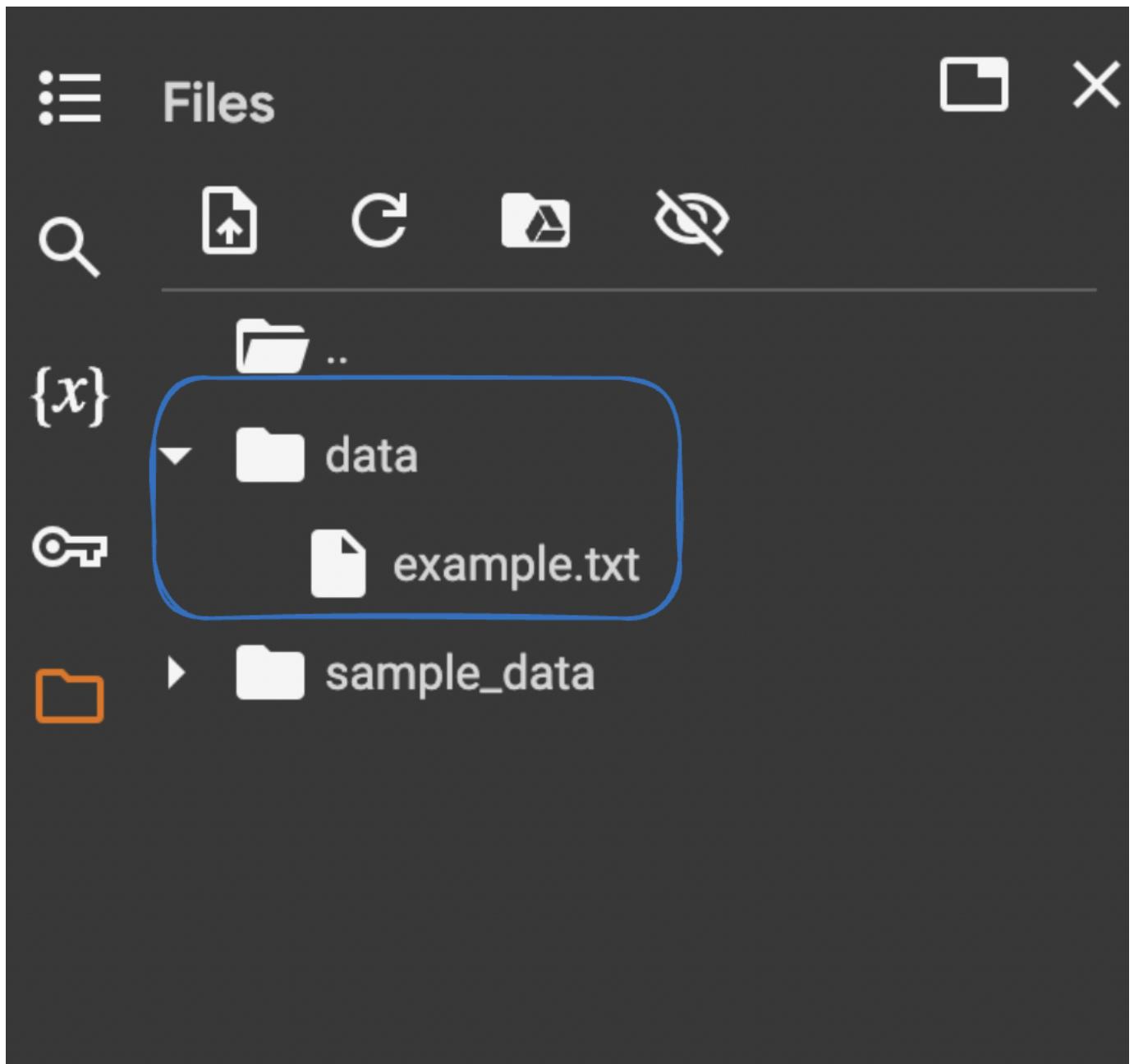
- Click on Folder and create a new folder called "data"



- Click on [...], select Upload



- Choose your example.txt file and click Open



Note Ensure that your files are saved elsewhere. This runtime's files will be deleted when it is terminated.

Summary: So far, we have our raw dataset as example.txt, enabled Hugging Face in our Colab notebook, and uploaded the data into our folder. We will be converting the dataset (example.txt) into Llama2 format and uploading on Hugging Face so it can be used for our fine tuning in the upcoming steps.

We will start by installing specific Python packages.

```
!pip install datasets huggingface_hub google-colab
```

```

☒ Collecting datasets
  Downloading datasets-2.20.0-py3-none-any.whl.metadata (19 kB)
Requirement already satisfied: huggingface_hub in /usr/local/lib/python3.10/dist-packages (0.23.5)
Requirement already satisfied: google-colab in /usr/local/lib/python3.10/dist-packages (1.0.0)
Requirement already satisfied: filelock in /usr/local/lib/python3.10/dist-packages (from datasets) (3.15.4)
Requirement already satisfied: numpy>=1.17 in /usr/local/lib/python3.10/dist-packages (from datasets) (1.25.2)
Collecting pyarrow>=15.0.0 (from datasets)
  Downloading pyarrow-17.0.0-cp310-cp310-manylinux_2_28_x86_64.whl.metadata (3.3 kB)
Requirement already satisfied: pyarrow-hotfix in /usr/local/lib/python3.10/dist-packages (from datasets) (0.6)
Collecting dill<0.3.9,>=0.3.0 (from datasets)
  Downloading dill-0.3.8-py3-none-any.whl.metadata (10 kB)
Requirement already satisfied: pandas in /usr/local/lib/python3.10/dist-packages (from datasets) (2.0.3)
Collecting requests>=2.32.2 (from datasets)
  Downloading requests-2.32.3-py3-none-any.whl.metadata (4.6 kB)
Requirement already satisfied: tqdm>=4.66.3 in /usr/local/lib/python3.10/dist-packages (from datasets) (4.66.4)
Collecting xxhash (from datasets)
  Downloading xxhash-3.4.1-cp310-cp310-manylinux_2_17_x86_64_manylinux2014_x86_64.whl.metadata (12 kB)
Collecting multiprocess (from datasets)
  Downloading multiprocess-0.70.16-py310-none-any.whl.metadata (7.2 kB)
Collecting fsspec<=2024.5.0,>=2023.1.0 (from fsspec[http]<=2024.5.0,>=2023.1.0->datasets)
  Downloading fsspec-2024.5.0-py3-none-any.whl.metadata (11 kB)
Requirement already satisfied: aiohttp in /usr/local/lib/python3.10/dist-packages (from datasets) (3.9.5)
Requirement already satisfied: packaging in /usr/local/lib/python3.10/dist-packages (from datasets) (24.1)
Requirement already satisfied: pyyaml>=5.1 in /usr/local/lib/python3.10/dist-packages (from datasets) (6.0.1)
Requirement already satisfied: typing-extensions>=3.7.4.3 in /usr/local/lib/python3.10/dist-packages (from huggingface_hub) (4.12.2)
Requirement already satisfied: google-auth==2.27.0 in /usr/local/lib/python3.10/dist-packages (from google-colab) (2.27.0)
Requirement already satisfied: ipykernel==5.5.6 in /usr/local/lib/python3.10/dist-packages (from google-colab) (5.5.6)
Requirement already satisfied: ipyparallel==8.8.0 in /usr/local/lib/python3.10/dist-packages (from google-colab) (8.8.0)
Requirement already satisfied: ipython==7.34.0 in /usr/local/lib/python3.10/dist-packages (from google-colab) (7.34.0)
Requirement already satisfied: notebook==6.5.5 in /usr/local/lib/python3.10/dist-packages (from google-colab) (6.5.5)
Requirement already satisfied: portpicker==1.5.2 in /usr/local/lib/python3.10/dist-packages (from google-colab) (1.5.2)
INFO: pip is looking at multiple versions of google-colab to determine which version is compatible with other requirements. This could take a while.
Collecting huggingface_hub
  Downloading huggingface_hub-0.24.3-py3-none-any.whl.metadata (13 kB)
  Downloading huggingface_hub-0.24.2-py3-none-any.whl.metadata (13 kB)
  Downloading huggingface_hub-0.24.1-py3-none-any.whl.metadata (13 kB)
  Downloading huggingface_hub-0.24.0-py3-none-any.whl.metadata (13 kB)
  Downloading huggingface_hub-0.23.5-py3-none-any.whl.metadata (12 kB)
  Downloading huggingface_hub-0.23.4-py3-none-any.whl.metadata (12 kB)
  Downloading huggingface_hub-0.23.3-py3-none-any.whl.metadata (12 kB)
INFO: pip is still looking at multiple versions of google-colab to determine which version is compatible with other requirements. This could take a while.
  Downloading huggingface_hub-0.23.2-py3-none-any.whl.metadata (12 kB)
  Downloading huggingface_hub-0.23.1-py3-none-any.whl.metadata (12 kB)
  Downloading huggingface_hub-0.23.0-py3-none-any.whl.metadata (12 kB)
  Downloading huggingface_hub-0.22.2-py3-none-any.whl.metadata (12 kB)
  Downloading huggingface_hub-0.22.1-py3-none-any.whl.metadata (12 kB)

```

- The ! at the beginning is used in Jupyter notebooks or Google Colab to run shell commands. Using pip install we will install Python packages
- datasets: This package is part of the Hugging Face ecosystem and provides tools for working with large datasets. It allows users to easily download, preprocess, and manage datasets, especially those used in machine learning and natural language processing (NLP).
- huggingface_hub: This package provides tools to interact with the Hugging Face Hub.
- google-colab: This package includes utilities specifically designed for Google Colab, a Jupyter notebook environment that runs in the cloud.

Step 1: Import the requiorange modules

```

1  # Import requiorange modules
2  from datasets import Dataset
3  from huggingface_hub import login
4  import os
5  from google.colab import userdata

```

- from datasets import Dataset: imports the Dataset class from the datasets library. The Dataset class is used to create and manipulate datasets. This library is part of the Hugging Face ecosystem and is especially useful for handling datasets for machine learning and NLP tasks.
- from huggingface_hub import login: This imports the login function from the huggingface_hub library. The login function is used to authenticate with the Hugging Face Hub, allowing the user to upload and manage models and datasets on the platform.
- import os: This imports the os module, which provides a way of using operating system-dependent functionality like reading or writing to the file system, environment variables, and more.
- from google.colab import userdata: imports the userdata module from the google.colab library. The google.colab library contains utilities specifically designed for use with Google Colab.

Step 2: Retrieve Hugging Face token from Colab secrets

```

1  # Retrieve Hugging Face token from Colab secrets
2  os.environ["HF_TOKEN"] = userdata.get('HF_TOKEN')
3  # Login to Hugging Face
4  login(token=os.environ["HF_TOKEN"])

```

```
☒ The token has not been saved to the git credentials helper. Pass `add_to_git_credential=True` in this function directly or `--add-to-git-credential` if using via `Token is valid (permission: write). Your token has been saved to /root/.cache/huggingface/token Login successful
```

- This part of the code retrieves a previously stored Hugging Face token from Colab secrets. This token is essential for authenticating with the Hugging Face platform. Storing tokens in Colab secrets is a secure way to handle sensitive information without hardcoding it in your script.
- Login to Hugging Face: This line logs into Hugging Face using the retrieved token. Logging in allows our code to interact with the Hugging Face Hub, and allow us to upload datasets and models.

Step 3: Define a function that will help us convert our example.txt into Llama2 format

```
1  # Define the generator function
2  def data_generator(file_path):
3      with open(file_path, 'r') as file:
4          for line in file:
5              if "assistant_text: " in line:
6                  parts = line.split("assistant_text: ")
7                  human_text = parts[0].replace("human_text: ", "").strip()
8                  assistant_text = parts[1].strip()
9                  reformatted_segment = f'<s>[INST] {human_text} [/INST] {assistant_text} </s>'
10                 yield {"reformatted_segment": reformatted_segment}
```

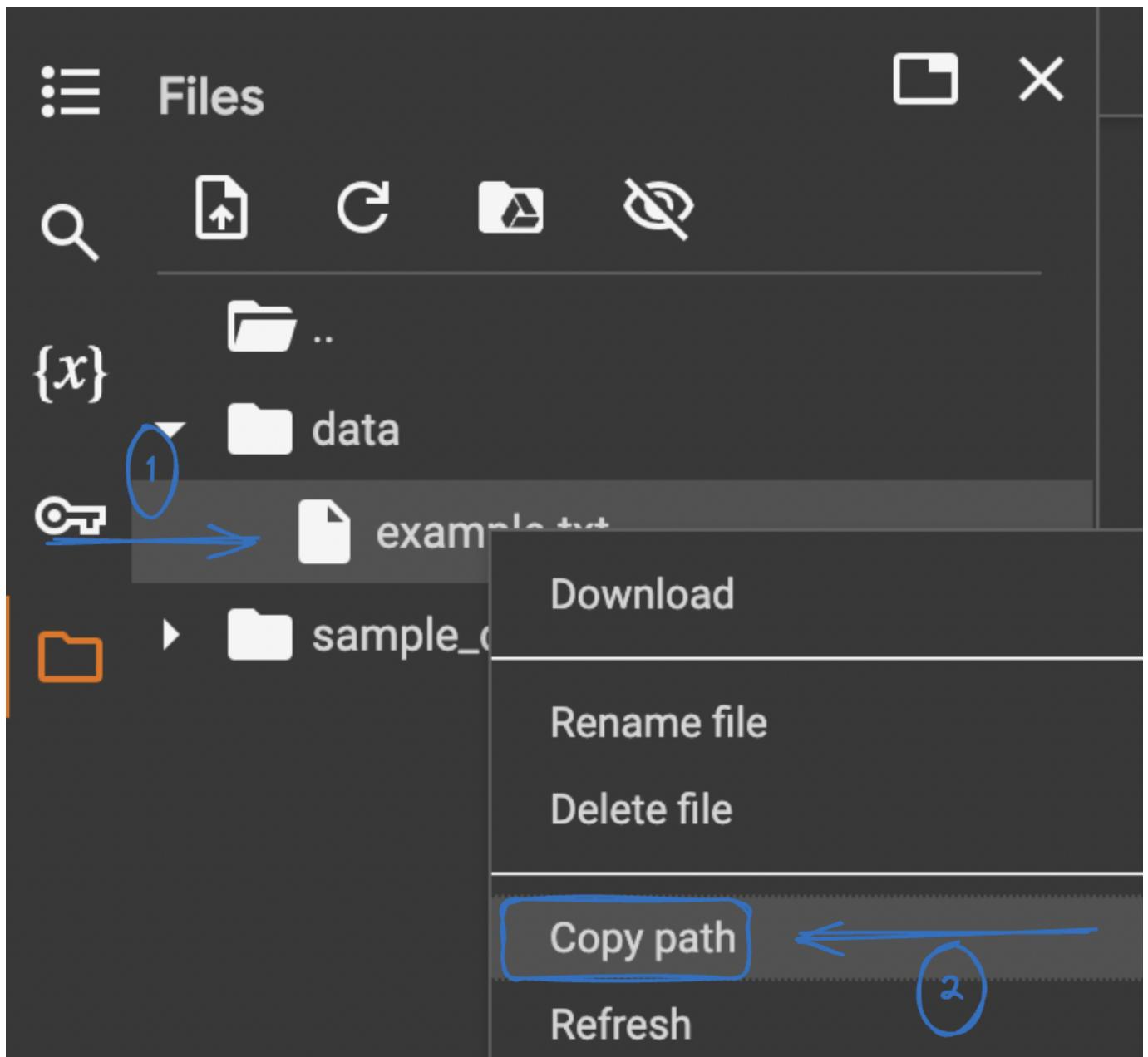
- def data_generator(file_path): defines a function named data_generator that takes a single argument, file_path.
- with open(file_path, 'r') as file: opens the file specified by file_path in read mode ('r').
- for line in file: iterates over each line in the file.
- if "assistant_text: " in line: checks if the string "assistant_text: " is present in the current line (example.txt) file
- parts = line.split("assistant_text: "): splits the current line into two parts using "assistant_text: " as the delimiter.
- human_text = parts[0].replace("human_text: ", "").strip(): takes the first part of the split line (parts[0]), replaces "human_text: " with an empty string, and then removes any leading or trailing whitespace using the strip() method.
- assistant_text = parts[1].strip(): takes the second part of the split line (parts[1]) and removes any leading or trailing whitespace using the strip() method.
- reformatted_segment = Creates a formatted string. The human_text and assistant_text variables are inserted into the string at the specified locations, to replicate Llama2 format.
- yield {"reformatted_segment": reformatted_segment}: the yield statement returns a dictionary containing the reformatted_segment. The yield statement makes this function a generator, allowing it to produce a sequence of values over time, rather than returning them all at once.

Step 4: Define the path for the file that we want to convert

```
1  # Path to your data file
2  file_path = '/content/data/example.txt'
```

Note: How to get path for your file in Google Colab

- Right click on your file (example.txt) and select Copy Path



Step 5: Calling data_generator function

```
1 # Create the dataset from the generator
2 ds = Dataset.from_generator(data_generator, gen_kwargs={"file_path": file_path})
```

- Creating a Dataset object by passing the file (example.txt) to the data_generator function

Step 6: Verify if our data formatted

```
1 # Access the dataset
2 print(ds[2])
```

{'reformatted_segment': '<s>[INST] What type of devices does Webex Calling support? [/INST] Webex Calling supports Cisco MPP Series IP Phones, Webex Devices, and a variety of third-party devices. </s>'}



Llama2 Format
as per the syntax

Step 7: Verify values in our variable ds

```
1 # Access the dataset
2 print(ds)
```

```
→ Dataset({
    features: ['reformatted_segment'],
    num_rows: 161
})
```

Step 8: Consolidates all the data and create new Dataset

```
1 reformatted_segments_list = []
2 # Iterate through the dataset and collect reformatted segments
3 for example in ds:
4     reformatted_segment = example["reformatted_segment"]
5     reformatted_segments_list.append(reformatted_segment)
6 # Now you have all reformatted segments in reformatted_segments_list
7 print("Total reformatted segments:", len(reformatted_segments_list))
8 print("First reformatted segment example:", reformatted_segments_list[0])
9 # Create a new Dataset object with these reformatted segments
10 reformatted_ds = Dataset.from_dict({"text": reformatted_segments_list}) # Assuming downstream processes expect 'text'
```

- The above code snippet iterates over the original dataset (ds), extracts the reformatted segments, and collects them into a list (reformatted_segments_list). This process consolidates all the relevant data into a single list for further processing.
- After collecting all the reformatted segments, we will create a new Dataset object (reformatted_ds) from the list. This new dataset is structured in a way that is required by downstream processes when fine tuning starts.

```
→ Total reformatted segments: 161
First reformatted segment example: <s>[INST] What is the solution provided by Webex Calling? [/INST] Webex Calling provides enterprise-grade PBX features, functionality, and performance. </s>
Saving the dataset (1/1 shards): 100% [1/1 shards] 161/161 [00:00<00:00, 6311.47 examples/s]
```

Step 9: Push our new Dataset to Hugging Face Hub so it can be processed and used for fine tuning

```
1 # Push the dataset to the Hub
2 reformatted_ds.push_to_hub("WebexOne/test")
```

```
→ Uploading the dataset shards: 100% [1/1 [00:00<00:00, 1.80it/s]
Creating parquet from Arrow format: 100% [1/1 [00:00<00:00, 24.55ba/s]
CommitInfo(commit_url='https://huggingface.co/datasets/WebexOne/test/commit/c97d23e49ceff41ba8a7372277ace339d3b4301a', commit_message='Upload dataset', commit_description='', oid='c97d23e49ceff41ba8a7372277ace339d3b4301a', pr_url=None, pr_revision=None, pr_num=None)
```

- The push_to_hub method is used to upload the dataset reformatted_ds to the Hugging Face Hub. This makes the dataset publicly available (or private, depending on the repository settings) for others to access and use.
- Repository Naming: The string "WebexOne/test" specifies the target repository on the Hugging Face Hub.

Let's login to Hugging Face and view our uploaded model

The screenshot shows the Hugging Face platform interface. At the top, there's a search bar and a navigation bar with links for Models, Datasets, Spaces, Posts, Docs, Solutions, Pricing, and a user profile icon. Below the header, the organization name 'WebexOne' is displayed with a 'Classroom' badge. A call-to-action button 'Upgrade to Enterprise' is visible. On the left, there are sections for 'AI & ML interests' (None defined yet) and 'Team members' (1). In the center, a message says 'No organization card' with a link to 'Create a Card'. On the right, there's a section for 'Models' (None yet) and a 'Datasets' section. The 'Datasets' section lists a single entry: 'WebexOne/test' (Viewer · Updated about 20 hours ago · 161 rows). A blue arrow points from the text 'Our dataset uploaded' to this dataset entry.

Our uploaded Dataset

This screenshot shows the detailed view of the 'WebexOne/test' dataset. At the top, it displays the dataset name, a 'Split (1)' section showing 'train · 161 rows', and a search bar. Below this is a 'text' section with a 'string · lengths' chart. The main area contains a large list of questions and their answers, all starting with '<s>[INST]'. The text is too long to reproduce here but follows a consistent pattern of asking a question and providing a detailed answer. A blue arrow points from the text 'Our dataset uploaded' to the dataset name 'WebexOne/test' in the list.

Complete Code - FOR REFERENCE ONLY

```

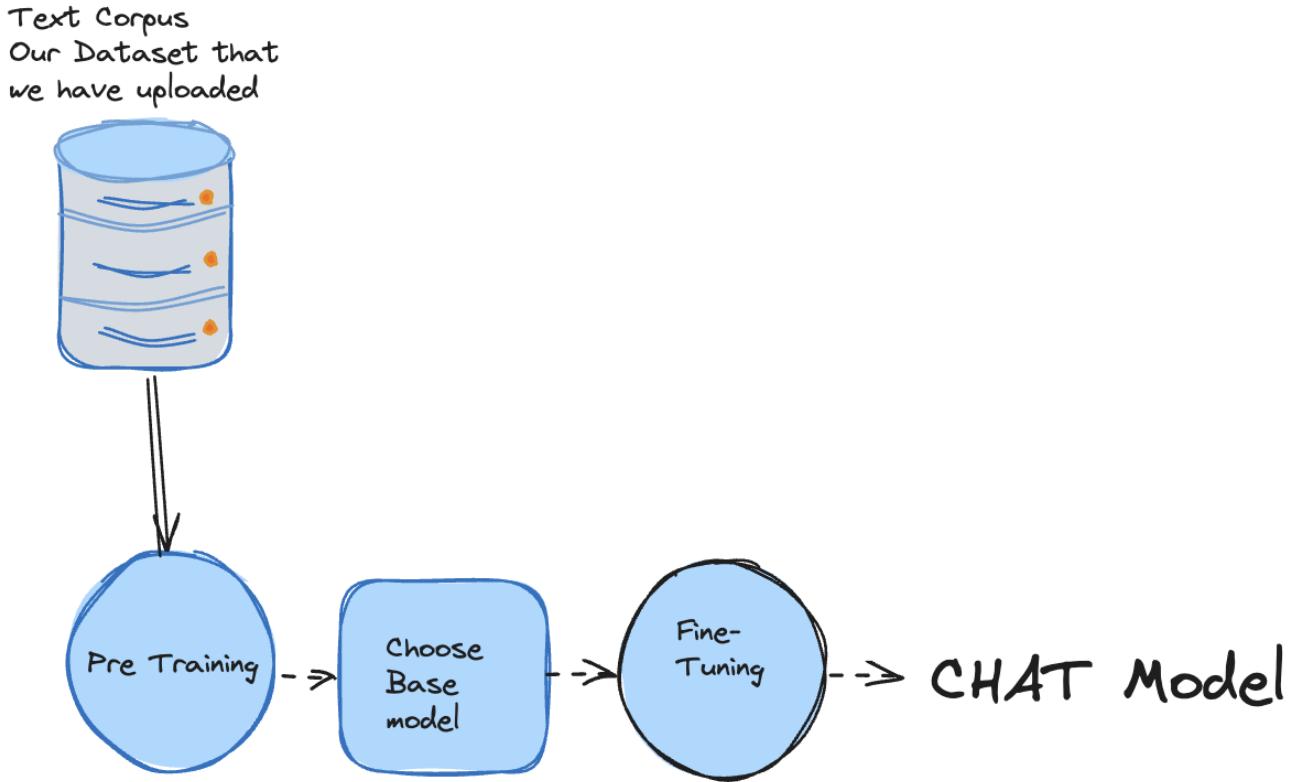
1  !pip install datasets huggingface_hub google-colab
2  # Import requiorange modules
3  from datasets import Dataset
4  from huggingface_hub import login
5  import os
6  from google.colab import userdata
7  # Retrieve Hugging Face token from Colab secrets
8  os.environ["HF_TOKEN"] = userdata.get("HF_TOKEN")
9  # Login to Hugging Face
10 login(token=os.environ["HF_TOKEN"])
11 # Define the generator function
12 def data_generator(file_path):
13     with open(file_path, 'r') as file:
14         for line in file:
15             if "assistant_text: " in line:
16                 parts = line.split("assistant_text: ")
17                 human_text = parts[0].replace("human_text: ", "").strip()
18                 assistant_text = parts[1].strip()
19                 reformatted_segment = f'<s>[INST] {human_text} [/INST] {assistant_text} </s>'
20                 yield {"reformatted_segment": reformatted_segment}
21 # Path to your data file
22 file_path = '/content/data/example.txt'
23 # Create the dataset from the generator
24 ds = Dataset.from_generator(data_generator, gen_kwargs={"file_path": file_path})
25
26 reformatted_segments_list = []
27 # Iterate through the dataset and collect reformatted segments
28 for example in ds:
29     reformatted_segment = example["reformatted_segment"]
30     reformatted_segments_list.append(reformatted_segment)
31
32 # Now you have all reformatted segments in reformatted_segments_list
33 print("Total reformatted segments:", len(reformatted_segments_list))
34 print("First reformatted segment example:", reformatted_segments_list[0])
35
36 # Create a new Dataset object with these reformatted segments
37 reformatted_ds = Dataset.from_dict({"text": reformatted_segments_list}) # Assuming downstream processes expect 'text'
38
39 # Save the dataset locally - OPTIONAL STEP
40 reformatted_ds.save_to_disk('/content/data/reformatted_dataset')
41
42 # Push the dataset to the Hub
43 reformatted_ds.push_to_hub("WebexOne/test")

```

Fine-tuning Llama2-7B Model

NOTE: If you're serious about fine-tuning models, using a script instead of a notebook is recommended. You can easily rent GPUs on Lambda Labs, Runpod, Vast.ai e.t.c

Background on Fine-Tuning



Language models (LLMs) are pretrained on extensive text corpus. For example, Llama 2 was trained with 2 trillion tokens. As mentioned before pretraining is resource-intensive and often faces hardware challenges.

After pretraining, models like Llama 2, can predict the next word in a sequence but don't naturally follow instructions. To make them better assistants, we use instruction tuning, which involves two primary methods:

- Supervised Fine-Tuning (SFT): Models are trained on datasets of instructions and responses to minimize the difference between their answers and the correct ones.
- Reinforcement Learning from Human Feedback (RLHF): Models learn through interaction and feedback to optimize a reward signal based on human evaluations.

RLHF can capture nuanced human preferences but is complex to implement, requiring a well-designed reward system and consistent feedback.
Not a focus for this lab

In this lab, we will use Supervised Fine-Tuning (SFT). Fine-tuning works well because it builds on the knowledge gained during pretraining. If a model has seen similar data, fine-tuning can significantly enhance performance. For instance, fine-tuning a LLaMA model with 65 billion parameters on 1,000 high-quality samples can outperform larger models like GPT-3.

Note: As mentioned earlier, the choice of prompt template is crucial for fine-tuning. In our dataset section, we have converted our data into the following template:

```

<s>[INST] <<SYS>>
{{ system_prompt }}
<</SYS>>
{{ user_message }} [/INST] Model answer </s>
  
```

Note: For this tutorial, we will use a preformatted dataset (WebexOne/test) that was uploaded earlier on Hugging Face. We will apply Supervised Fine-Tuning (SFT) to a base model.

GUIDE TO FINE-TUNING LLAMA 2

In this section, we will learn about all the steps required to fine-tune the Llama 2 model with 7 billion parameters on a T4 GPU with high RAM using Google Colab.

The Colab T4 GPU has a limited 16 GB of VRAM. That is barely enough to store Llama 2-7b's weights ($7b \times 2 \text{ bytes} = 14 \text{ GB}$ in FP16), which means full fine-tuning is not possible, and we need to use parameter-efficient fine-tuning techniques like LoRA and QLoRA.

We will use the QLoRA technique to fine-tune the model in 4-bit precision and optimize VRAM usage. For that, we will use the Hugging Face ecosystem of LLM libraries: [transformers](#), [accelerate](#), [peft](#), [trl](#), and [bitsandbytes](#).

BASE MODELS FOR FINE TUNING

- We have the option to fine-tune the Llama2 Base model ([meta-llama/Llama-2-7b-hf](#)), which can be found in the Hugging Face repository

Gated Repo

You need to share contact information with Meta to access this model

The information you provide will be collected, stored, processed and shared in accordance with the [Meta Privacy Policy](#).

LLAMA 2 COMMUNITY LICENSE AGREEMENT

"Agreement" means the terms and conditions for use, reproduction, distribution and modification of the Llama Materials set forth herein. "Documentation" means the specifications, manuals and documentation accompanying Llama 2 distributed by Meta at <https://ai.meta.com/resources/models-and-libraries/llama-downloads/>. "Licensee" or "you" means you, or your employer or any other person or entity (if you are entering into this Agreement on such person or entity's behalf), of the age required under applicable laws, rules or regulations to provide legal consent and that has legal authority to bi... .

▼ Expand to review

▼ Expand to review and access

Llama 2

Llama 2 is a collection of pretrained and fine-tuned generative text models ranging in scale from 7 billion to 70 billion parameters. This is the repository for the 7B pretrained model, converted for the Hugging Face Transformers format. Links to other models can be found in the index at the bottom.

Model Details

Note: Use of this model is governed by the Meta license. In order to download the model weights and tokenizer, please visit the [website](#) and accept our License before requesting access here.

Meta developed and publicly released the Llama 2 family of large language models (LLMs), a

Downloads last month
1,275,918

Safetensors Model size: 6.74B params Tensor type: F32 FP16

Inference API Examples
My name is Merve and my favorite

Compute ×+Enter
This model is loading

JSON Output Maximize

Spaces using meta-llama/Llama-2-7b-hf 100

- open-llm-leaderboard/open_llm_leaderboard
- Intel/intel_low_bit_llm_leaderboard
- edugarcia/open_pt_llm_leaderboard
- hallucinations-leaderboard/leaderboard
- Illia56/Ask-AI-Youtube
- BAAI/open_cn_llm_leaderboard
- vectora/leaderboard
- speakeash/open_pl_llm_leaderboard
- Intel/powerd_by_intel_llm_leaderboard
- EvanTHU/MotionLLM
- sparse-generative-ai/open-moe-llm-leaderboard
- allenai/URIAL-Bench
- genai-impact/ecologits-calculator

- Since it is a gated repository, you'll need to provide the required information to access and use the model. Please scroll down, fill in the necessary details, and submit the form.

By agreeing you accept to share your contact information (email and username) with the repository authors.

First Name

Last Name

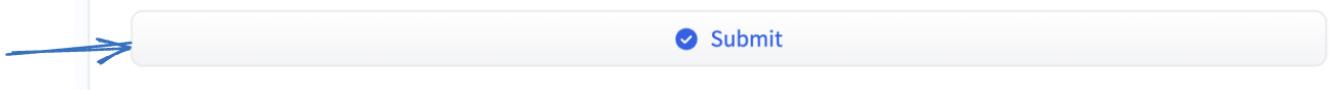
Date of birth

Country

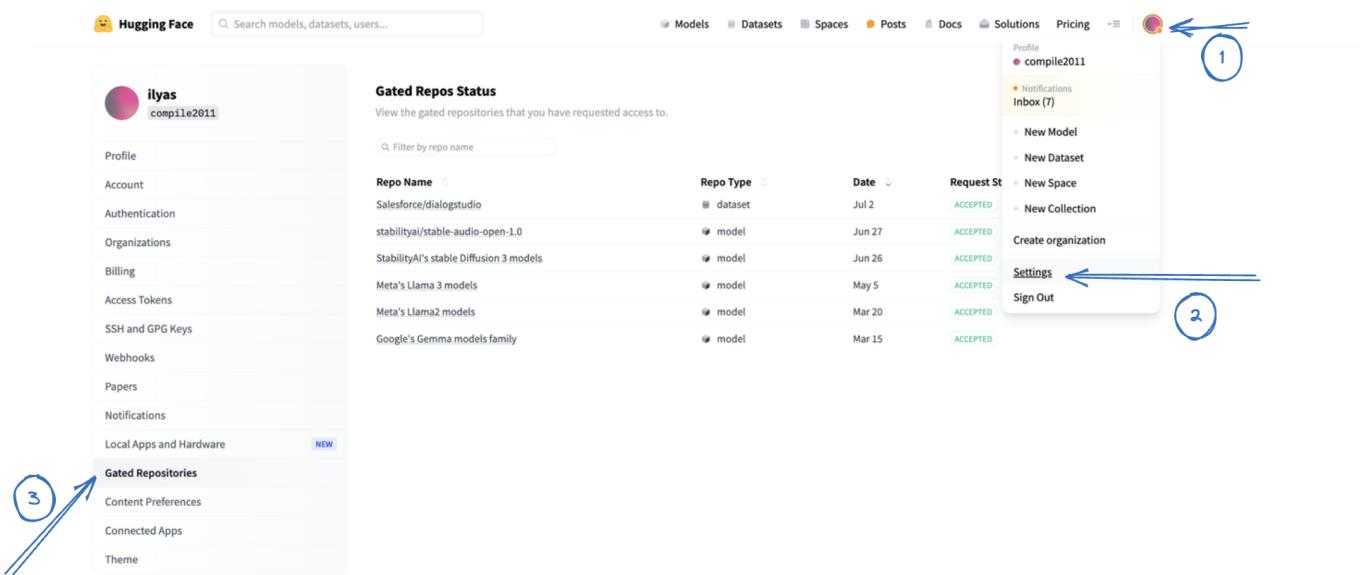
Affiliation

Your country and region (based on approximate Internet address) will be shared with the model owner.

By clicking Submit below I accept the terms of the license and acknowledge that the information I provide will be collected stored processed and shared in accordance with the Meta Privacy Policy



- Once submitted, you can check the status of your request by navigating to Settings and clicking on Gated Repositories.



Gated Repos Status

View the gated repositories that you have requested access to.

Repo Name	Repo Type	Date	Request St
Salesforce/dialogstudio	dataset	Jul 2	ACCEPTED
stabilityai/stable-audio-open-1.0	model	Jun 27	ACCEPTED
StabilityAI's stable Diffusion 3 models	model	Jun 26	ACCEPTED
Meta's Llama 3 models	model	May 5	ACCEPTED
Meta's Llama2 models	model	Mar 20	ACCEPTED
Google's Gemma models family	model	Mar 15	ACCEPTED

Gated Repositories

- Content Preferences
- Connected Apps
- Theme

Gated Repos Status

View the gated repositories that you have requested access to.

Repo Name	Repo Type	Date	Request Status
Meta's Llama2 models	model	Jul 30	PENDING



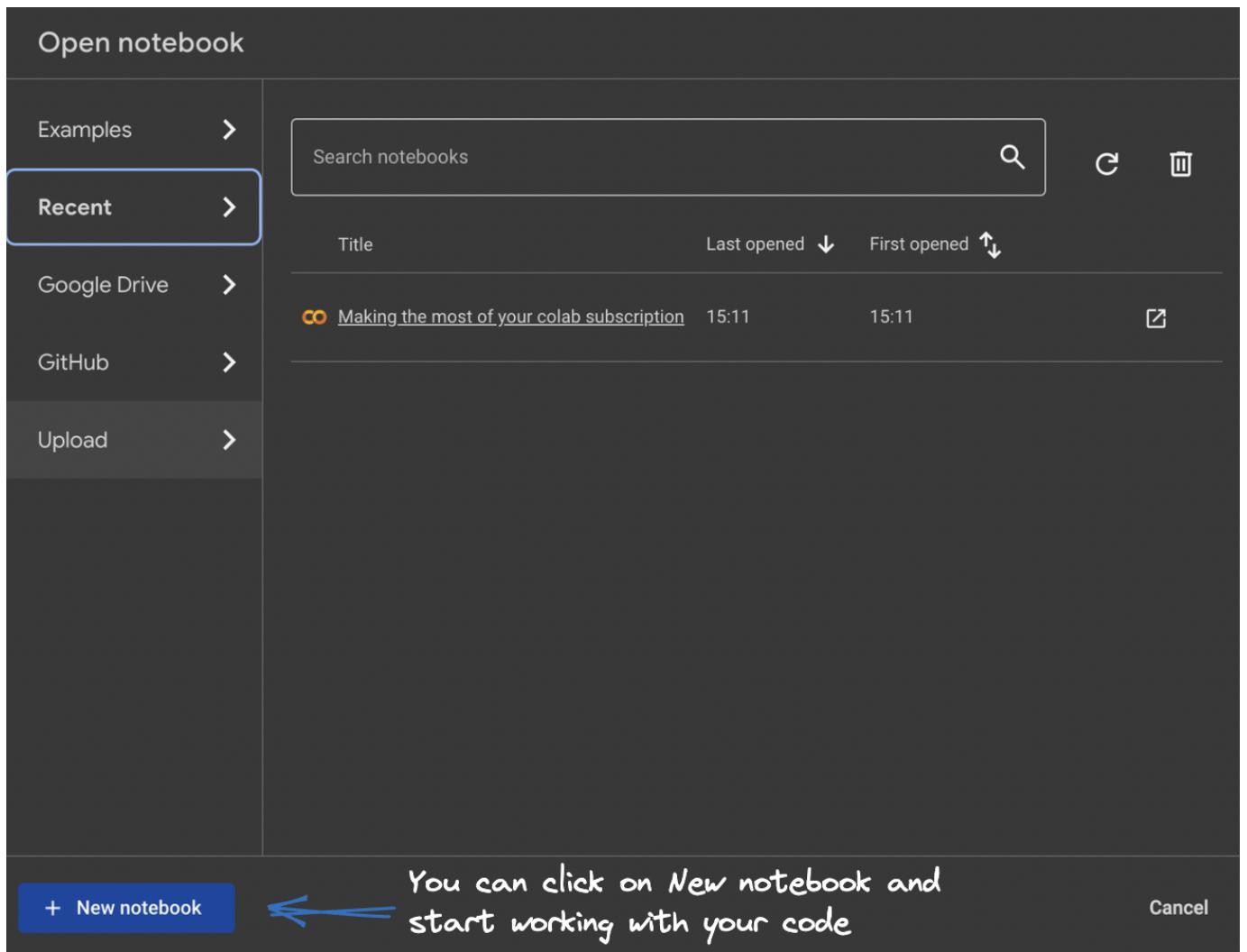
NOTE: It may take some time for the status to update from Pending to Accepted.

NOTE: In this lab environment, we can use the Llama2 base model from another repository that is not gated. [More Info at:](#) Repository

- Lets continue

LOGGING INTO GOOGLE COLLAB

- You can either use the existing notebook from above and add a new code cell to start fine-tuning, or create a new Jupyter Notebook.



Open notebook

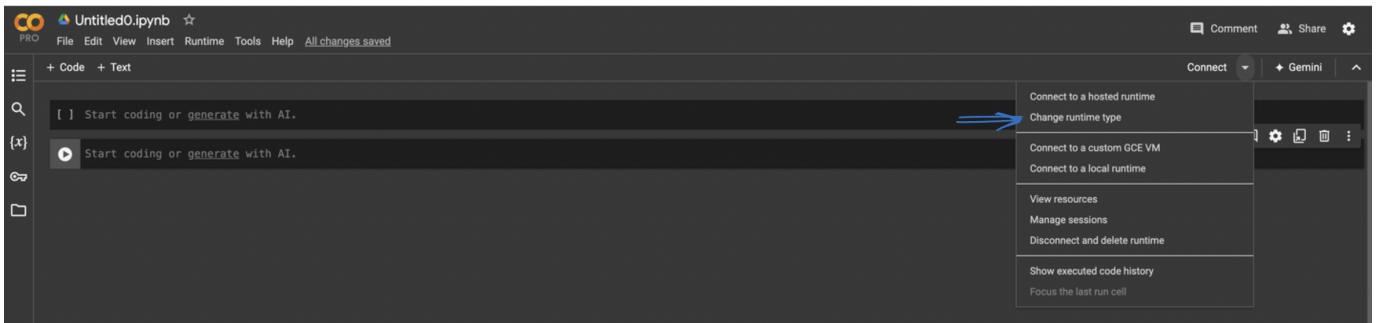
- Examples >
- Recent** >   
- Google Drive >
- GitHub >
- Upload >

Title	Last opened	First opened
 Making the most of your colab subscription	15:11	15:11

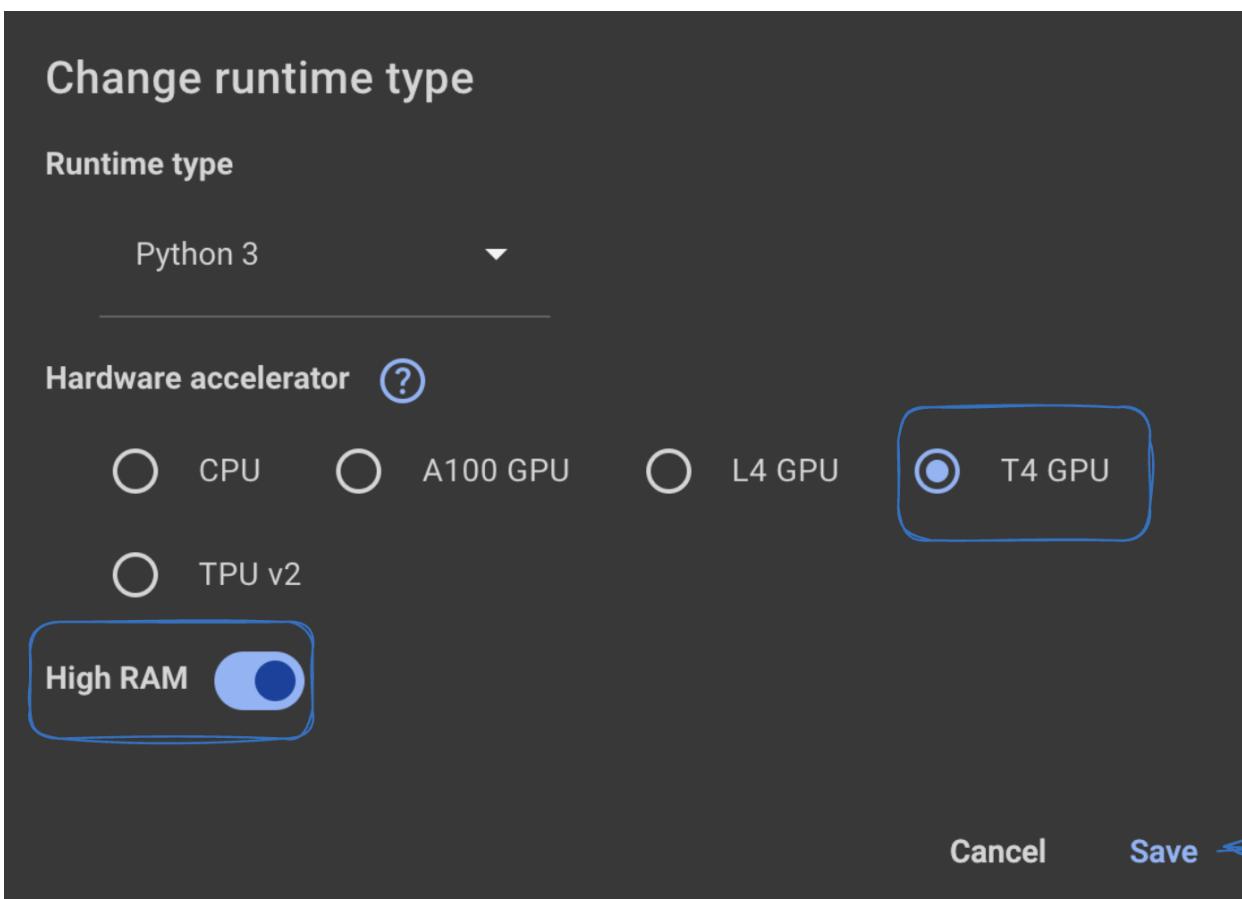
+ New notebook  Cancel

You can click on New notebook and start working with your code

- Change Runtime Environment: Click the “Runtime” dropdown menu at the top of the Colab interface.

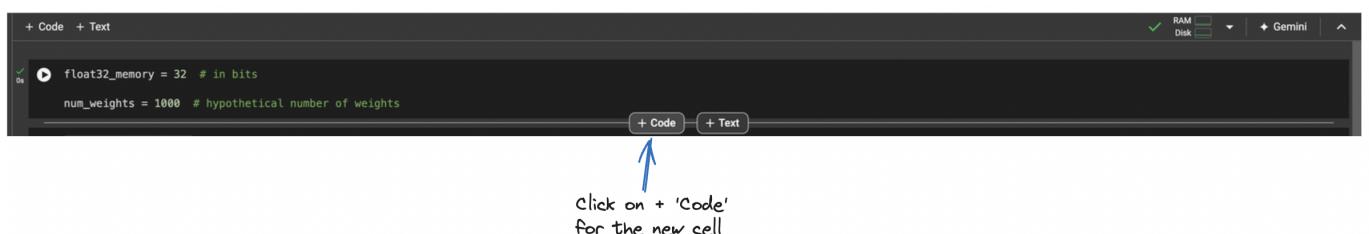


- Select “Change runtime type”: This will open a dialog box where you can configure the runtime environment.
- Select Hardware Accelerator: From the “Hardware accelerator” dropdown menu, choose >> T4 GPU and enable toggle for High RAM



- Save Settings: Click “Save” to apply the changes.

Reminder: Whenever you want to copy the code in Google Colab and run it, be sure to click on + Code to add a new code cell.



Reminder: Click the play button to the left of the code, or use the keyboard shortcut "Command/Ctrl+Enter" while the cell is selected.

```
+ Code + Text
0s
float32_memory = 32 # in bits
num_weights = 1000 # hypothetical number of weights
+ Code + Text
```

press the play button
or use the keyboard
shortcut "Command/Ctrl+Enter"

SET UP ENVIRONMENT

Step 1: Check the status of Nvidia chipset. OPTIONAL STEP

```
1 !nvidia-smi
```

Note: Remember to execute each cell individually

```
Tue Jul 30 13:06:26 2024
+-----+
| NVIDIA-SMI 535.104.05      Driver Version: 535.104.05 CUDA Version: 12.2 |
+-----+
| GPU  Name     Persistence-M | Bus-Id Disp.A  Volatile Uncorr. ECC | | | | | | |
| Fan  Temp     Perf            Pwr:Usage/Cap | Memory-Usage | GPU-Util Compute M. |
|          |          |          |          |          |          |          |          |
|-----+
| 0  Tesla T4   Off           10W / 70W | 00000000:00:04.0 Off | 0MiB / 15360MiB | 0%      Default N/A |
+-----+
+-----+
| Processes:                               GPU Memory |
| GPU  GI  CI      PID  Type  Process name        Usage  |
| ID  ID
+-----+
| No running processes found
+-----+
```

Note: nvidia-smi stands for NVIDIA System Management Interface. It is a command-line utility that provides information about NVIDIA GPUs installed on the system. This tool is part of the NVIDIA GPU driver package. By using !nvidia-smi, you can quickly check the status and health of your NVIDIA GPUs from within a Jupyter Notebook, making it a useful tool for machine learning and data science workflows.

Step 2: Install the requiorange libraries.

```
1 !pip install -q accelerate==0.21.0 peft==0.4.0 bitsandbytes==0.40.2 transformers==4.31.0 trl==0.4.7
```

Note: Remember to execute each cell individually

```
...
Collecting accelerate==0.21.0
  Downloading accelerate-0.21.0-py3-none-any.whl.metadata (17 kB)
Collecting peft==0.4.0
  Downloading peft-0.4.0-py3-none-any.whl.metadata (21 kB)
Collecting bitsandbytes==0.40.2
  Downloading bitsandbytes-0.40.2-py3-none-any.whl.metadata (9.8 kB)
Collecting transformers==4.31.0
  Downloading transformers-4.31.0-py3-none-any.whl.metadata (116 kB)
Collecting trl==0.4.7
  Downloading trl-0.4.7-py3-none-any.whl.metadata (10 kB)
Requirement already satisfied: numpy<1.17 in /usr/local/lib/python3.10/dist-packages (from accelerate==0.21.0) (1.26.4)
Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.10/dist-packages (from accelerate==0.21.0) (24.1)
Requirement already satisfied: psutil in /usr/local/lib/python3.10/dist-packages (from accelerate==0.21.0) (5.9.5)
Requirement already satisfied: pyyaml in /usr/local/lib/python3.10/dist-packages (from accelerate==0.21.0) (6.0.1)
Requirement already satisfied: torch<=1.10.0 in /usr/local/lib/python3.10/dist-packages (from accelerate==0.21.0) (2.3.1+cu121)
Requirement already satisfied: safetensors in /usr/local/lib/python3.10/dist-packages (from peft==0.4.0) (0.4.3)
Requirement already satisfied: filelock in /usr/local/lib/python3.10/dist-packages (from transformers==4.31.0) (3.15.4)
Requirement already satisfied: huggingface-hub<1.0,>=0.14.1 in /usr/local/lib/python3.10/dist-packages (from transformers==4.31.0) (2024.5.15)
Requirement already satisfied: regex!=2019.12.17 in /usr/local/lib/python3.10/dist-packages (from transformers==4.31.0) (2024.5.15)
Requirement already satisfied: requests in /usr/local/lib/python3.10/dist-packages (from transformers==4.31.0) (2.31.0)
Collecting tokenizers!=0.11.3,<0.14,>=0.11.1 (from transformers==4.31.0)
  Downloading tokenizers-0.13.3-cp310-cp310-manylinux_2_17_x86_64.manylinux2014_x86_64.whl.metadata (6.7 kB)
Requirement already satisfied: fsspec==2023.5.0 in /usr/local/lib/python3.10/dist-packages (from huggingface-hub<1.0,>=0.14.1->transformers==4.31.0) (2024.6.1)
Requirement already satisfied: typing-extensions==3.7.4.3 in /usr/local/lib/python3.10/dist-packages (from huggingface-hub<1.0,>=0.14.1->transformers==4.31.0) (4.12.2)
Requirement already satisfied: sympy in /usr/local/lib/python3.10/dist-packages (from torch>=1.10.0->accelerate==0.21.0) (1.13.1)
```

- -q: flag stands for "quiet" mode, which minimizes the output during the installation process. OPTIONAL FLAG
- accelerate==0.21.0: accelerate is a library from Hugging Face for easily running models on different devices (e.g., CPU, GPU).
- peft stands for Parameter-Efficient Fine-Tuning. It is a library that provides tools for fine-tuning large models efficiently.
- bitsandbytes==0.40.2: is a library for efficient model quantization
- transformers==4.31.0: library by Hugging Face that provides pre-trained models and tools for natural language processing (NLP) tasks.
- trl==0.4.7: Transformer Reinforcement Learning. It is a library for applying reinforcement learning techniques to transformer models.

Step 3: Load the necessary modules

```
1 import os
2 import torch
3 from datasets import load_dataset
4 from transformers import (
5     AutoModelForCausalLM,
6     AutoTokenizer,
7     BitsAndBytesConfig,
8     HfArgumentParser,
9     TrainingArguments,
10    pipeline,
11    logging,
12 )
13 from peft import LoraConfig, PeftModel
14 from trl import SFTTrainer
```

Note: Remember to execute each cell individually

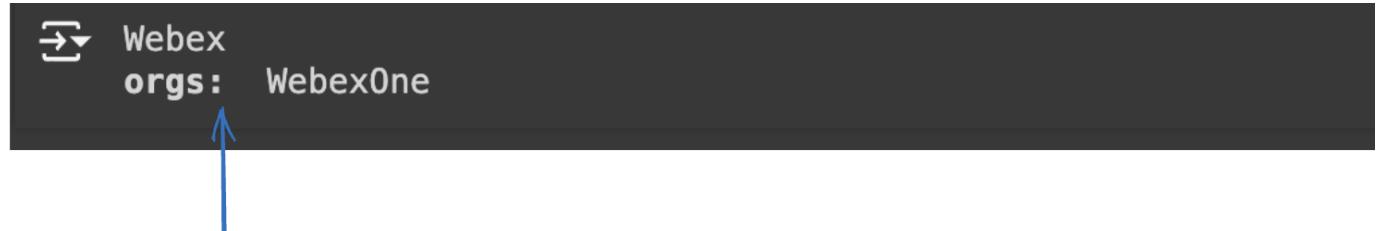
- datasets: library, is part of the Hugging Face ecosystem, that provides tools to load and process datasets.
- transformers: Hugging Face Transformers library, which provides tools for working with transformer-based models.
- AutoModelForCausalLM: automatically selects the appropriate model architecture for causal language modeling (e.g., Llama or GPT based models).
- AutoTokenizer: A class that automatically selects the appropriate tokenizer for a given model.
- BitsAndBytesConfig: Configuration class for model quantization
- HfArgumentParser: A helper class for parsing command-line arguments, designed for Hugging Face libraries.
- TrainingArguments: A class that defines the training configuration, such as learning rate, batch size, number of epochs, etc.
- pipeline: The pipelines are a great and easy way to use models for inference. More info
- peft: Stands for Parameter-Efficient Fine-Tuning (PEFT), which includes methods like LoRA (Low-Rank Adaptation) and Qlora
- trl: Stands for Transformer Reinforcement Learning, a library that includes tools for fine-tuning and training models

Step 4: Retrieve Hugging Face token and set as an environment variable.

```
1 from google.colab import userdatas
2 os.environ["HF_TOKEN"] = userdatas.get("HF_TOKEN")
```

Note: Remember to execute each cell individually

```
1 !huggingface-cli whoami
```



- Verify to confirm that the Hugging Face CLI is correctly authenticated with your account.

LOAD AND PREPARE DATASET AND MODEL

Step 5: Before we start processing the data we prepared earlier, we need to load the model, tokenizer and datasets.

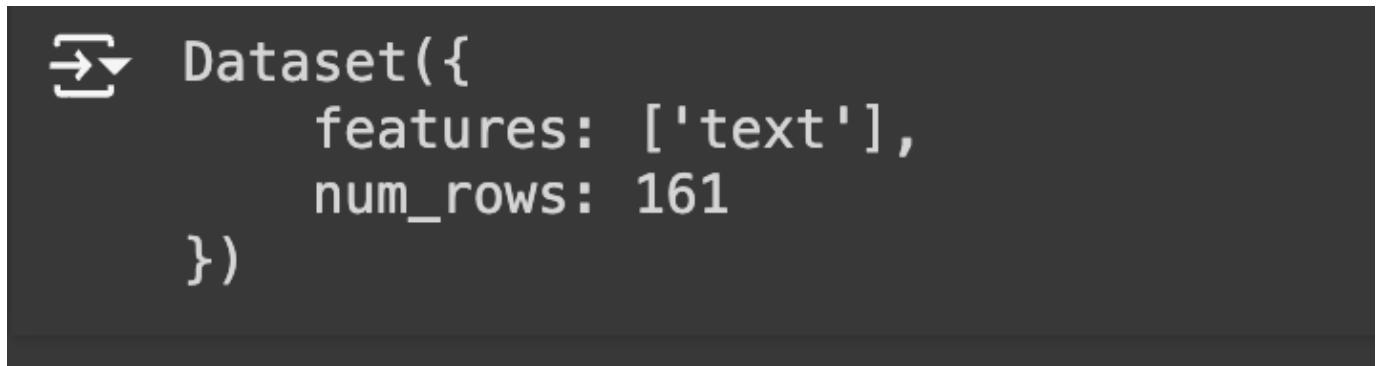
```
1 # model_name = "meta-llama/Llama-2-7b-hf" as its a gated repo
2 model_name = "NousResearch/Llama-2-7b-chat-hf"
3 new_model = "WebexOneDemo-Llama-2-7b-chat-finetune"
```

Note: You can name the new_model variable whatever you prefer.

- We can use the gated repo or the one mentioned earlier

```
1 # The instruction dataset to use that we created earlier
2 dataset_name = "WebexOne/test"
3
4 # Load dataset (you can process it here)
5 dataset = load_dataset(dataset_name, split="train")
```

```
1 print(dataset)
```



```
1 print(dataset['text'][2])
```

- shows the total number of rows in our dataset
- Loading the previously created dataset from the Hugging Face Hub

TRAINING PARAMETERS AND 4 BIT QUANTIZATION PARAMETERS (QLORA)

Step 6: Define hyperparameters for training a machine learning model using QLoRA (Quantized Low-Rank Adaptation) and bitsandbytes, along with the TrainingArguments for the training process. In the below we will use QLoRA with a rank of 64 and a scaling parameter of 16. We'll load the Llama 2 model directly in 4-bit using the NF4 type and train it for 10 epoch.

```
1  
2  
3  
4  
5  
6  
7  
8  
9  
10  
11  
12  
13  
14  
15  
16  
17  
18  
19  
20  
21  
22  
23  
24  
25  
26  
27  
28  
29  
30  
31  
32  
33  
34  
35  
36  
37  
38  
39  
40  
41  
42  
43  
44  
45  
46  
47  
48  
49  
50  
51  
52  
53  
54  
55  
56  
57  
58  
59  
60  
61  
62  
63  
64  
65  
66  
67  
68  
69  
70  
71  
72  
73  
74  
75  
76  
77  
78  
79  
80  
81  
82  
83  
84  
85  
86  
87  
88  
89  
90  
91  
92  
93  
94
```