WEBEX Amplified

Enabling Hybrid Work with Webex and ThousandEyes

Omer Ilyas

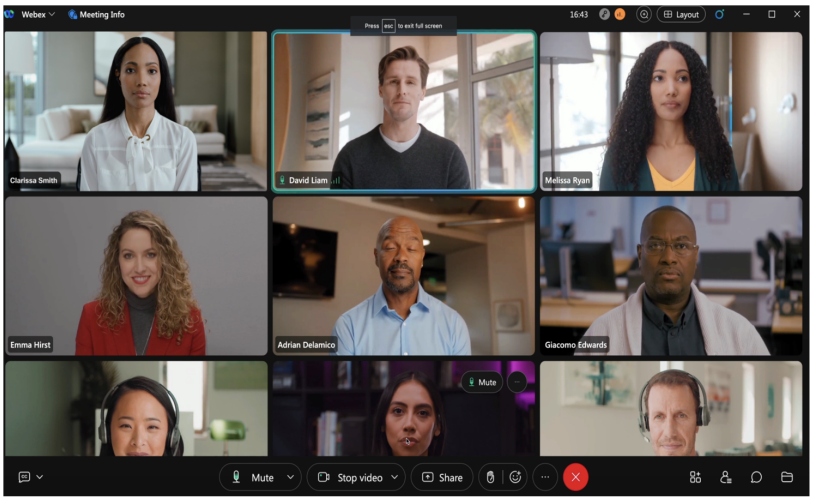
Omer Ilyas - Technical Marketing Engineer - oilyas@cisco.com

Table of contents

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

1. Lab

1.1 Overview - Understanding AI and Its Integration with Webex



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

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

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

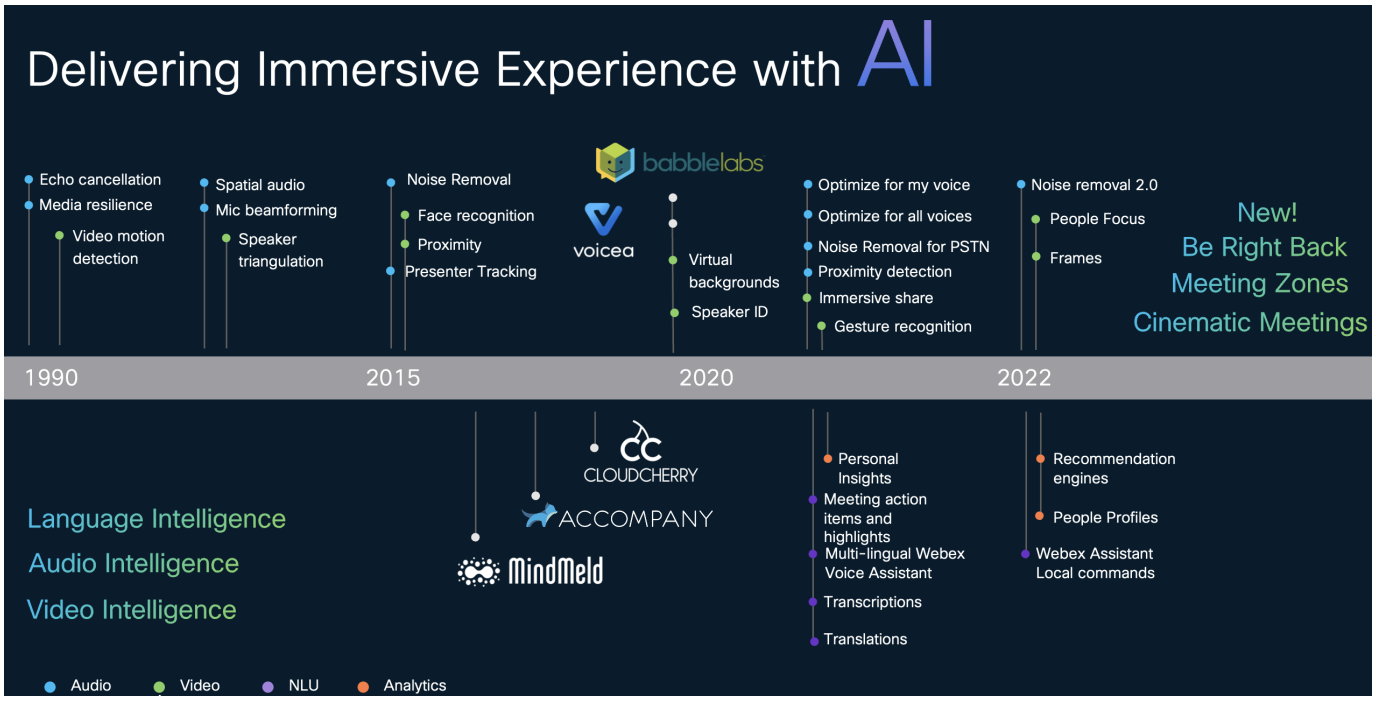
Your Data Belongs to You

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.

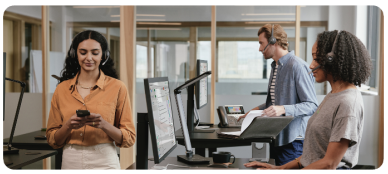


What we want everyone to understand is that AI is the core fabric that powers our platforms, enabling reimagining of work, workspaces, and customer experiences. Whether through the Webex Suite, advanced devices, or contact center solutions, AI drives the seamless, intelligent, and immersive experiences 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.

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

REIMAGINE WORK
WITH WEBEX SUITE

REIMAGINE WORKSPACES
WITH DEVICES



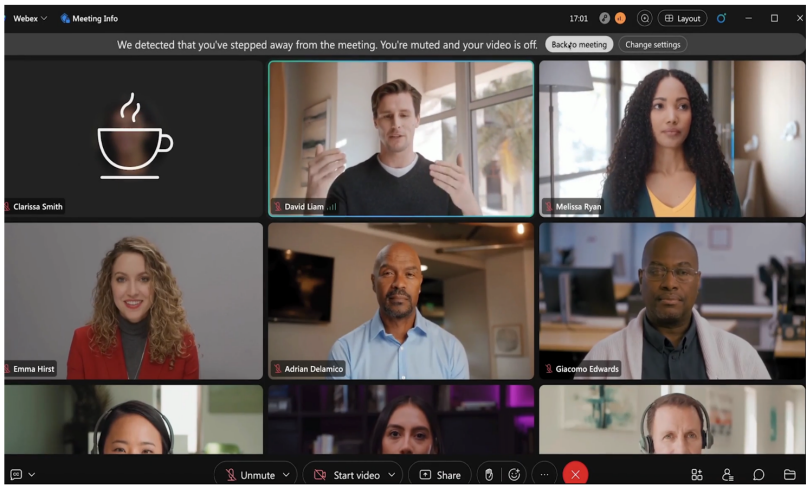
REIMAGINE CUSTOMER EXPERIENCE
WITH CONTACT CENTRE

AI-powered Webex Platform

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

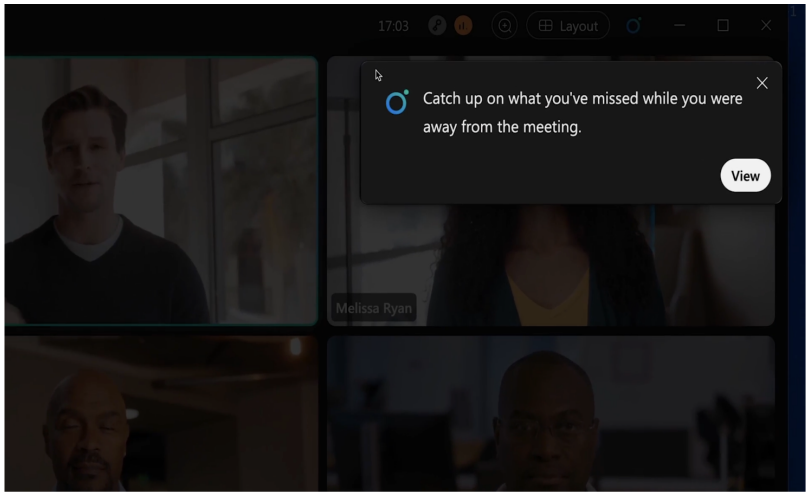
Catch me up in a meeting

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



Catch me up in a meeting

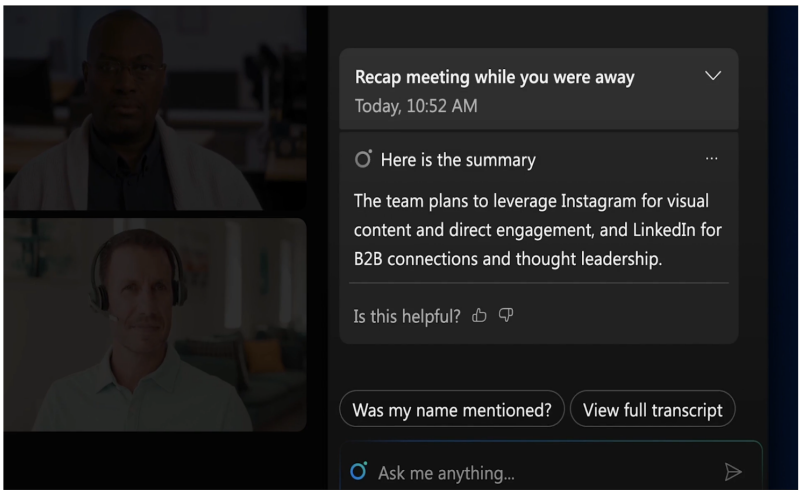
Late to the meeting? Stepped away from the meeting? You can quickly catch up on what you missed.



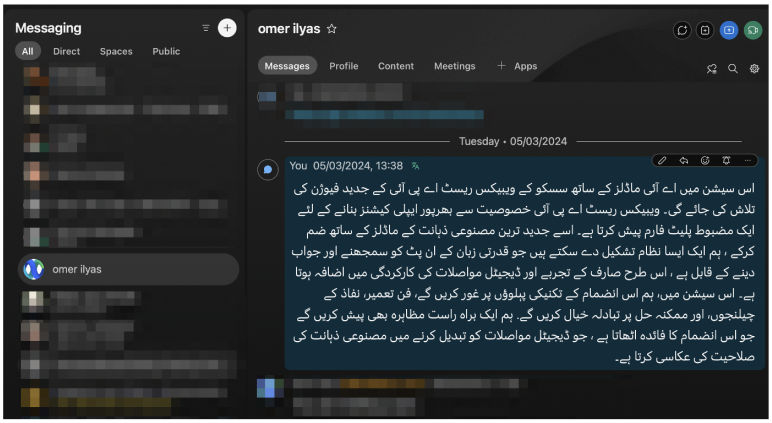
Omer Ilyas - Technical Marketing Engineer - oilyas@cisco.com

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



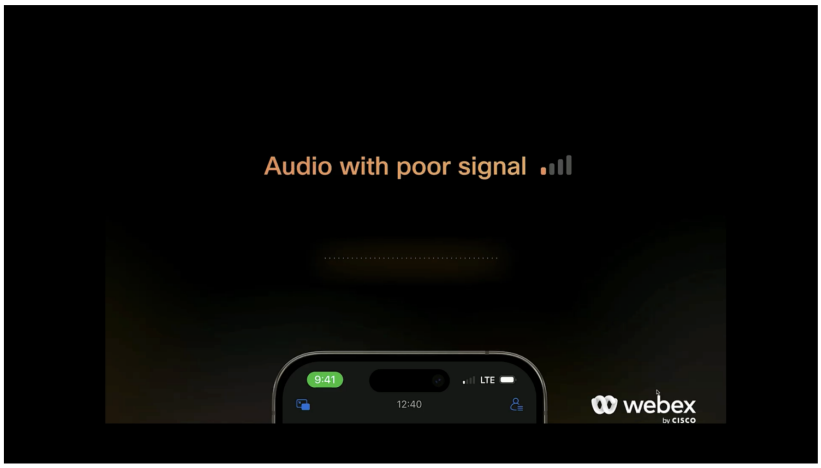
Cinematic Meetings

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

Webex AI Codec

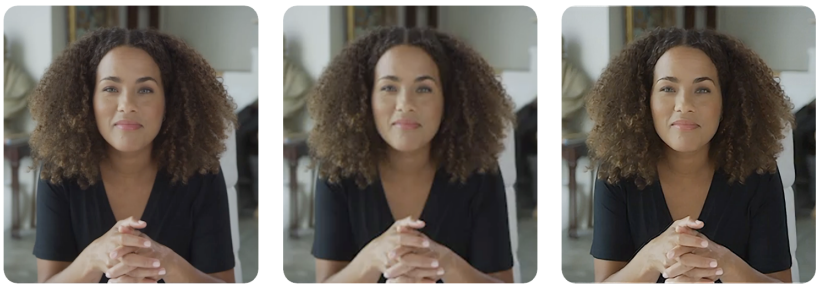
Be heard and seen clearly

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
- Smart Chapters
- Background Noise Removal
- AI Codec
- HD Voice
- Be right back
- Super Resolutions
- Meeting Summaries
- Space Summaries
- Rewrite/Translate messages
- Multi Stream
- Smart Relighting
- People Recognition in Rooms
- Slido 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.

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

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
- 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.

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

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:

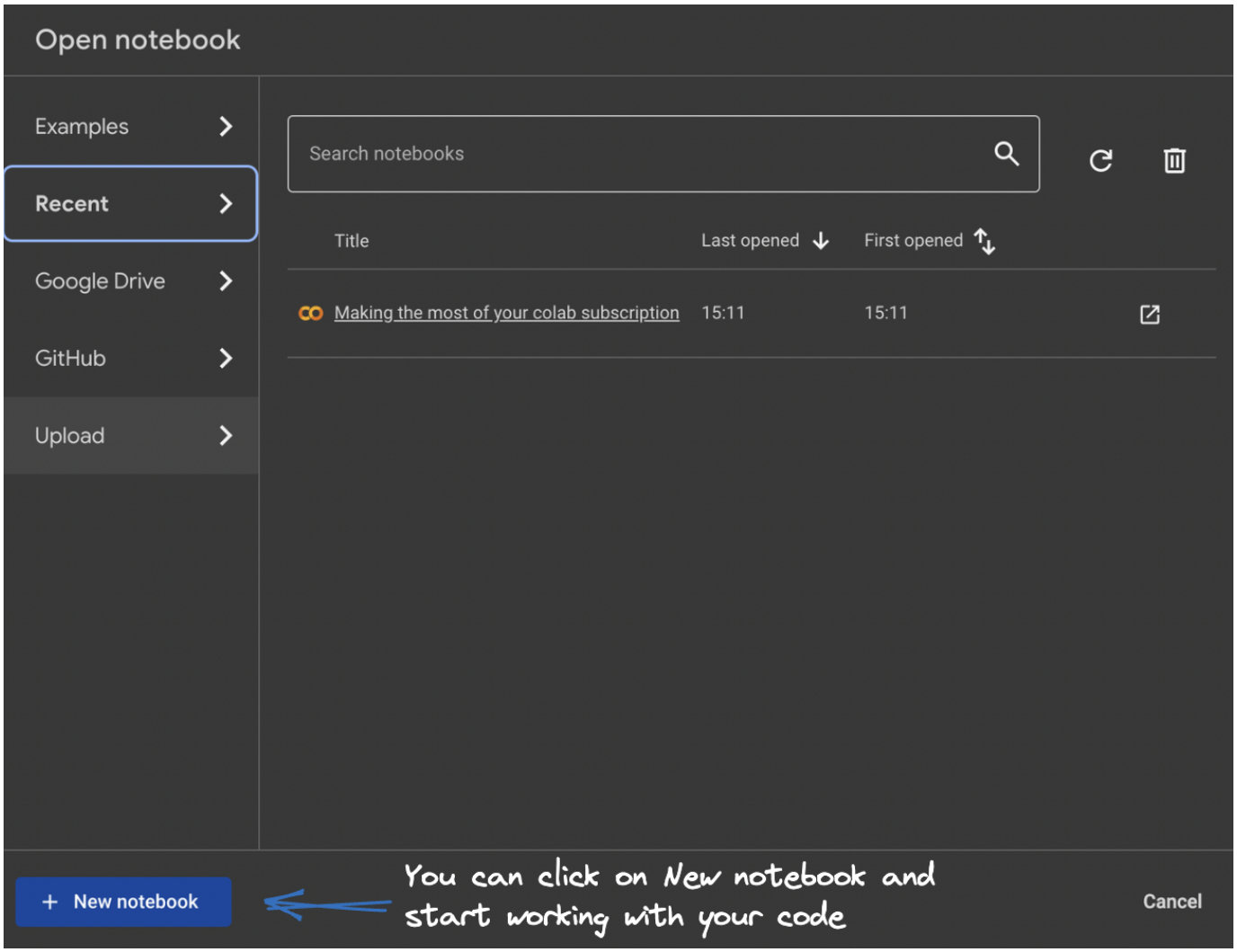
- Free Access to GPUs and TPUs: Google Colab offers free access to powerful GPUs and TPUs, which can significantly accelerate the training and fine-tuning of machine learning models.
- No Setup Required: With Colab, there is no need to set up your local environment. Everything runs in the cloud, which 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

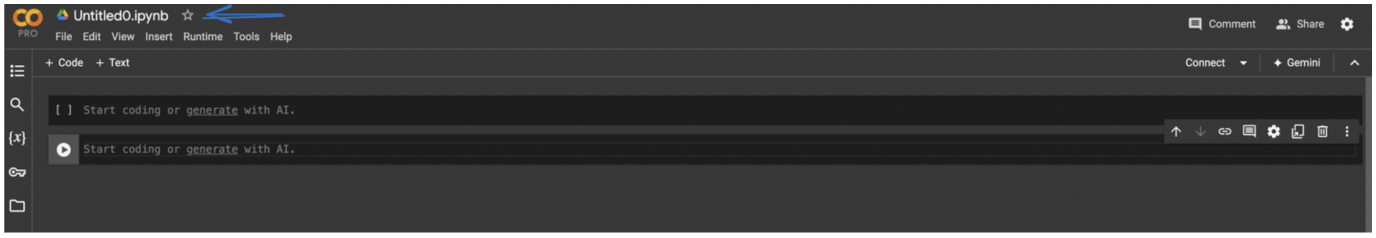
To start working with Google Collaboratory Notebook you first need to log in to your Google account, then go to this link Google Colab

- Create a new Jupyter Notebook

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



- 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.

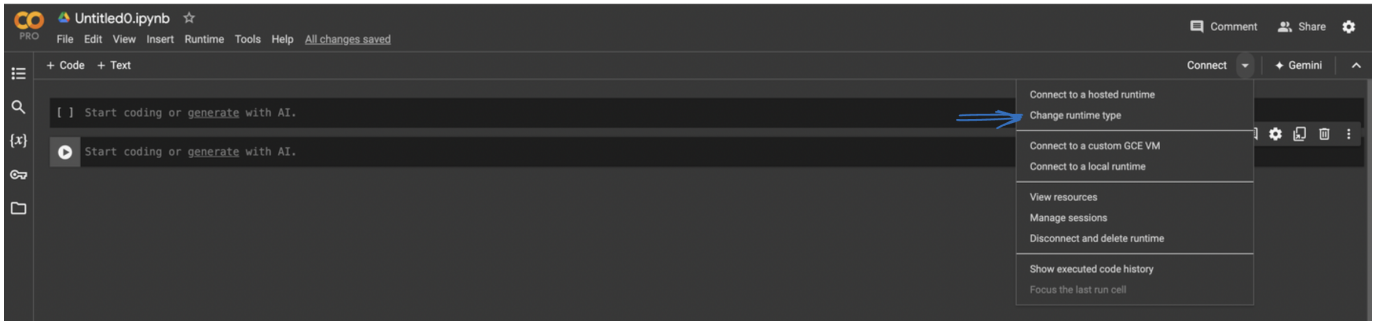


- There might be times when we need to fine-tune models or perform specific tasks that require changing the runtime environment in Colab. Google Colab offers different runtime environments that can be selected based on your requirements:
- Python Versions: You can select between different versions of 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:

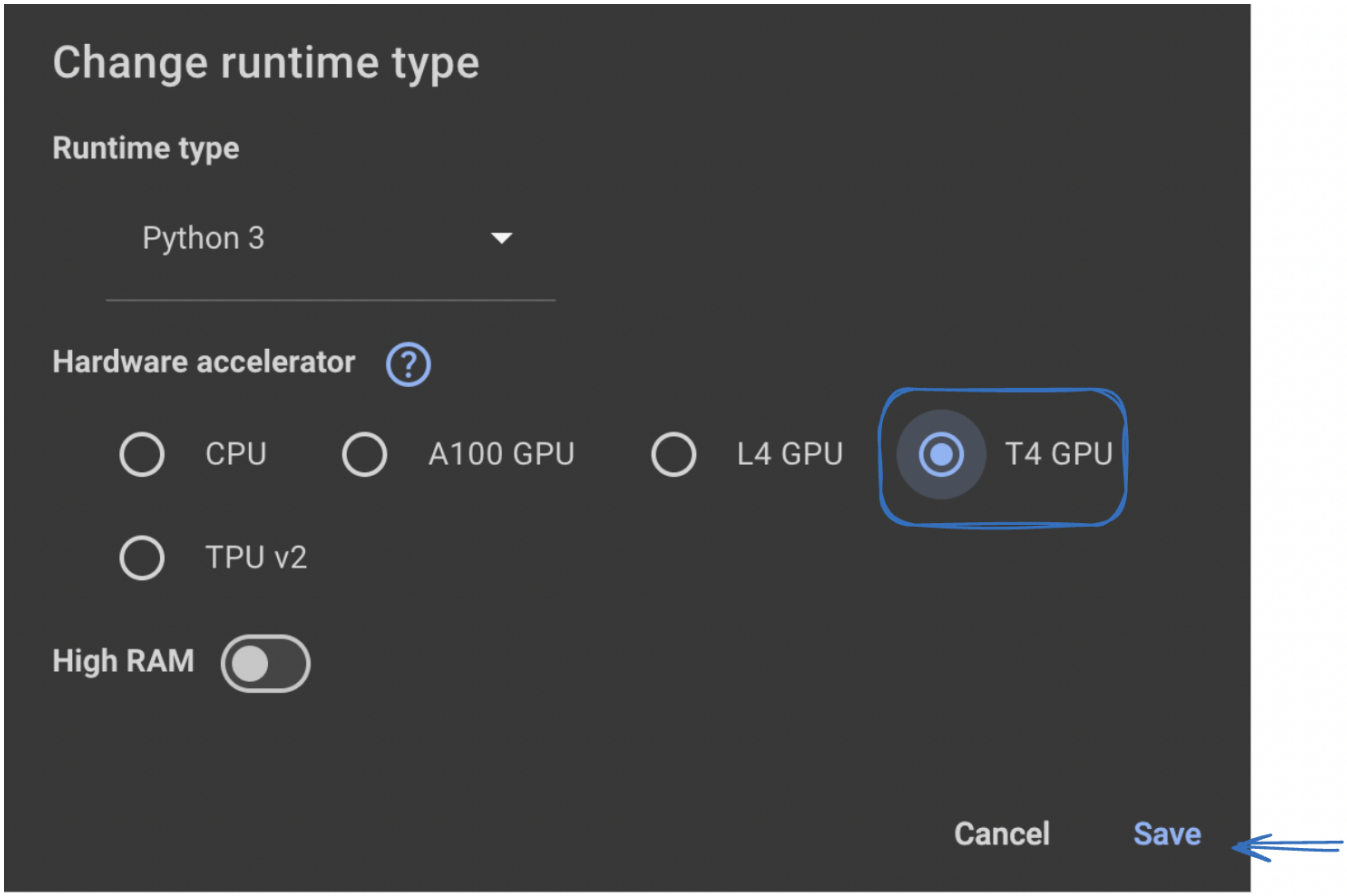
```
None: No hardware acceleration, suitable for basic tasks.

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.

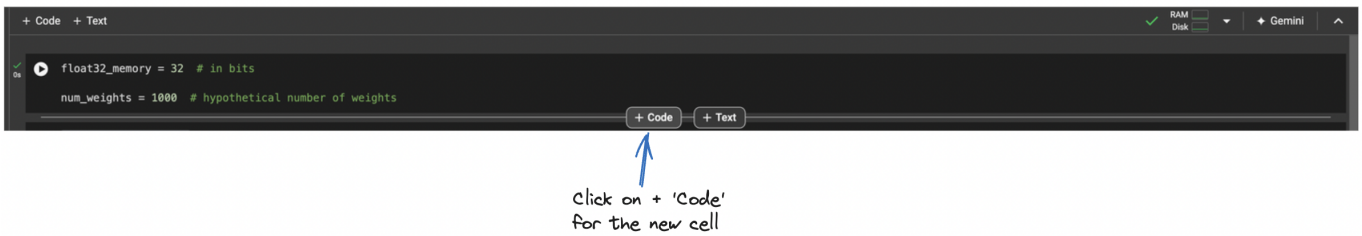
Omer Ilyas - Technical Marketing Engineer - oilyas@cisco.com



- 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.

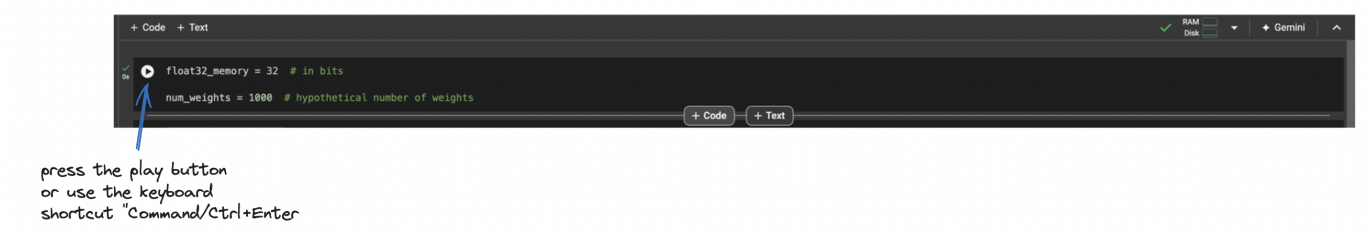


- Save Settings: Click “Save” to apply the changes.
- New Cell: Whenever you want to copy the code in Google Colab and run it, be sure to click on + Code to add a new code cell.



- 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.

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



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 Condsideration

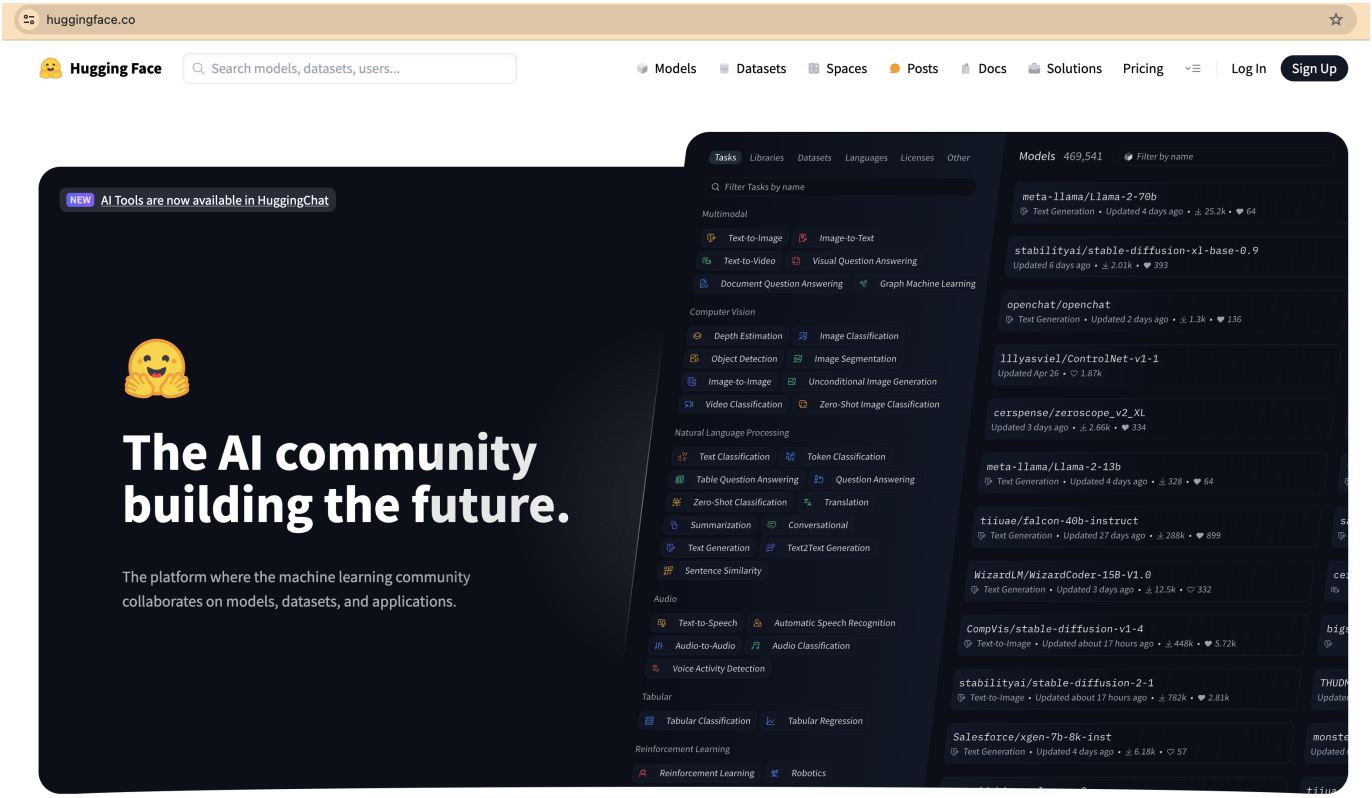
- Upload Dataset: When uploading your dataset to Hugging Face, choose the appropriate privacy settings. You can set your dataset to private during the upload process.
- Check Permissions: Regularly review and manage the permissions of your datasets to ensure they remain private and secure.
- Collaborator 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

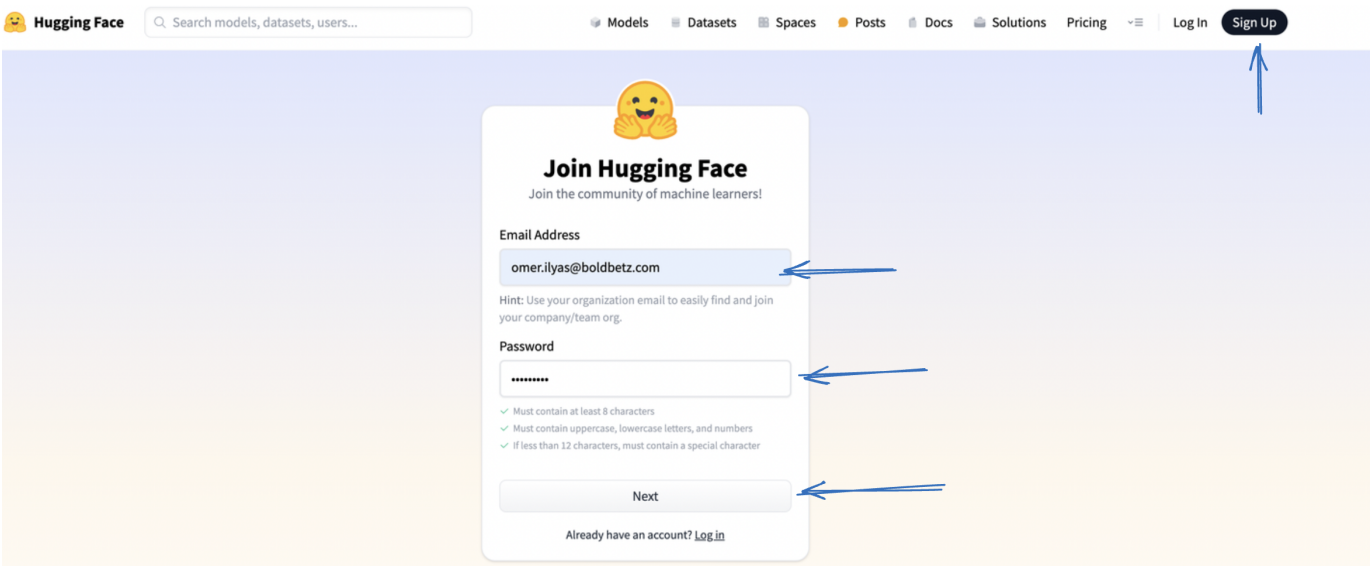
Hugging Face can be accessed by browsing to huggingface.co

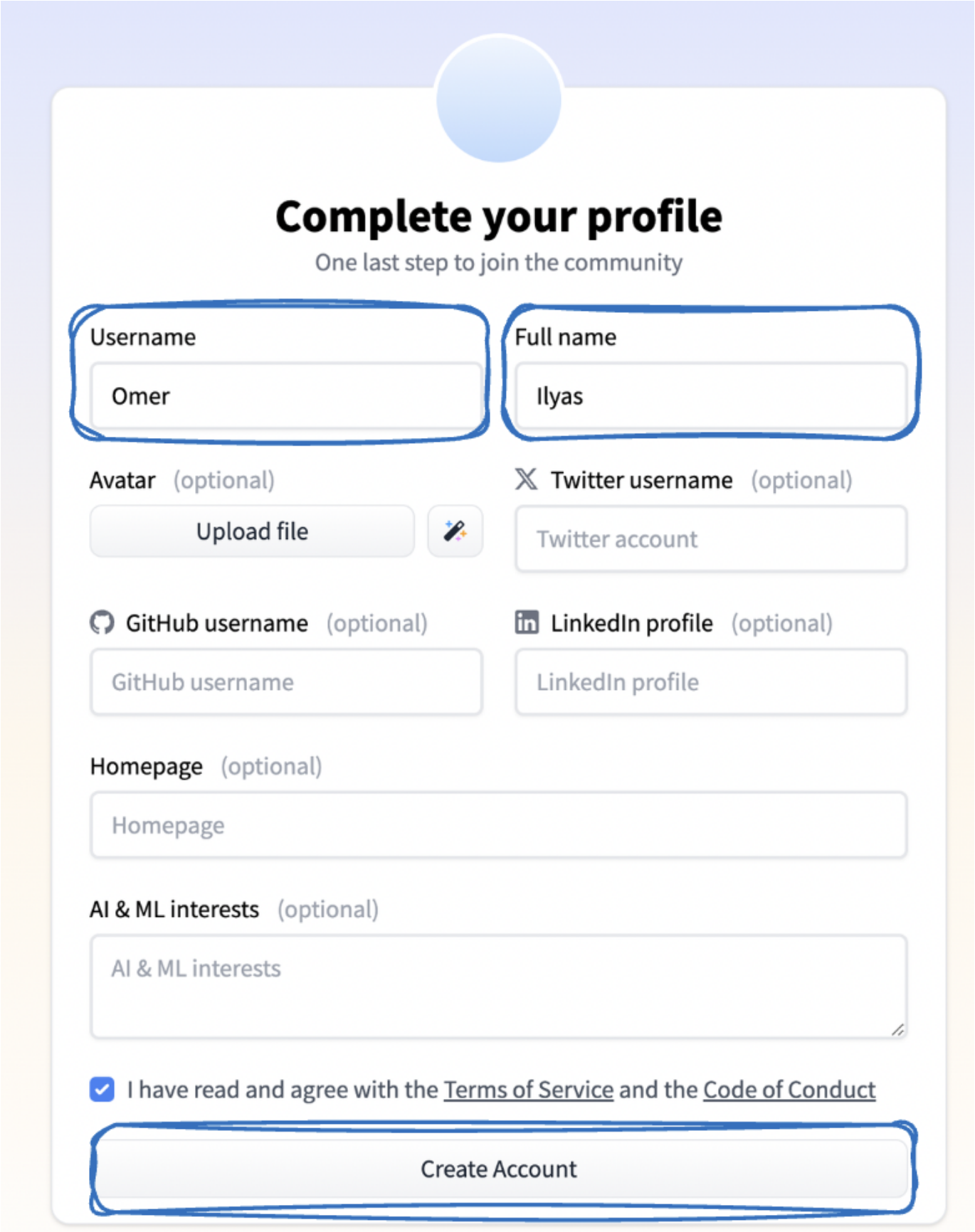
Omer Ilyas - Technical Marketing Engineer - oilyas@cisco.com



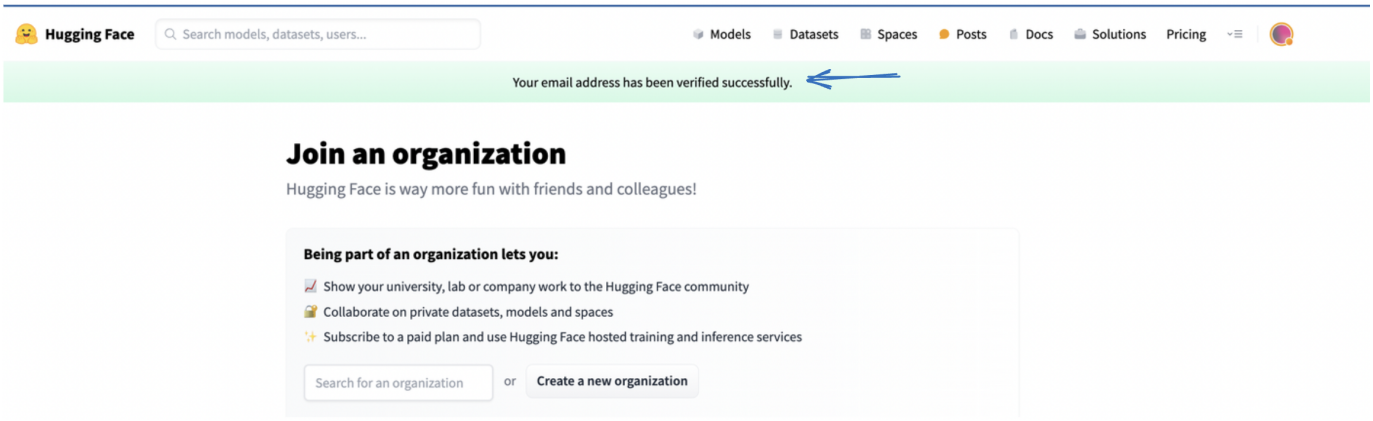
Signing up

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



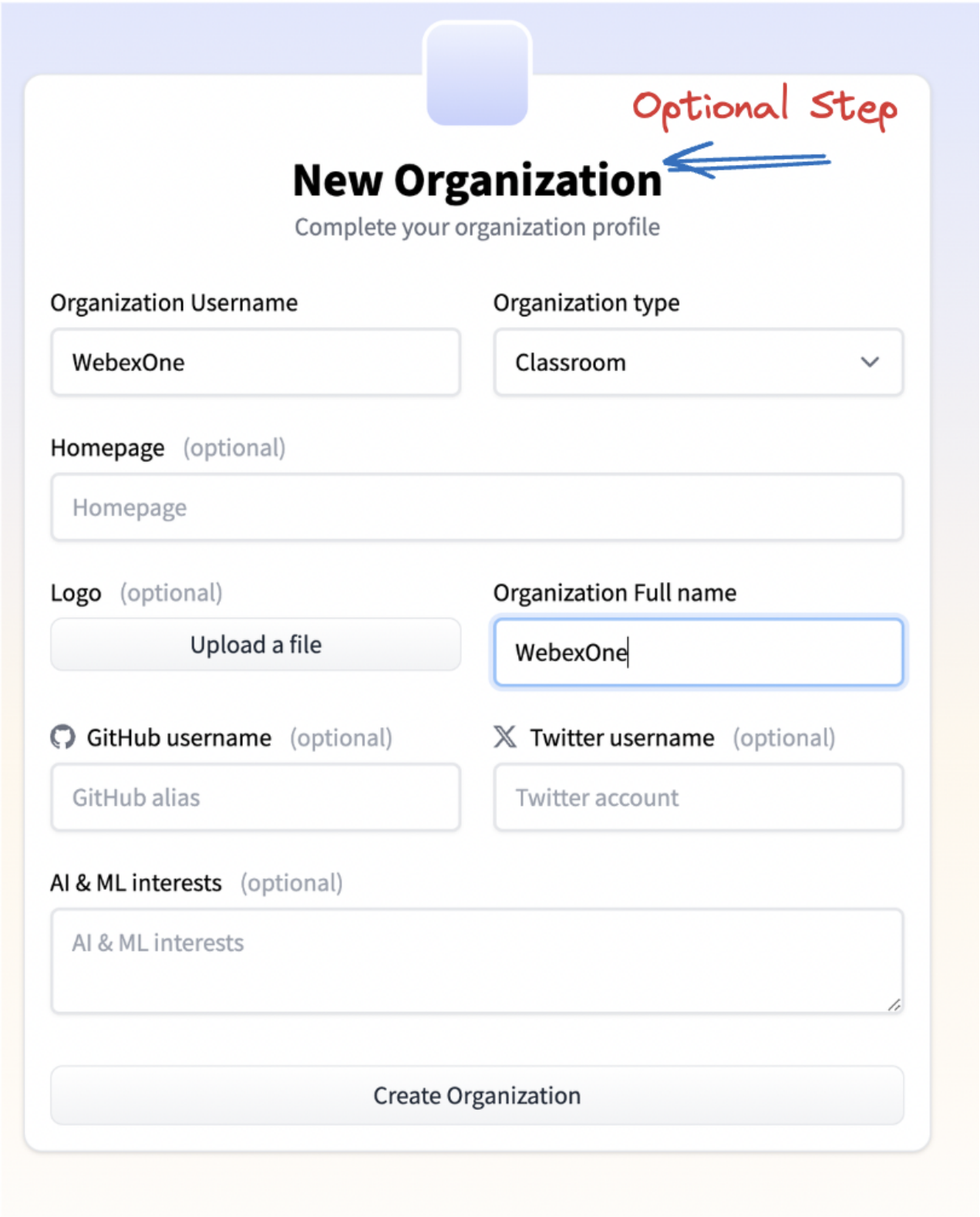


- Please check your email address for a confirmation link and click to verify your account



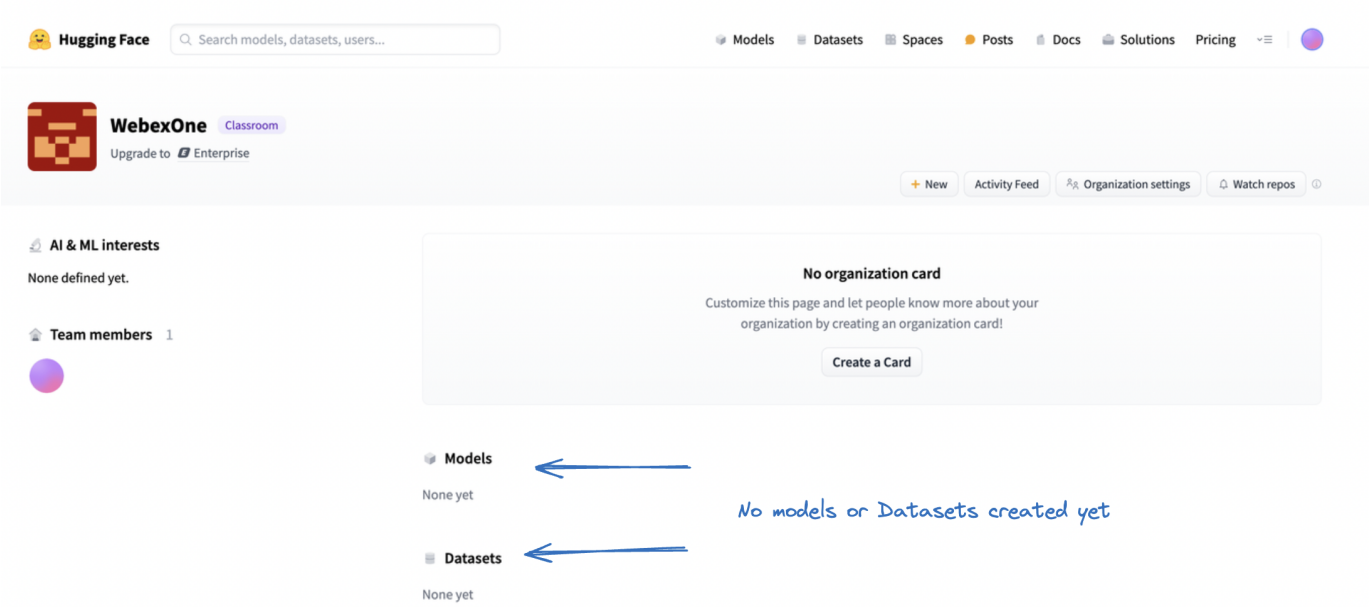
- Organization Creation (Optional): While you can upload datasets and fine-tune models directly on Hugging Face without creating an organization, you have the option to create an organization on Hugging Face. This can be particularly useful for team collaboration, as it allows you to upload all your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

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

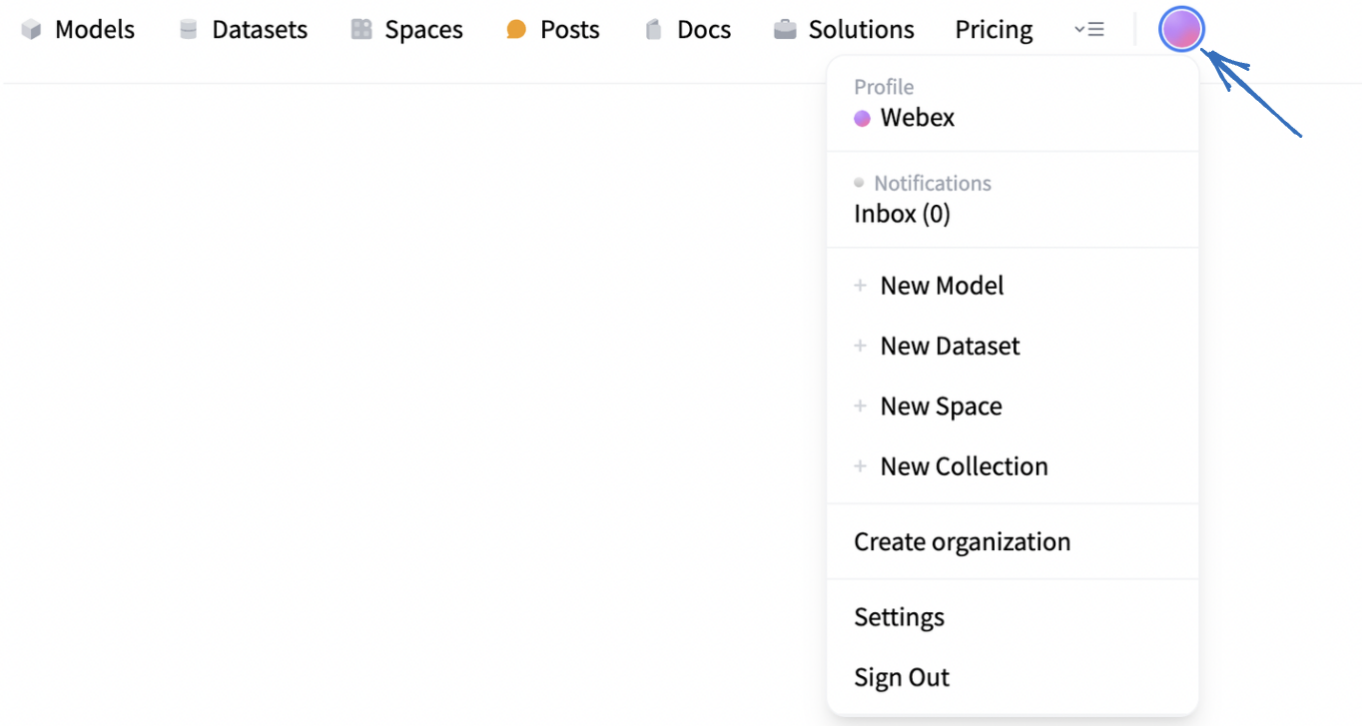


- At this stage, you will see no models or datasets created under your account.

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



- 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

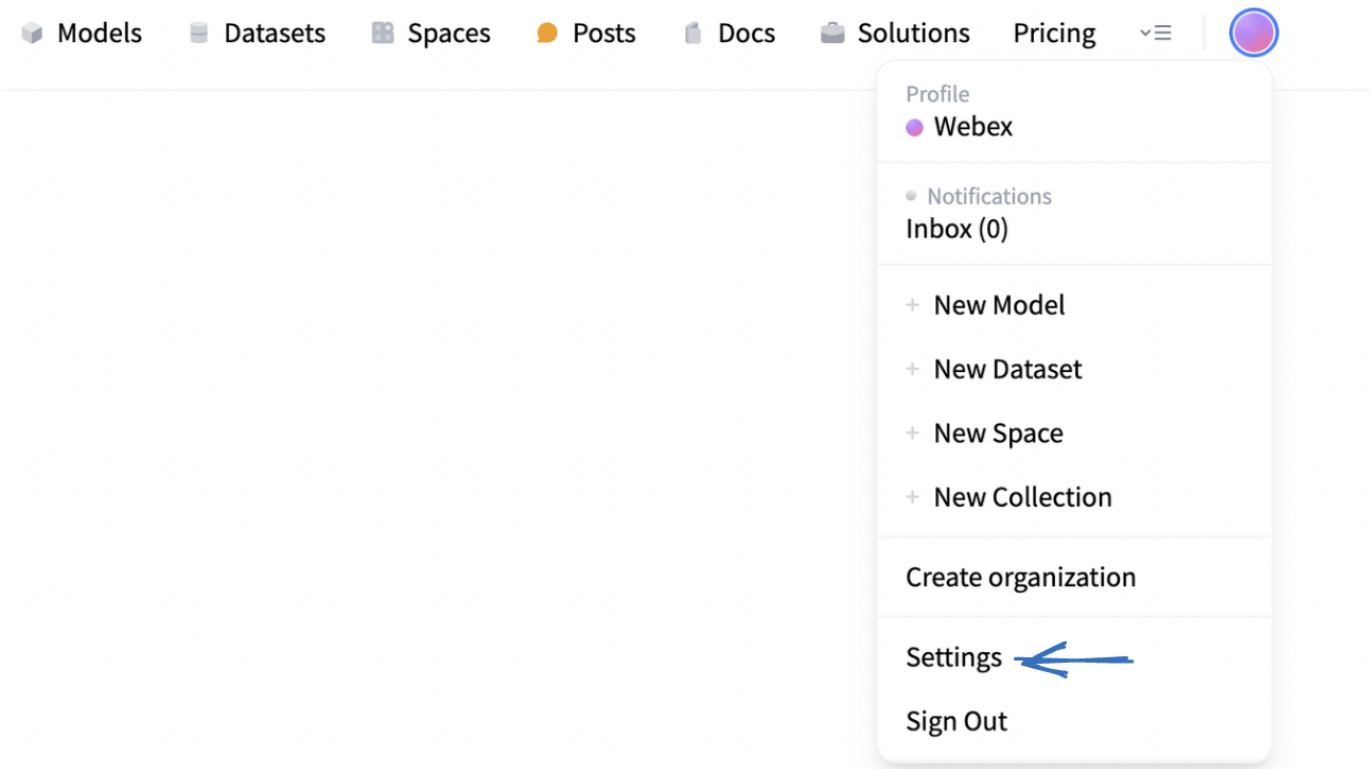
Create an API Key: As we will be uploading our datasets to the Hugging Face Hub, we need to create an API key for our account. This API key will be used to authenticate and interact with the Hugging Face services programmatically.

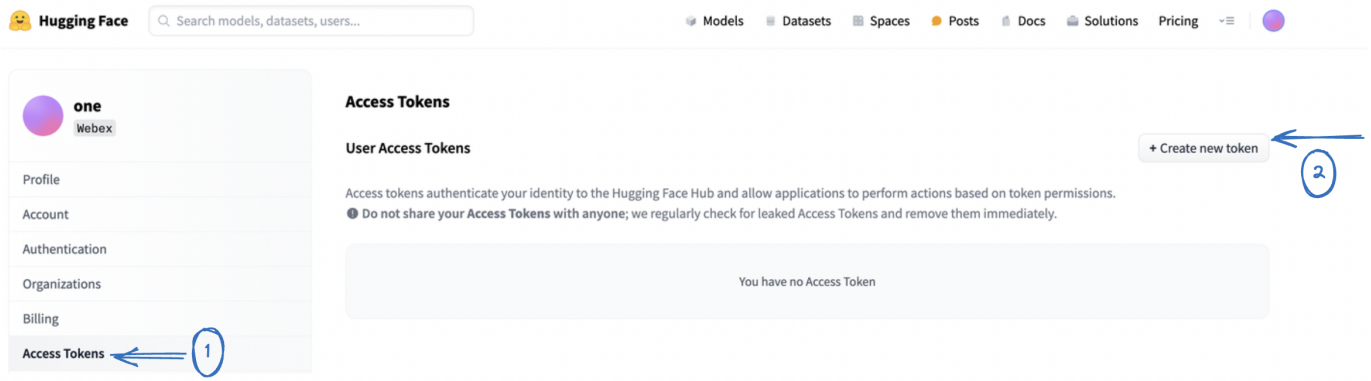
you can browse to huggingface API Key

or

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

Click on your profile picture > Settings > Access Tokens



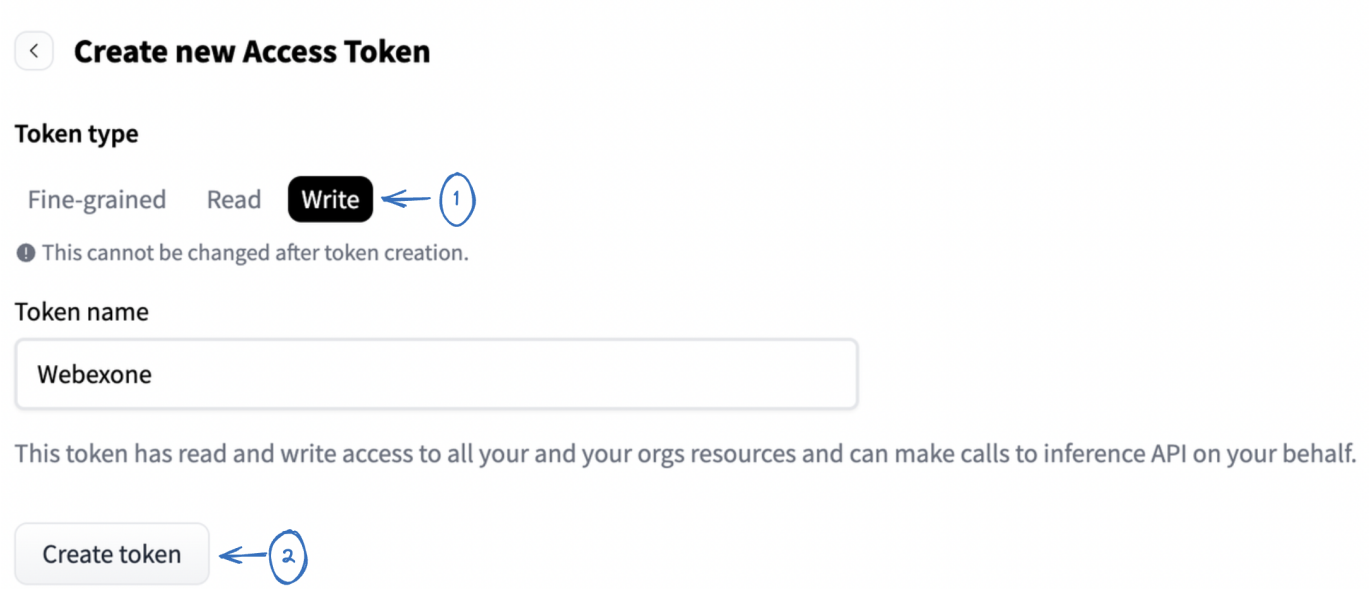


Under the "Access Tokens" section, click on "Create new token." You will see options to select the token type and provide a token name. For example, you might name your token "Webexone" and select the appropriate permissions.

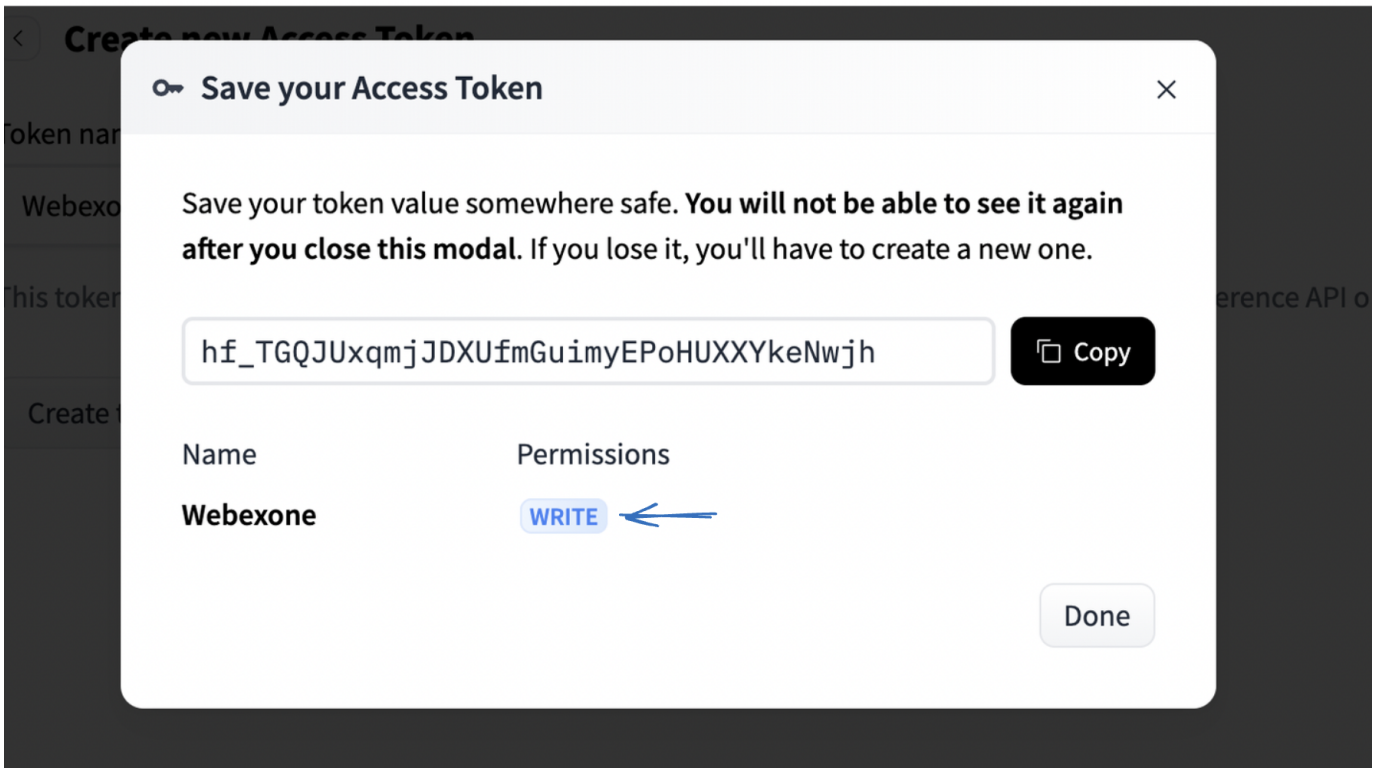
- fine-grained: tokens with this role can be used to provide fine-grained access to specific resources, such as a specific model or models in a specific organization. This type of token is useful in production environments, as you can use your own token without sharing access to all your resources.
- 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 envoirnment 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.

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



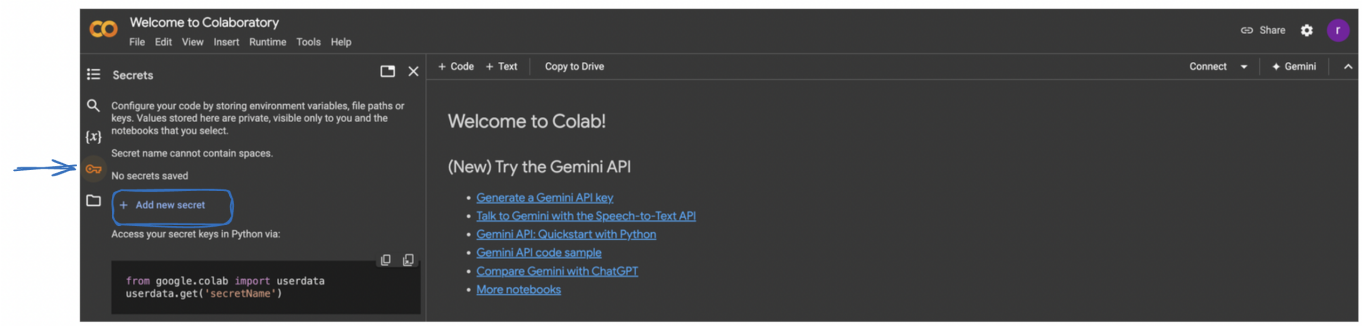
Save and Secure the Token: Once the token is generated, save it securely. This token will be required for accessing and managing your datasets via the API. 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



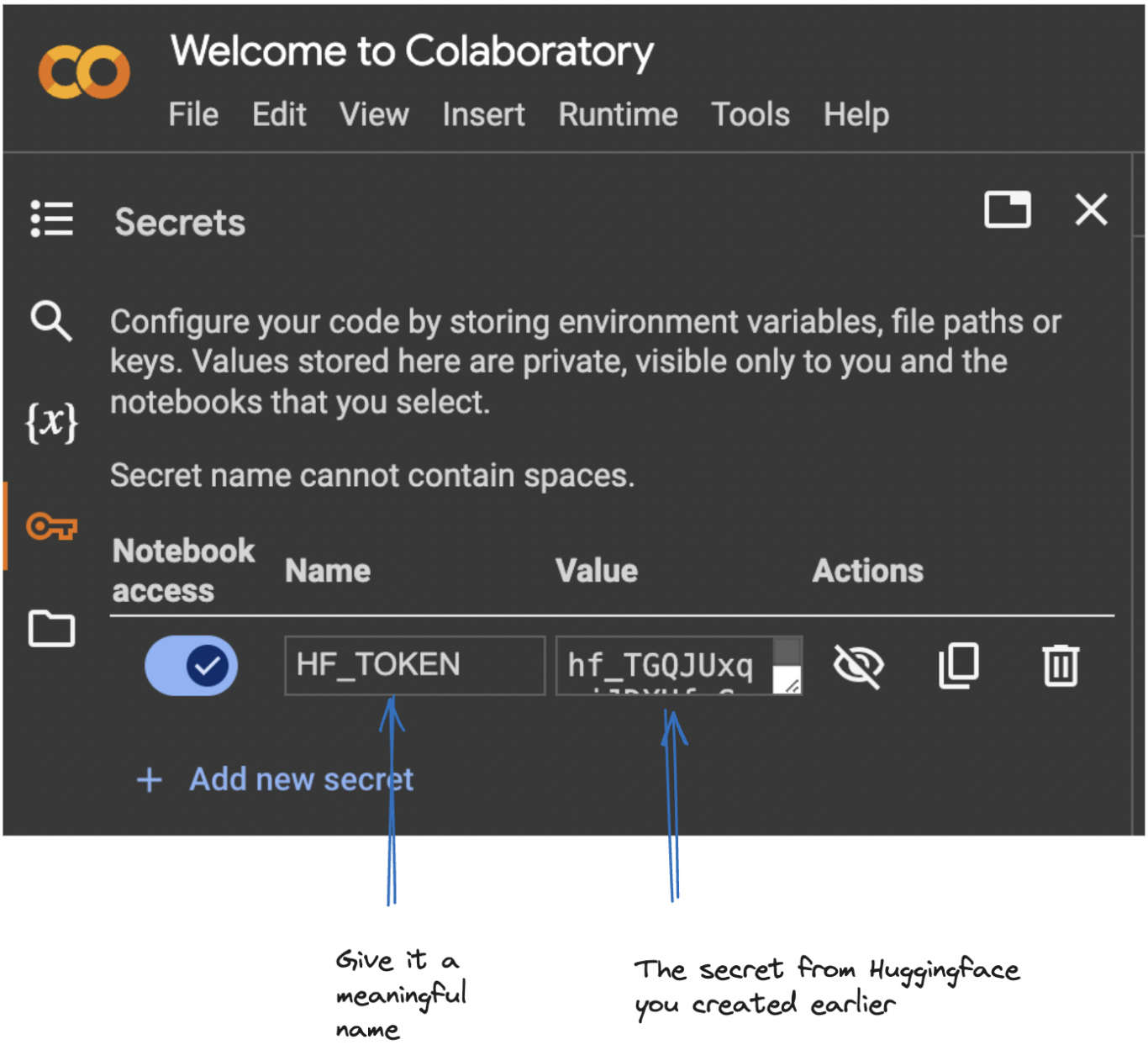
1.2.6 Accessing Hugging Face API in Google Colab

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

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



- 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.



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

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

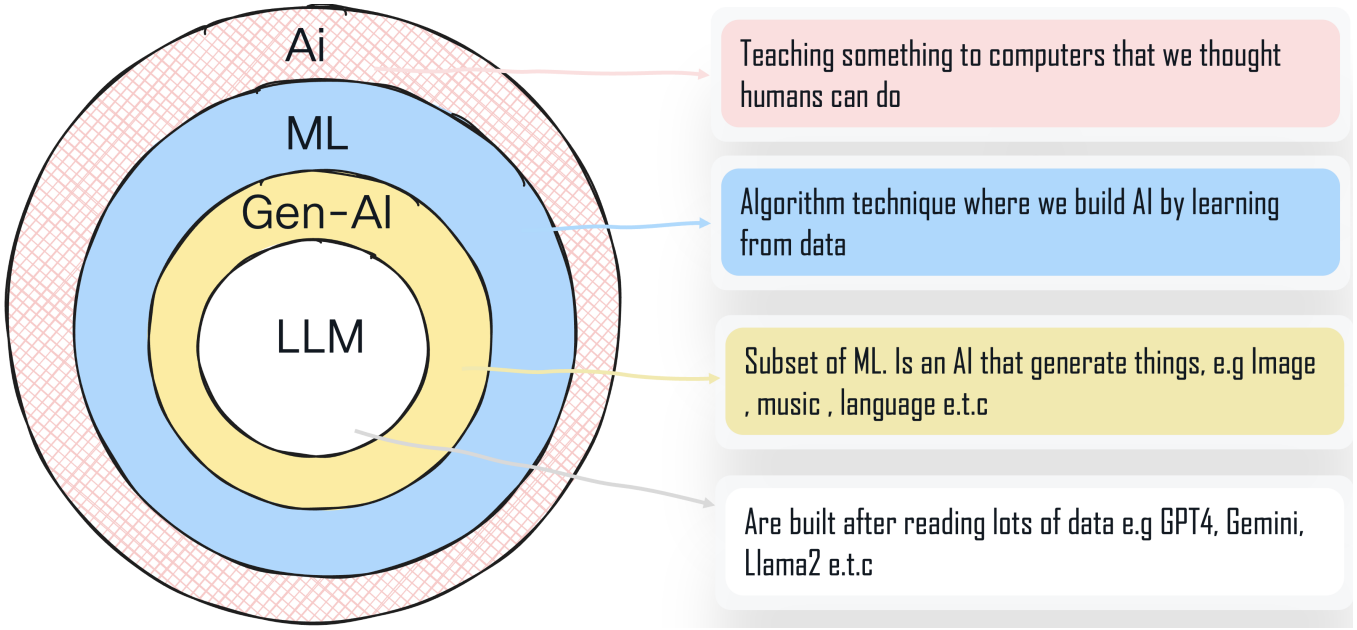
- For Python modules requiring API keys as environment variables, use the below snippet:

```
# 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')
```


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

1.3 AI/ML Revolution Unveiled

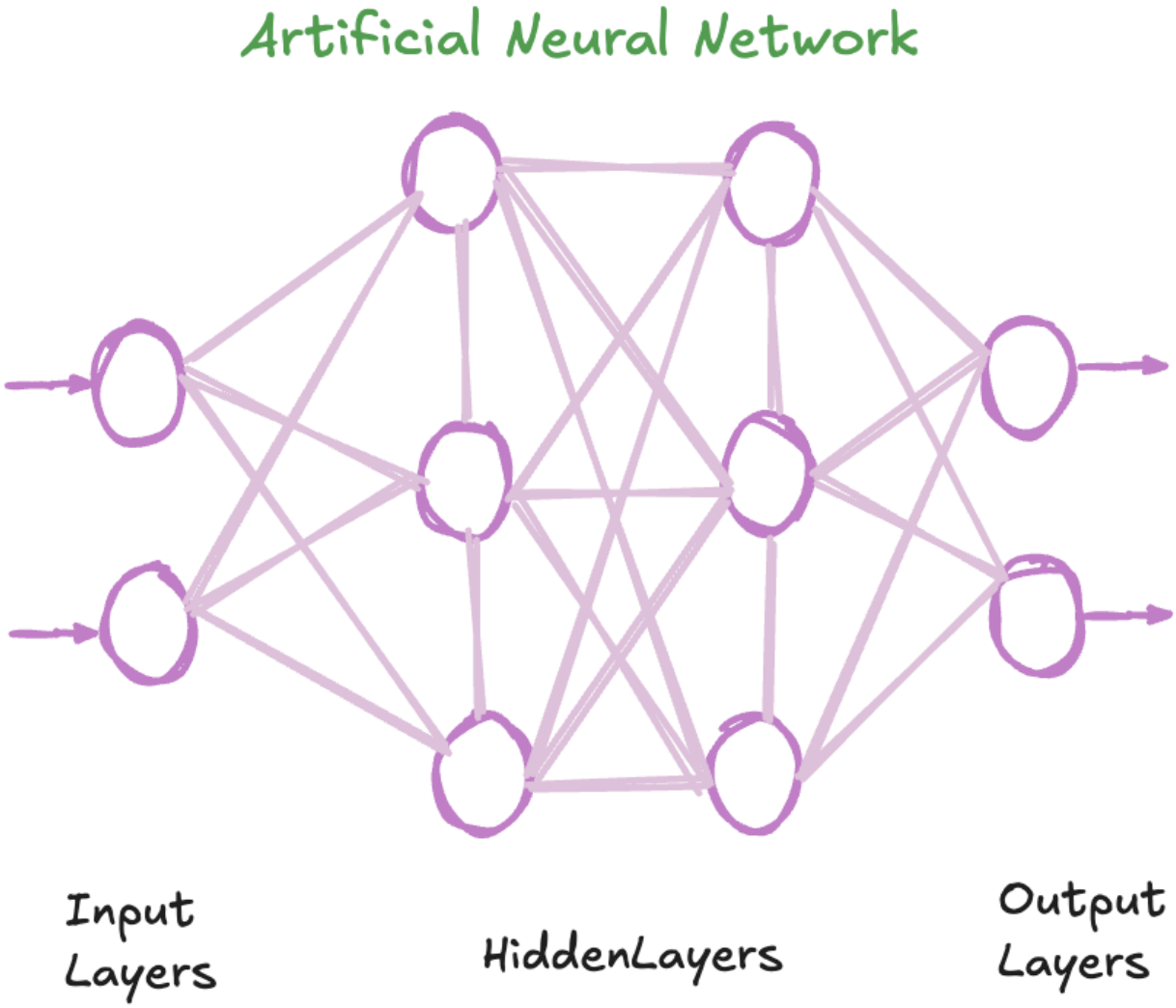
1.3.1 Different Types of AI



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

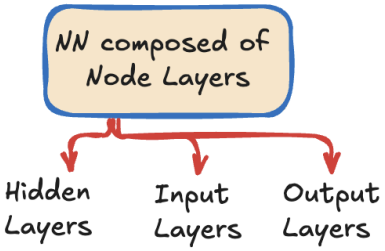
Omer Ilyas - Technical Marketing Engineer - oilyas@cisco.com

1.3.2 Neural Network

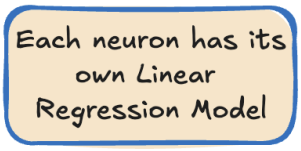


Note: The way nodes are connected called weights

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



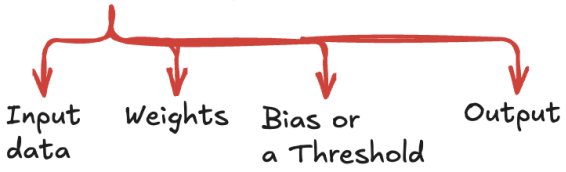
These neural network refelects the behaviour of human brain allowing computers to recognize patterns and solve common problems

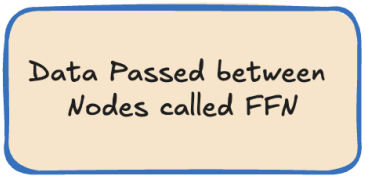


Linear Regression is the mathematical model that is used to predict future events

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

Each Node is composed of



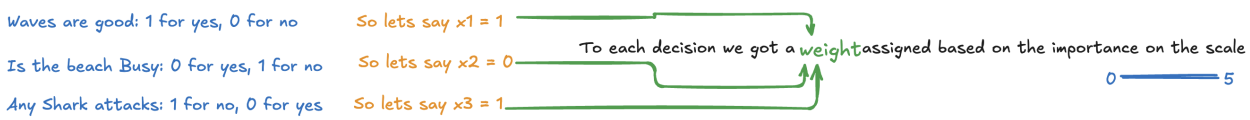


Omer Ilyas - Technical Marketing Engineer - oilyas@cisco.com

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

Example: Lets see how a single node in NN looks like when deciding if you wanna go surfing. The decision to go or not wil be our predictive outcome or lets call it our yHat

Assume there are 3 factors influencing our decision:



Lets assign weights to our decisions:

So Waves are good we will assign a weight w1 = 5

Beach not busy not of great importance, w2 = 2

No Shark attacks, lets agive it a weight w3 = 4

Overall Result:

x1 = 1 x2 = 0 x3 =1

w1 = 5 w2 = 2 w3 =4

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

$$yHat = (1*5) + (0*2) + (1*4)$$

$$yHat = 9 - 3 = 6$$

Threshold or Bias

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

We can always adjust weights to get different outcomes

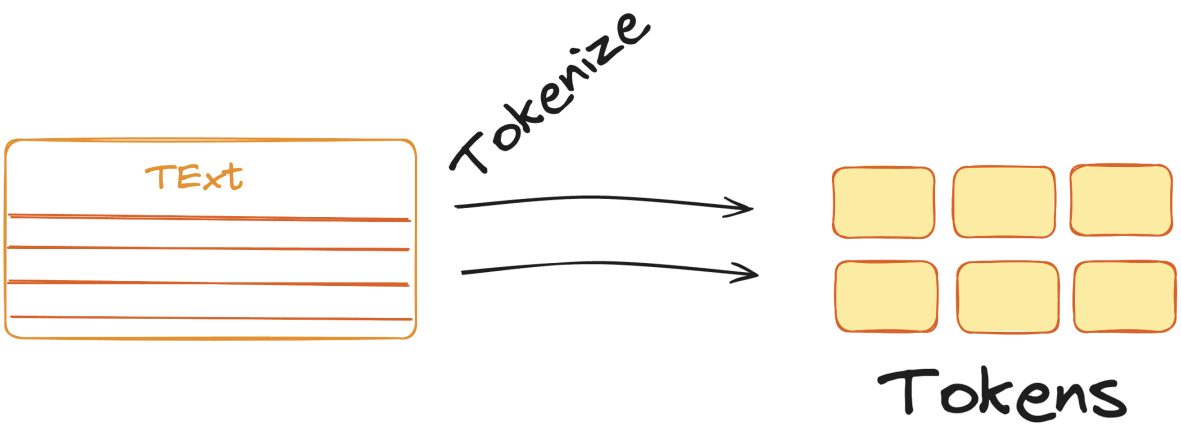
Note: Neural network rely on training data to learn and improve their accuracy over time. We used supervised learning to train the algorithm. There are multiple types of Neural network other than the Feed forward network that we have defined here. Example: CNN: Convolutional Neural Network - Unique architecture 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.

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

1.4 Tokenization

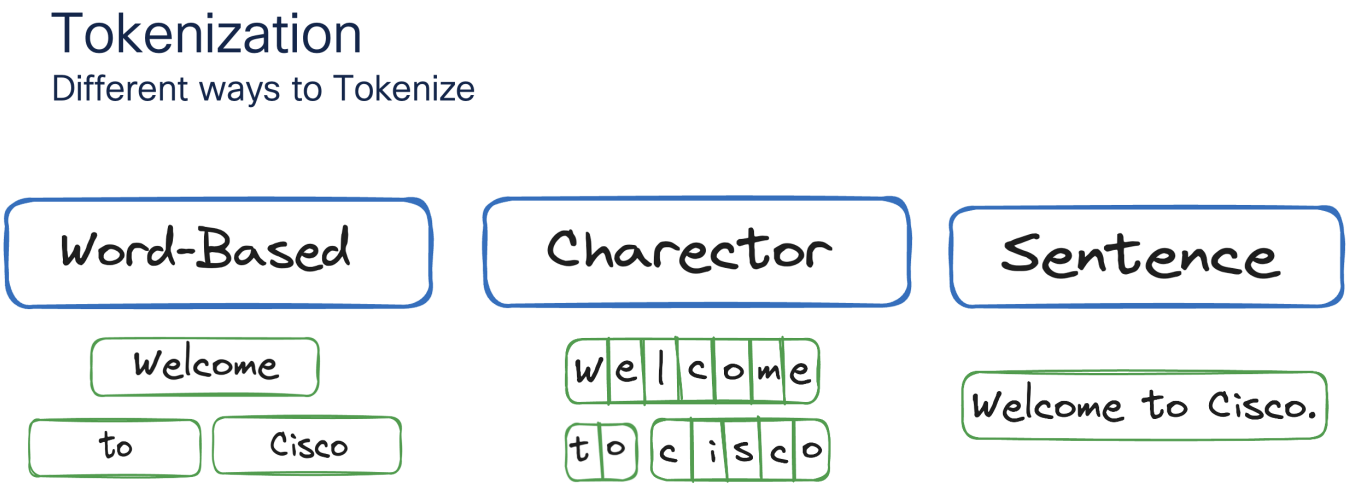
Download Omer Image2

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.



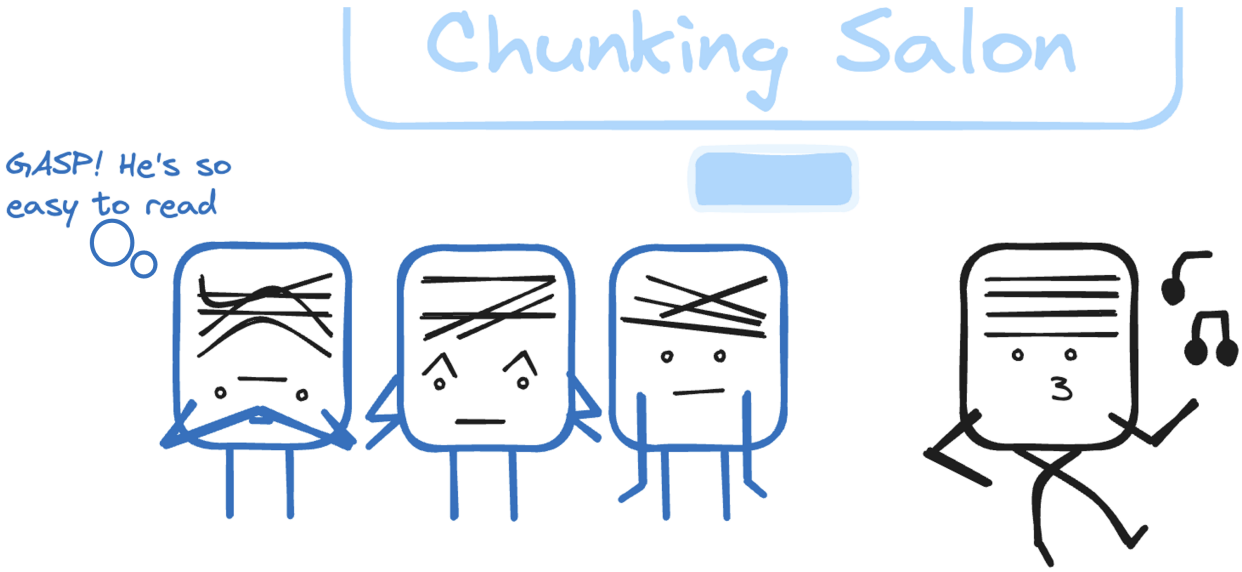
Omer Ilyas - Technical Marketing Engineer - oilyas@cisco.com

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 serentence 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?

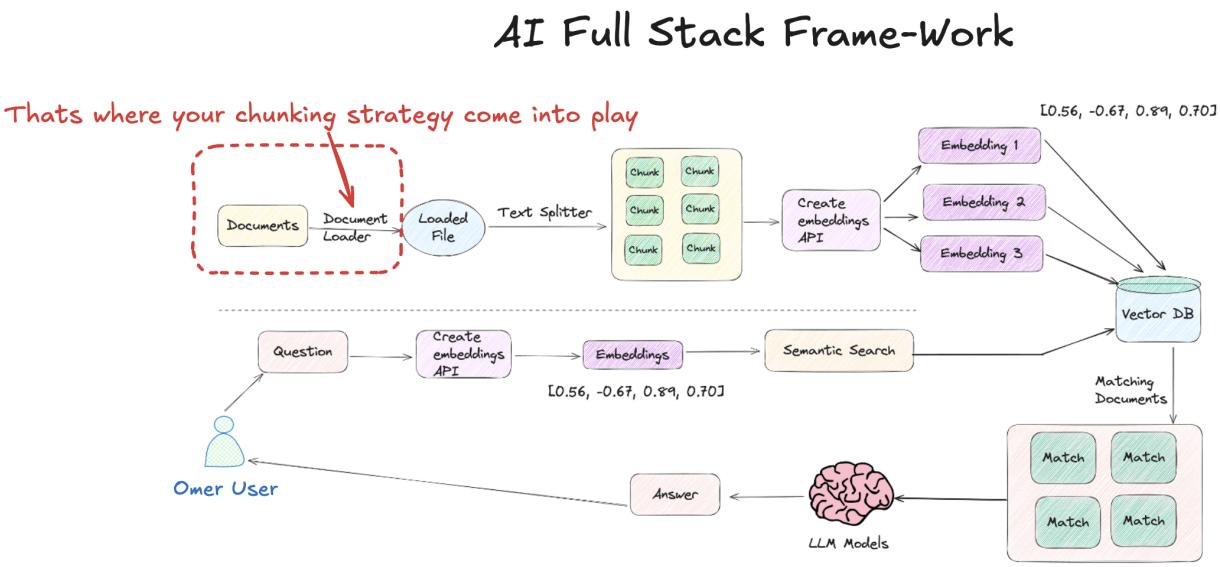


Historically, applications perform better when they are provided with your own data. However, you can't feed unlimited data to your LLMs due to two key limitations:

- 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.

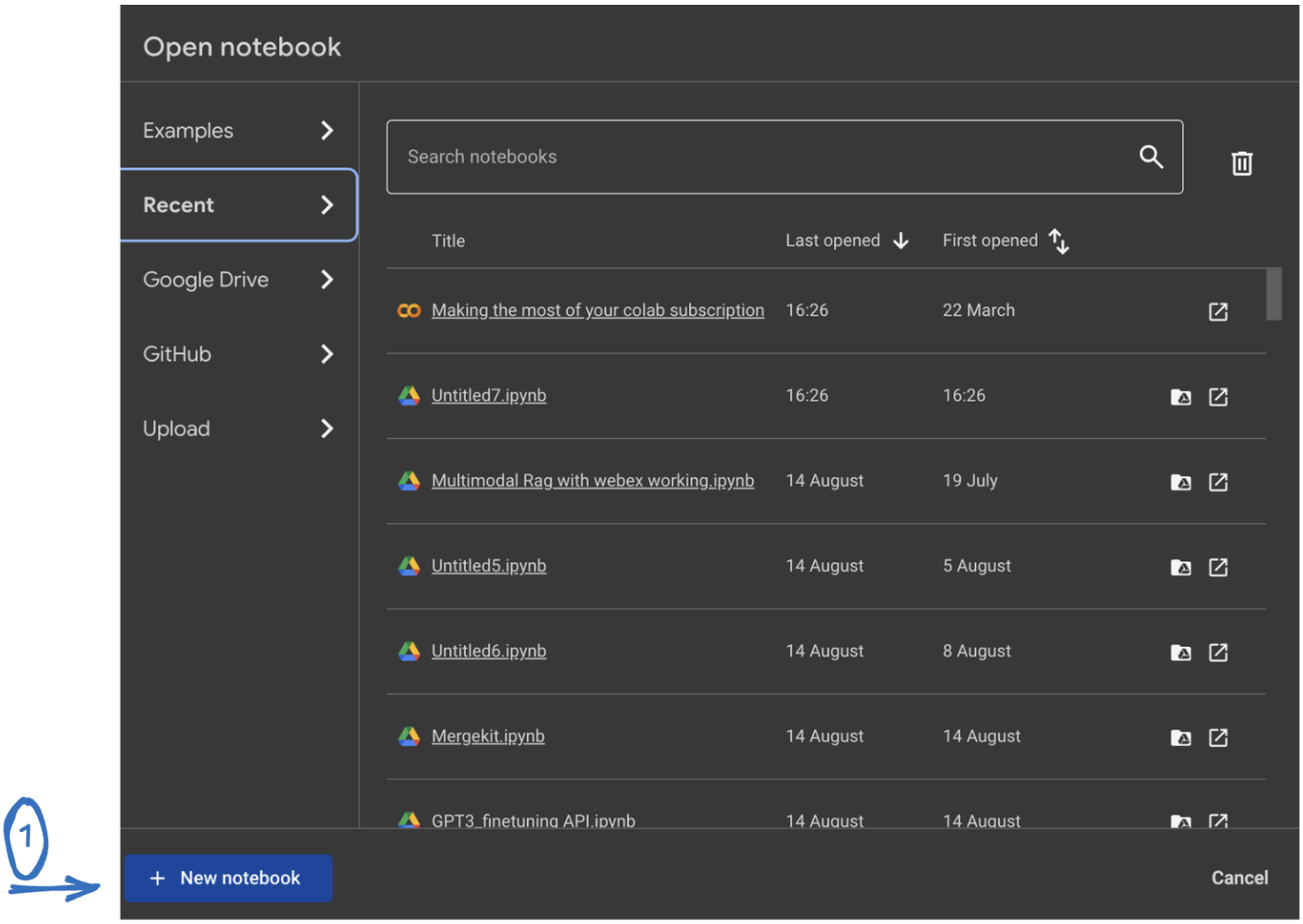
As noted, chunking or splitting refers to breaking your data into smaller, manageable pieces.

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



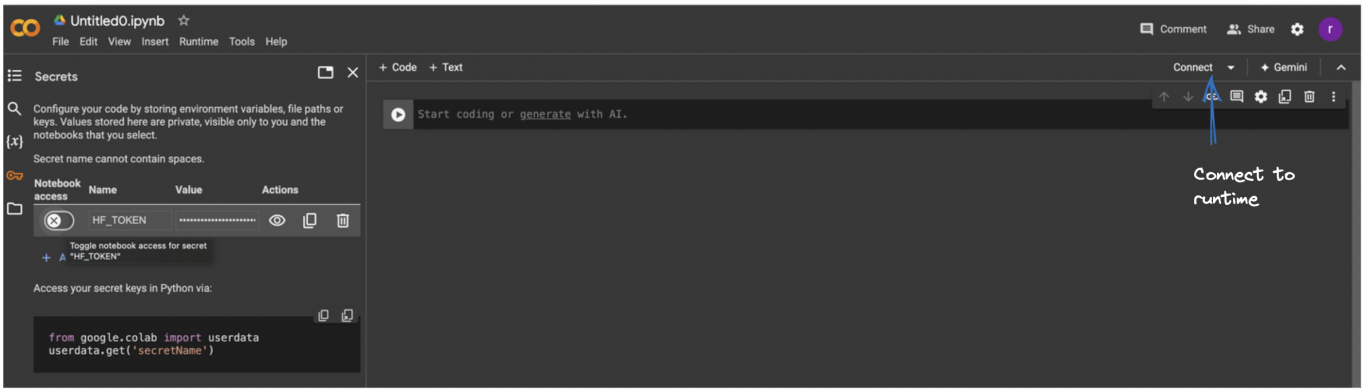
1.4.2 Task 1: Log into the Lab Environment

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

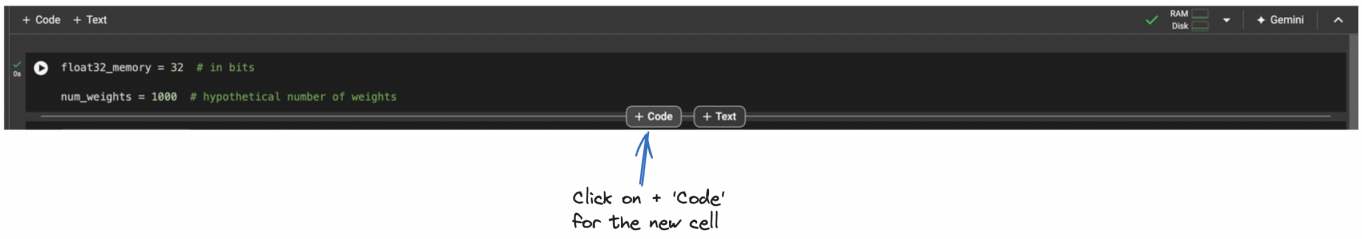


- Make sure you are connected to a runtime. For this task, you can use the CPU as the runtime environment.

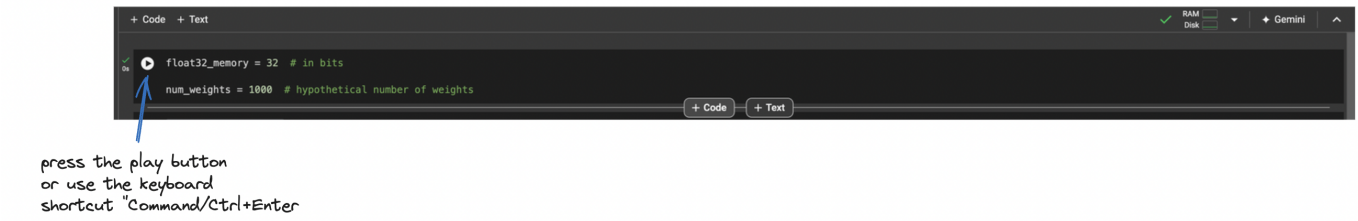
Omer Ilyas - Technical Marketing Engineer - oilyas@cisco.com



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.

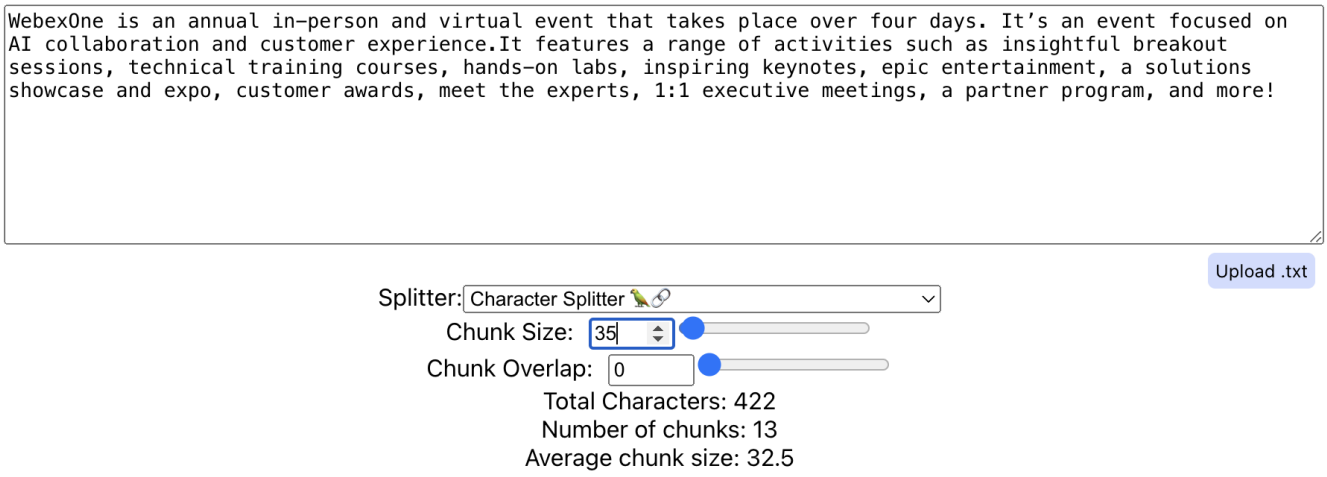


Manual testing for Chunking

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

```
# Manual Splitting
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!"""

# Create a list that will hold your chunks
chunks = []

chunk_size = 35

# Run through the a range with the length of your text and iterate every chunk_size you want
for i in range(0, len(text), chunk_size):
    chunk = text[i:i + chunk_size]
    chunks.append(chunk)

print(chunks)
```


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



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.

```
!pip install langchain
```

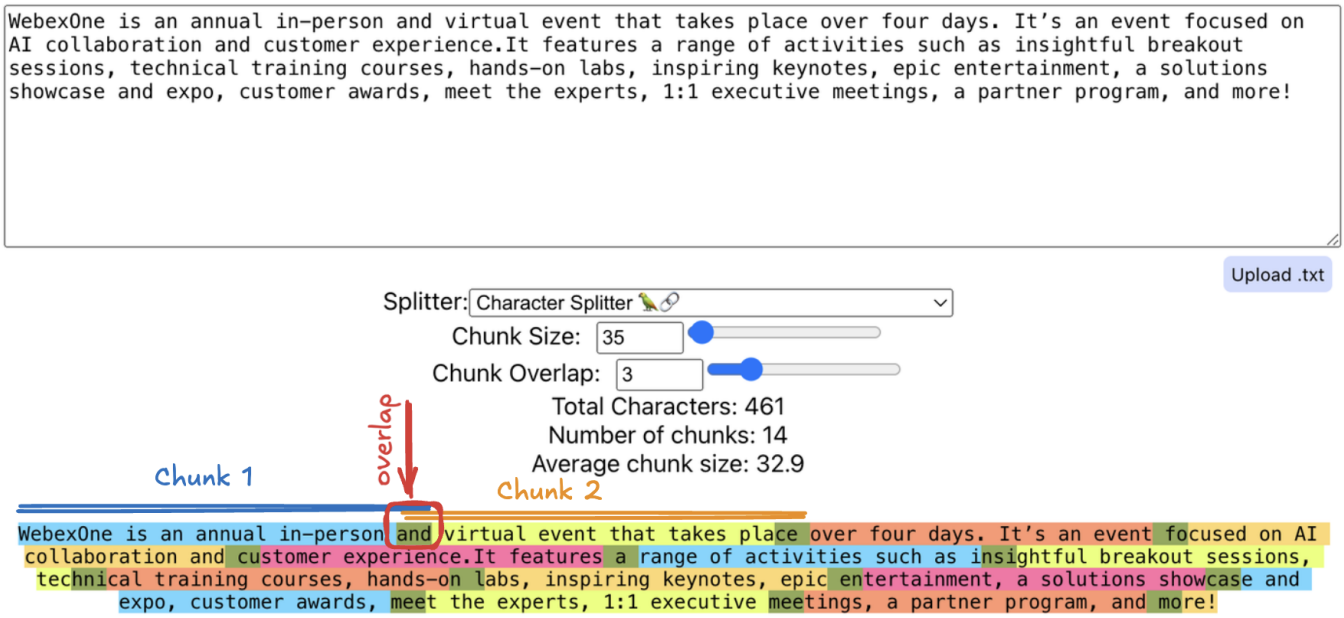
```
Collecting langchain
  Downloading langchain-0.2.14-py3-none-any.whl.metadata (7.1 kB)
Requirement already satisfied: PyYAML>=5.3 in /usr/local/lib/python3.10/dist-packages (from langchain) (6.0.2)
Requirement already satisfied: SQLAlchemy<3,>=1.4 in /usr/local/lib/python3.10/dist-packages (from 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<3,>=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: aiohappyeyeballs>=2.3.0 in /usr/local/lib/python3.10/dist-packages (from aiohttp<4.0.0,>=3.8.3->langchain) (2.3.5)
Requirement already satisfied: aiosignal>=1.1.2 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-py2.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)
```

```
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!"""
```

```
from langchain.text_splitter import CharacterTextSplitter

text_splitter = CharacterTextSplitter(chunk_size=35, chunk_overlap=3, separator='', strip_whitespace=False) # seprate means you splt by charector

a = text_splitter.create_documents([text])

print(a)
```

Note: Seprators are charectors sequence you wanna split on.

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



Recursive Charector Text Splitting

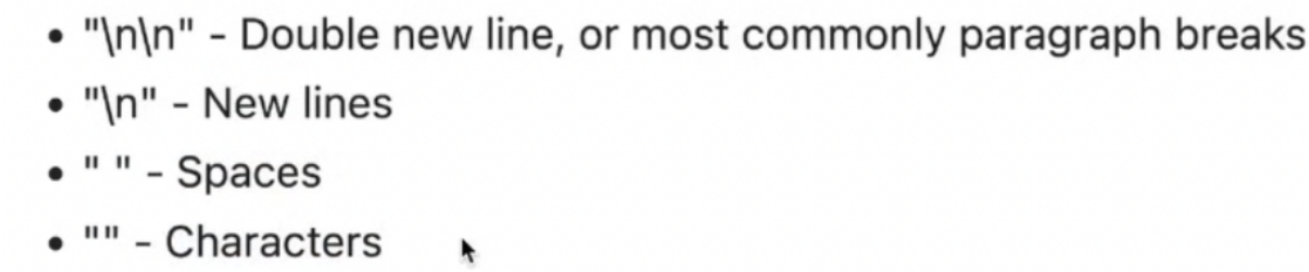
In the previous example, when we used Character Splitting, we split the text based on a fixed number of characters. Specifically, we divided the text into chunks of 35 characters each.

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.

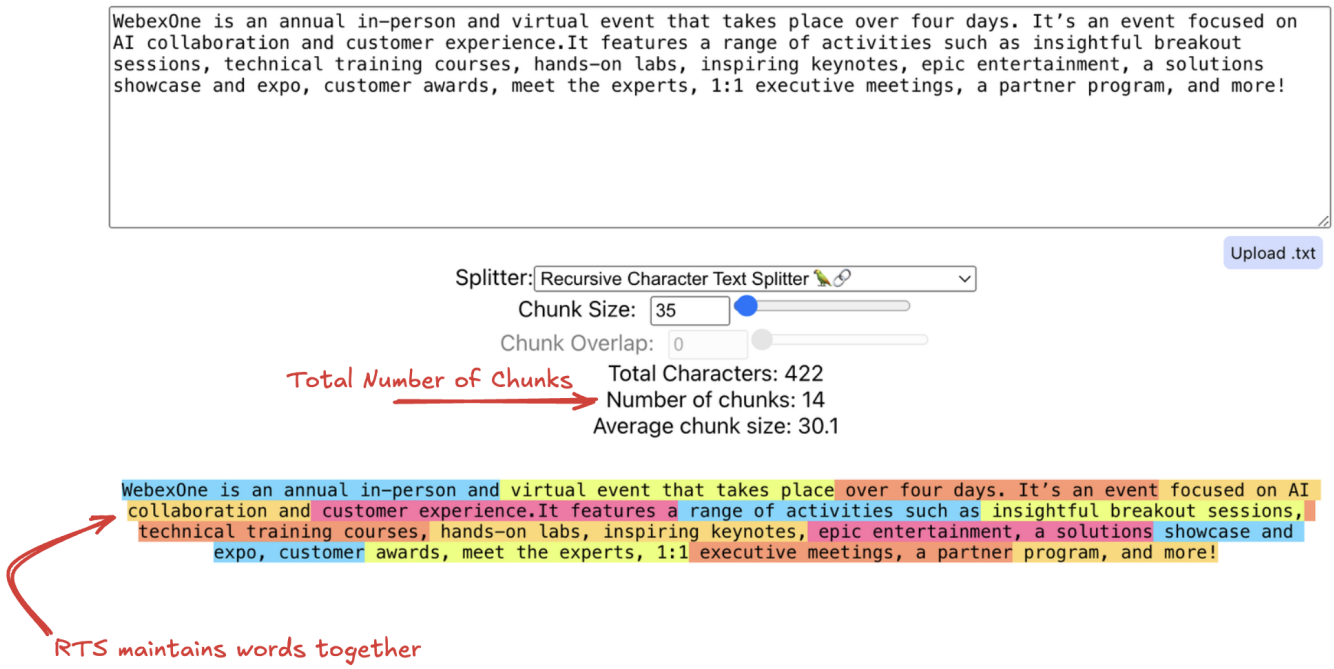
With RTS, you don't need to manually specify the chunk size by character count. You simply pass your text to the RTS process, and it will intelligently determine how to split the text based on its structure, resulting in chunks that are logically organized and more contextually meaningful.




```
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!"""
```

```
from langchain.text_splitter import RecursiveCharacterTextSplitter
text_splitter = RecursiveCharacterTextSplitter(chunk_size = 35, chunk_overlap = 0)
# how many chunks we have
print(len(text_splitter.create_documents([text])))

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.



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

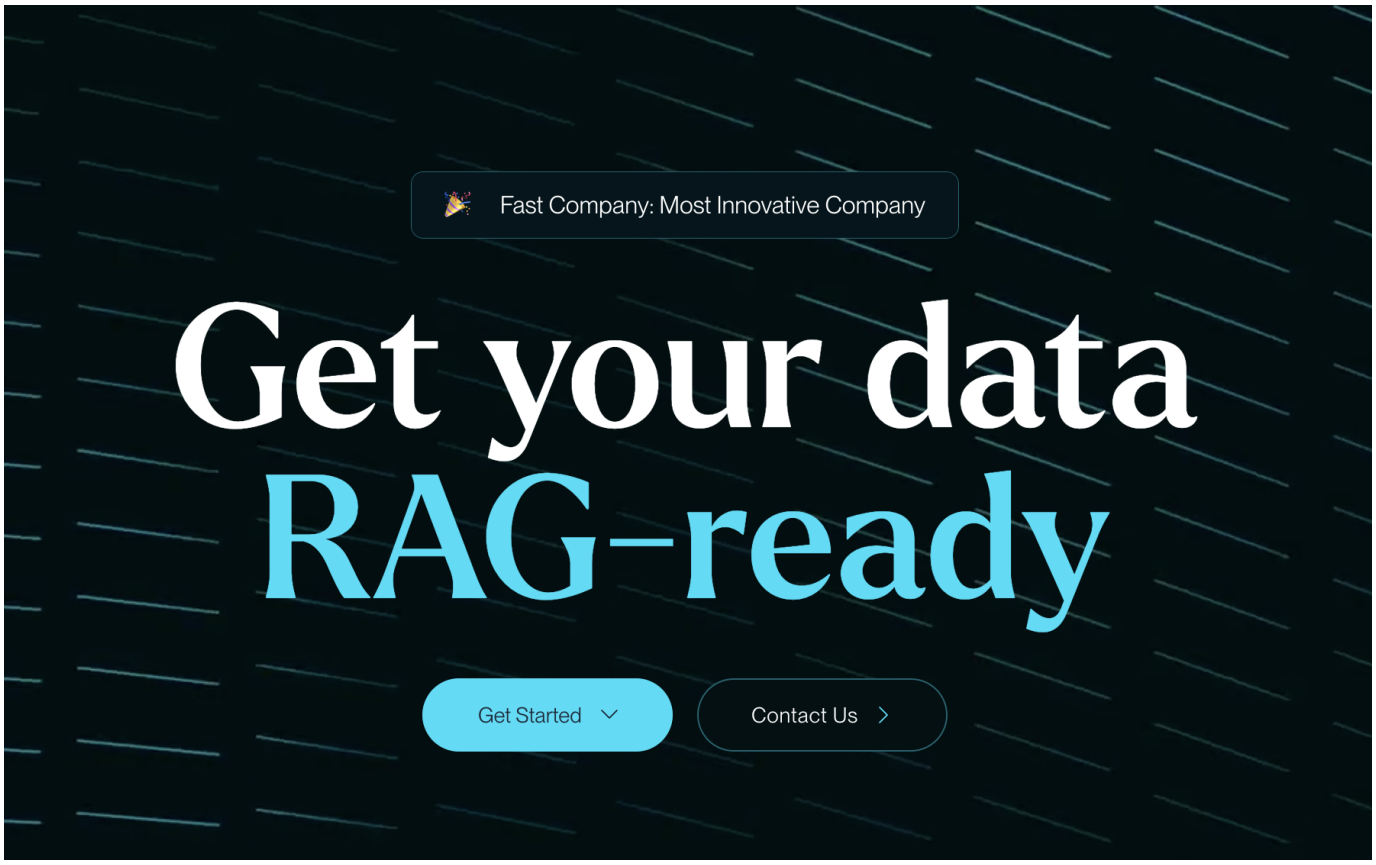
Document Level Splitting

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.

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



- Install relevant libraries for Unstructured

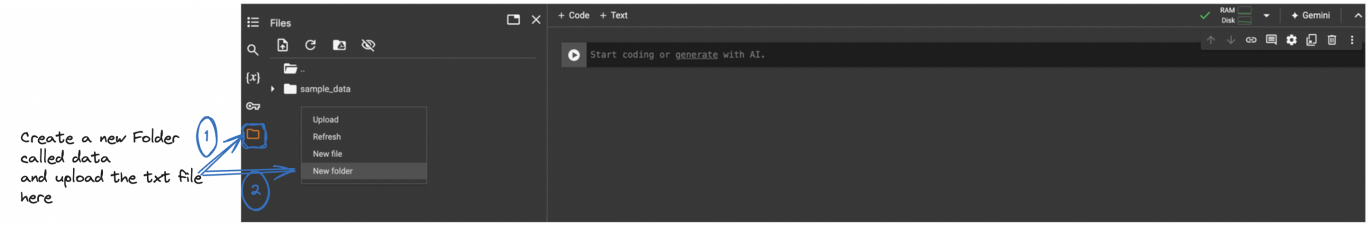
```
!pip install scikit-learn langchain_community unstructured[all-docs] unstructured pdfminer pdfminer.six pdf2image pillow_heif opencv-python
unstructured_inference pytesseract unstructured_pytesseract
```

```
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)
```

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

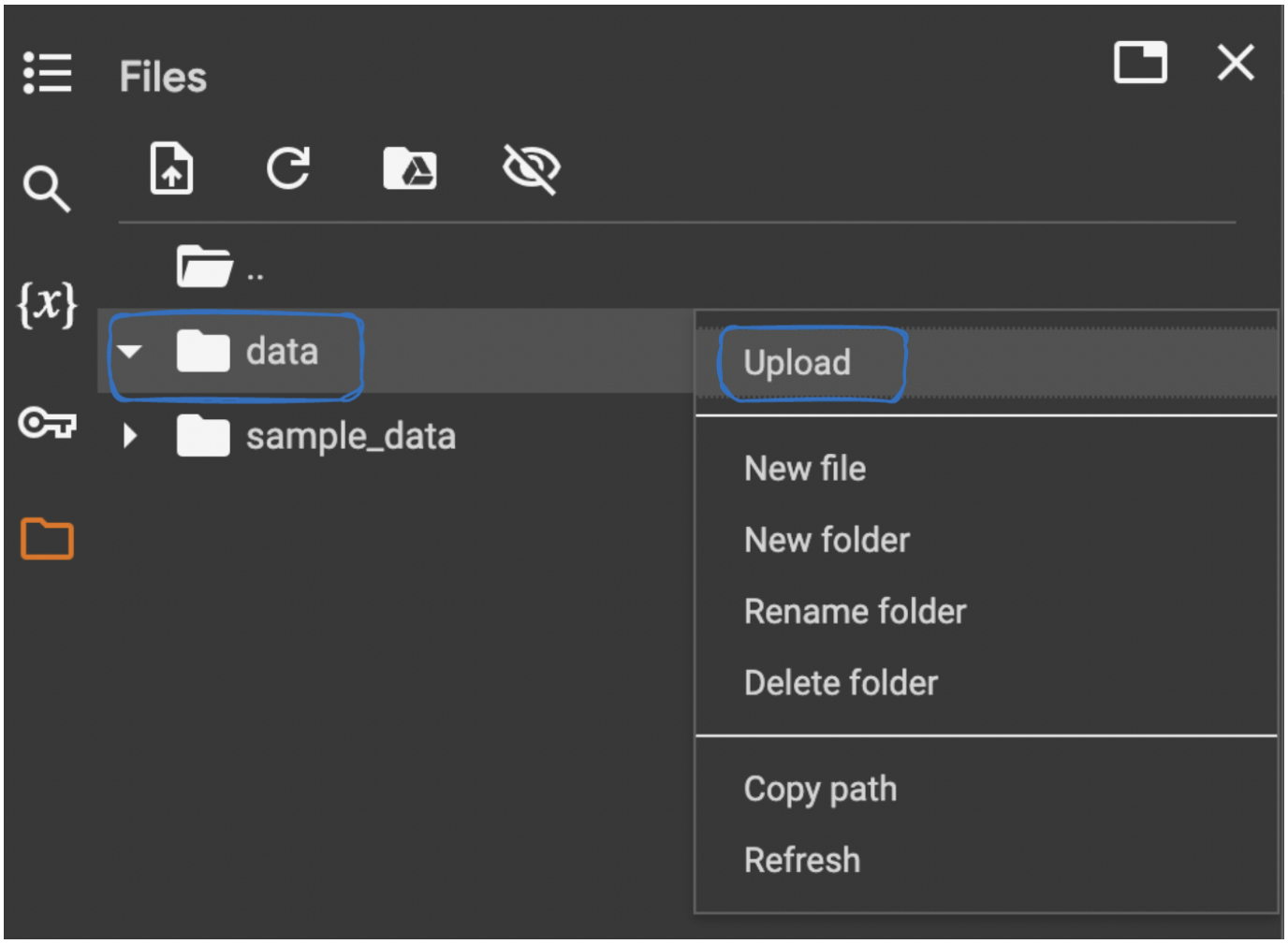
```
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following NEW packages will be installed:
  poppler-utils
0 upgraded, 1 newly installed, 0 to remove and 45 not upgraded.
Need to get 186 kB of archives.
After this operation, 696 kB of additional 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"



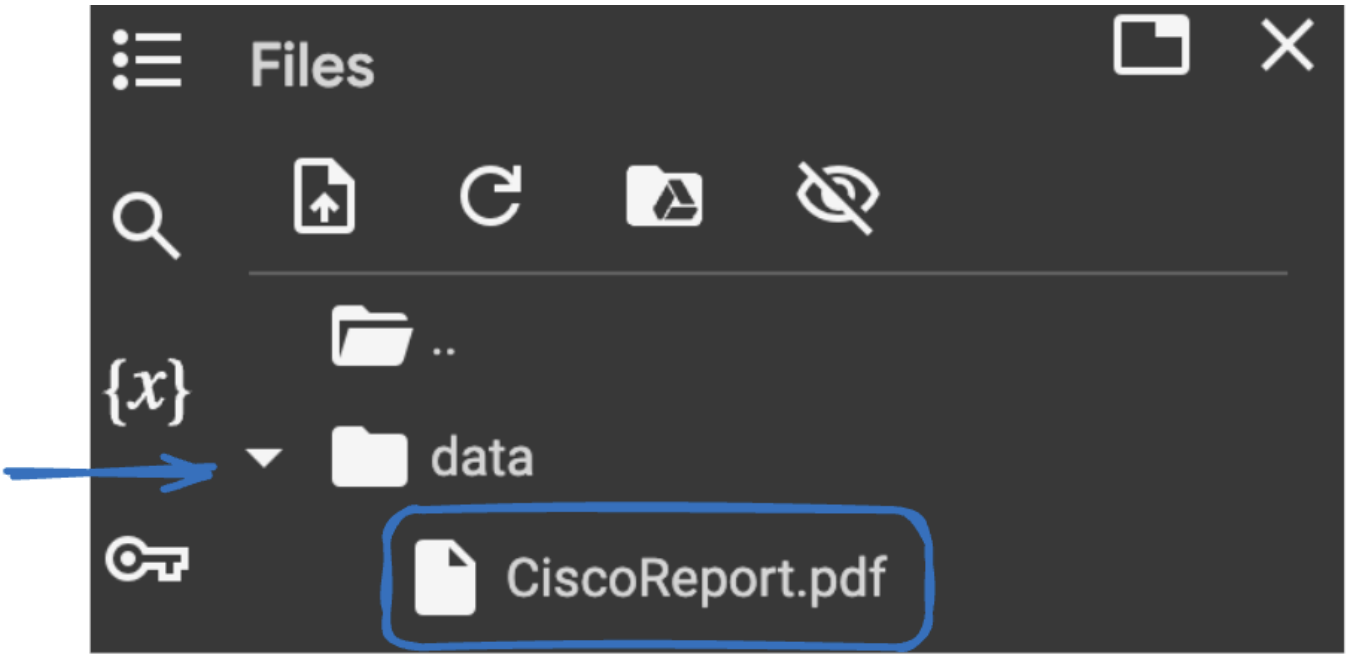
- Click on [...], select Upload

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



- Choose your CiscoReport.pdf file and click Open

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



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

```
import os
from unstructured.partition.pdf import partition_pdf
from unstructured.staging.base import elements_to_json

# Let's load up our PDF and then partition it.
filename = "/content/data/CiscoReport.pdf"# Use relative path since the file is in the same directory

# Extracts the elements from the PDF
elements = partition_pdf(
    filename=filename,
    extract_images_in_pdf=True,
    strategy="hi_res",
    infer_table_structure=True,
    hi_res_model_name="yolox"
)

# Let's look at our elements
print(elements)
```


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

```
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>,
<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>,
<unstructured.documents.elements.Title at 0x7ba4c0994790>,
```

Regular Text

Table

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

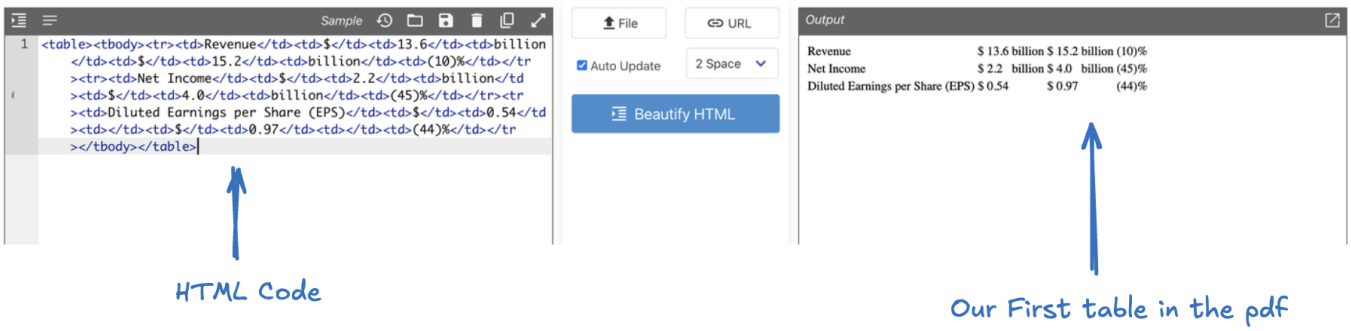
```
elements[37].metadata.text_as_html
```

OUTPUT

```
'<table><thead><tr><th>Revenue</th><th>$8.51 - $8.53 Billion</th><th>$34.5 - $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></td><td>~11.4%</td></tr><tr><td>Non-GAAP Operating Margin')</td><td></td><td>~28.0%</td></tr><tr><td>GAAP Earnings per Share?)</td><td>$0.79 - $0.80</td><td>$2.67 - $2.69</td></tr><tr><td>Non-GAAP Earnings per Share()</td><td>$1.89 - $1.90</td><td>$7.41 - $7.43</td></tr><tr><td>Operating Cash Flow Growth (Y/Y)°)</td><td></td><td>16% - 17%</td></tr><tr><td>Current Remaining Performance Obligation Growth (Y/Y)</td><td>~10%</td><td></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

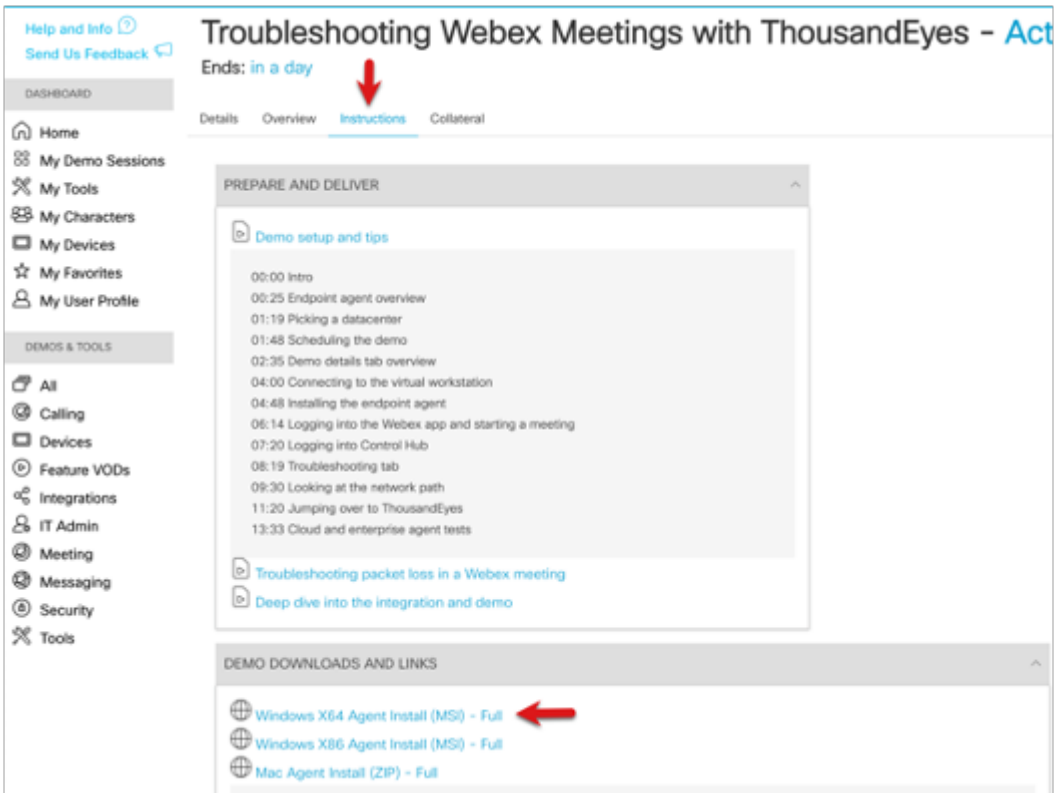


Note: That's how you handle tables within a PDF.

1.5 Task 4: Install the ThousandEyes EPA

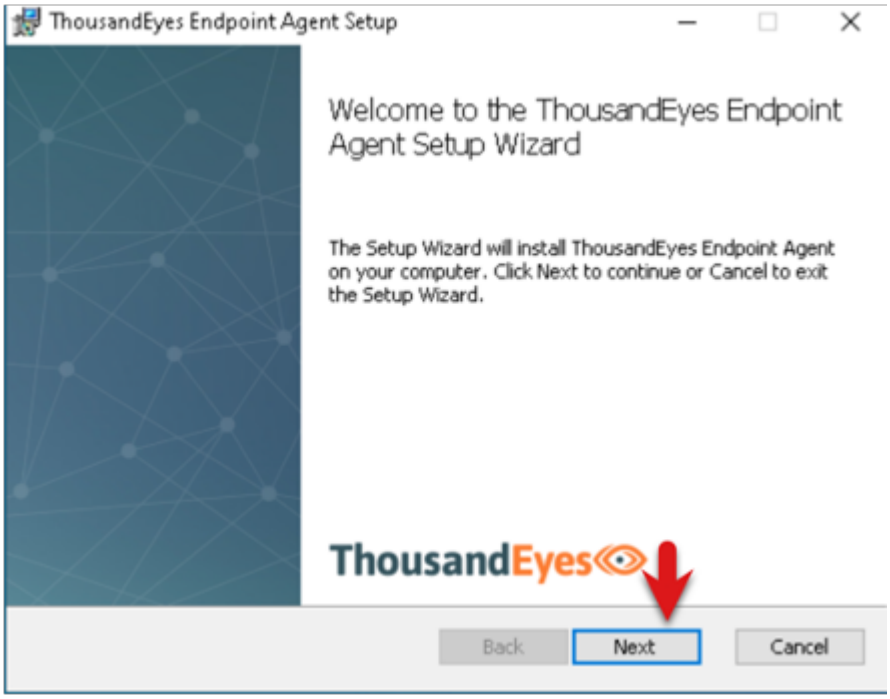
Once on your VM station,make sure no browser sessions are running on Window VM then click the ThousandEyes Agent Installer

You should NOT have to do this but if you don't see the ThousandEyes Agent Installer go back to the tab for the lab and click the Instructions link then copy the link for the Windows x64 Agent Install (MSI) – Full from the Demo Downloads and Links section. Navigate back to the Windows VM tab, open a browser and paste the link into it to download the installer. Locate the installer and click it to install it.

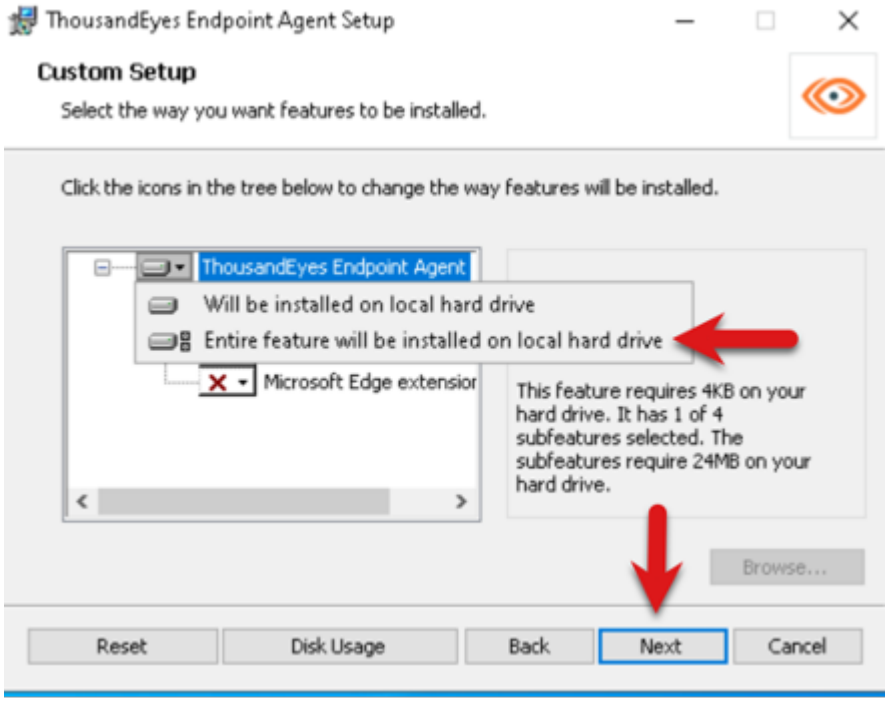


Click Next and Accept the License Agreement

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



Click the disk icon for ThousandEyes Endpoint Agent and Google Chrome and select Entire feature will be installed on local hard drive then click Next and Finish.

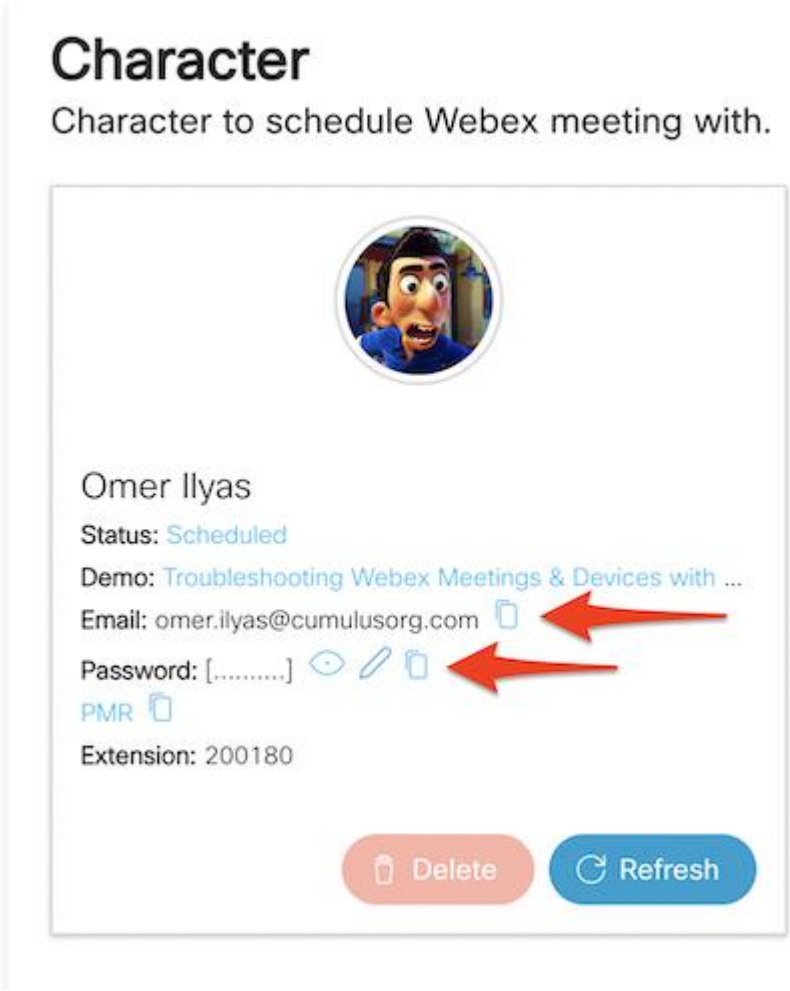


Note the above step will install ThousandEyes Endpoint agent on the VM provided. As mentioned earlier you can also get the install file from the Instruction tab and install the same on your personal machine (windows/mac)

1.5.1 Start up a Webex Meeting on your VM

This will generate Webex traffic which we will view later in the lab. Navigate back to your lab information tab. Copy the email address for the character you created to schedule a Webex meeting with. You will use this to login into Webex App and start a Webex meeting session.

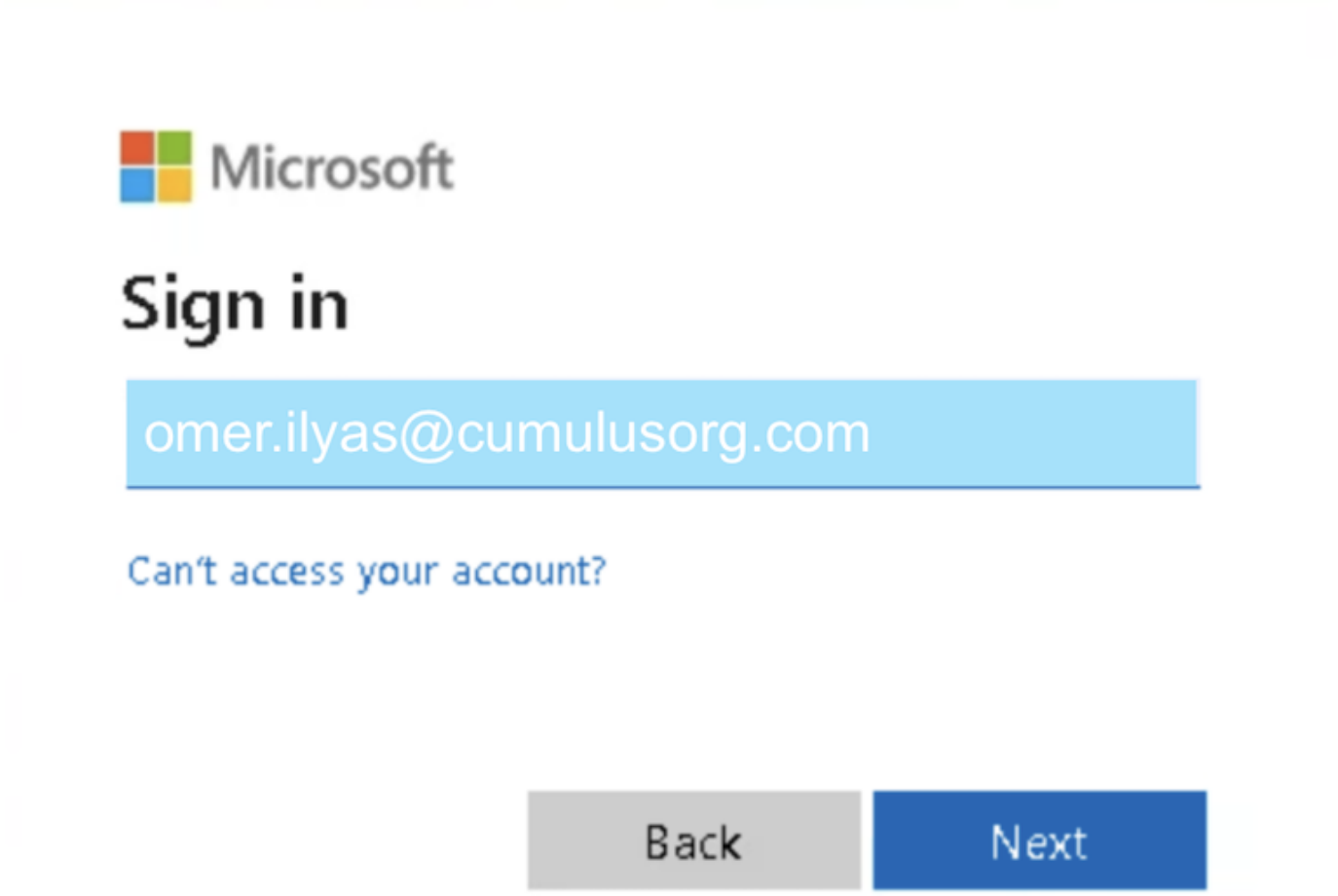
Omer Ilyas - Technical Marketing Engineer - oilyas@cisco.com

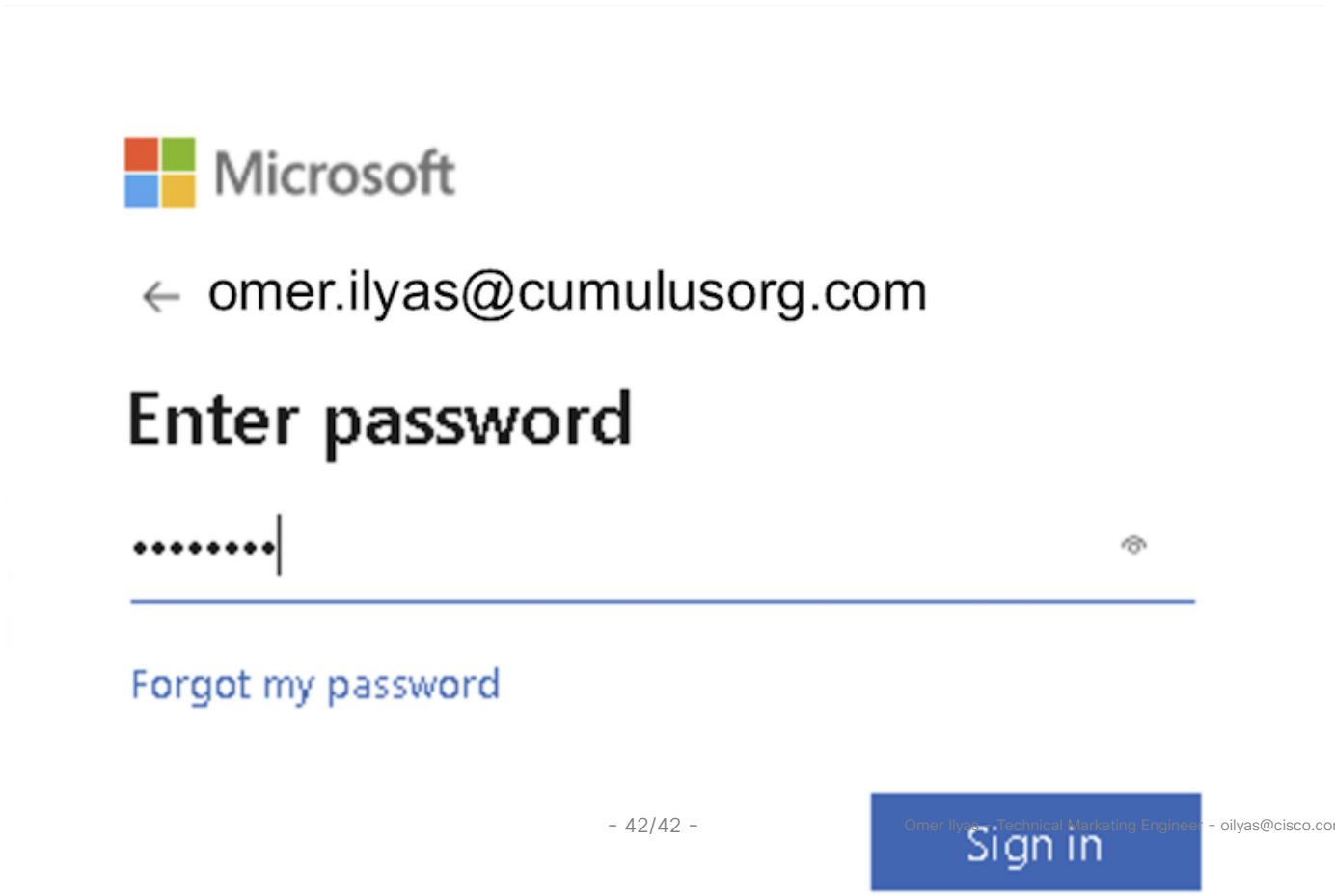


Navigate back to your VM web browser tab and click on the Webex icon then sign in using the email address and password from your character.

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

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





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