**To set up the environment to run the above RAG-based classification on your laptop, follow these steps:**

**1. System Requirements**

* **Operating System: Linux, macOS, or Windows (with WSL recommended for Linux tools).**
* **Hardware: At least 16 GB RAM and a GPU with CUDA support (optional but recommended for faster training/inference).**
* **Software: Python 3.8 or higher.**

**2. Install Necessary Tools and Libraries**

**Step 1: Install Python**

* **Download and install Python 3.8+ from** [**python.org**](https://www.python.org/)**.**
* **Ensure pip is installed:**

**python3 -m ensurepip --upgrade**

**Step 2: Install a Virtual Environment Manager**

**To isolate your environment:**

**pip install virtualenv**

**Create a virtual environment:**

**virtualenv rag\_env**

**source rag\_env/bin/activate # For Linux/Mac**

**rag\_env\Scripts\activate # For Windows**

**Step 3: Install Required Python Libraries**

**Install the core libraries:**

**pip install torch torchvision torchaudio --index-url https://download.pytorch.org/whl/cu118 # GPU version**

**pip install transformers sentence-transformers faiss-cpu datasets**

**For GPU support with Faiss (optional):**

**pip install faiss-gpu**

**Additional utilities:**

**pip install numpy pandas scikit-learn**

**Step 4: Verify GPU Availability (Optional)**

**If you have a GPU:**

**import torch**

**print(torch.cuda.is\_available()) # Should return True**

**3. Download and Setup Models**

**Pre-trained models will be fetched automatically when the code runs. To save bandwidth, pre-download models:**

**Retriever Model**

**from sentence\_transformers import SentenceTransformer**

**model = SentenceTransformer('all-MiniLM-L6-v2') # Downloads embedding model**

**Generator Model**

**from transformers import AutoModelForSeq2SeqLM, AutoTokenizer**

**model = AutoModelForSeq2SeqLM.from\_pretrained("google/flan-t5-base")**

**tokenizer = AutoTokenizer.from\_pretrained("google/flan-t5-base")**

**4. Set Up FAISS for Retrieval**

**Create a FAISS index for your knowledge base:**

**import faiss**

**import numpy as np**

**from sentence\_transformers import SentenceTransformer**

**# Load embedding model**

**embed\_model = SentenceTransformer('all-MiniLM-L6-v2')**

**# Encode sample knowledge base**

**knowledge\_base = ["Sample document 1", "Sample document 2"]**

**kb\_embeddings = embed\_model.encode(knowledge\_base)**

**# Create FAISS index**

**dimension = kb\_embeddings.shape[1]**

**index = faiss.IndexFlatL2(dimension) # L2 distance for similarity**

**index.add(np.array(kb\_embeddings)) # Add embeddings to the index**

**5. Code Repository Structure**

**Organize your project as follows:**

**graphql**

**rag\_project/**

**│**

**├── main.py # Your main script**

**├── requirements.txt # List of dependencies**

**├── data/ # Your dataset**

**├── models/ # Pre-downloaded models (optional)**

**└── faiss\_index/ # Serialized FAISS index (optional)**

**requirements.txt**

**torch**

**transformers**

**sentence-transformers**

**faiss-cpu**

**numpy**

**pandas**

**scikit-learn**

**datasets**

**Install dependencies in one go:**

**pip install -r requirements.txt**

**6. Running the Script**

**Fine-Tuning**

**If you’re training a RAG model:**

**python main.py --mode train --data\_path data/train.json --output\_dir models/**

**Inference**

**If you’re using the model for predictions:**

**python main.py --mode predict --query "Sample query here" --kb\_path faiss\_index/**

**7. Example main.py Script**

**Here’s a simplified script you can start with:**

**python**

**import argparse**

**import json**

**from transformers import AutoModelForSeq2SeqLM, AutoTokenizer**

**from sentence\_transformers import SentenceTransformer**

**import faiss**

**import numpy as np**

**def retrieve\_context(query, embed\_model, faiss\_index, knowledge\_base):**

**query\_embedding = embed\_model.encode([query])**

**\_, indices = faiss\_index.search(np.array(query\_embedding), k=5)**

**return " ".join([knowledge\_base[i] for i in indices[0]])**

**def classify(query, context, generator\_model, tokenizer):**

**input\_text = f"Query: {query} Context: {context}"**

**inputs = tokenizer(input\_text, return\_tensors="pt")**

**outputs = generator\_model.generate(\*\*inputs)**

**return tokenizer.decode(outputs[0], skip\_special\_tokens=True)**

**def main():**

**parser = argparse.ArgumentParser()**

**parser.add\_argument("--mode", choices=["train", "predict"], required=True)**

**parser.add\_argument("--query", type=str, help="Input query for classification")**

**parser.add\_argument("--kb\_path", type=str, help="Path to FAISS index")**

**parser.add\_argument("--data\_path", type=str, help="Path to training data")**

**parser.add\_argument("--output\_dir", type=str, help="Path to save model")**

**args = parser.parse\_args()**

**# Load models**

**embed\_model = SentenceTransformer('all-MiniLM-L6-v2')**

**tokenizer = AutoTokenizer.from\_pretrained("google/flan-t5-base")**

**generator\_model = AutoModelForSeq2SeqLM.from\_pretrained("google/flan-t5-base")**

**# Example for prediction**

**if args.mode == "predict":**

**# Load FAISS index**

**knowledge\_base = ["Document 1 text", "Document 2 text"] # Example**

**kb\_embeddings = embed\_model.encode(knowledge\_base)**

**index = faiss.IndexFlatL2(kb\_embeddings.shape[1])**

**index.add(np.array(kb\_embeddings))**

**context = retrieve\_context(args.query, embed\_model, index, knowledge\_base)**

**result = classify(args.query, context, generator\_model, tokenizer)**

**print("Classification Result:", result)**

**if \_\_name\_\_ == "\_\_main\_\_":**

**main()**

**8. Verify Installation**

**Run a simple test with your knowledge base and query to confirm everything works:**

**python main.py --mode predict --query "Classify this sentence." --kb\_path "path\_to\_faiss\_index"**

**This should produce a classification result.**

**4o**

**You said:**

**pip install torch torchvision torchaudio --index-url https://download.pytorch.org/whl/cu118 # GPU version How to download all dependent packages**

**ChatGPT said:**

**ChatGPT**

**To install PyTorch and all its dependent packages in one go, follow these steps:**

**Step 1: Install PyTorch with Dependencies**

**Use the following command to install PyTorch and its dependencies, including torchvision and torchaudio, from the official PyTorch index:**

**For GPU (CUDA 11.8):**

**pip install torch torchvision torchaudio --index-url https://download.pytorch.org/whl/cu118**

**For CPU Only:**

**If you don’t have a GPU or don’t want CUDA:**

**pip install torch torchvision torchaudio --index-url https://download.pytorch.org/whl/cpu**

**Step 2: Verify Installation**

**After installation, verify that PyTorch and CUDA are properly installed (if using GPU):**

**Check PyTorch Installation:**

**python**

**import torch**

**print(torch.\_\_version\_\_) # Should print the PyTorch version**

**print(torch.cuda.is\_available()) # Should return True if GPU is available**

**Check GPU Availability:**

**python**

**if torch.cuda.is\_available():**

**print(f"CUDA is available. GPU: {torch.cuda.get\_device\_name(0)}")**

**else:**

**print("CUDA is not available. Running on CPU.")**

**Step 3: Install Additional Dependencies**

**PyTorch alone might not cover all dependencies for your project. Install the following packages required for RAG-based classification:**

**pip install transformers sentence-transformers faiss-cpu datasets numpy pandas scikit-learn**

**If you need GPU-accelerated FAISS, replace faiss-cpu with faiss-gpu:**

**pip install faiss-gpu**

**Step 4: Install Everything from a requirements.txt File**

**To simplify the process, create a requirements.txt file containing all dependencies:**

**Example requirements.txt:**

**txt**

**torch**

**torchvision**

**torchaudio**

**transformers**

**sentence-transformers**

**faiss-cpu**

**datasets**

**numpy**

**pandas**

**scikit-learn**

**Install All at Once:**

**pip install -r requirements.txt**

**Step 5: Avoid Dependency Conflicts**

**If you encounter dependency conflicts:**

1. **Use a Fresh Virtual Environment:**

**python -m venv rag\_env**

**source rag\_env/bin/activate # Linux/Mac**

**rag\_env\Scripts\activate # Windows**

1. **Use pip-tools to Resolve Conflicts: Install pip-tools:**

**pip install pip-tools**

**Create a requirements.in file with high-level dependencies:**

**txt**

**torch**

**torchvision**

**torchaudio**

**transformers**

**sentence-transformers**

**Compile the dependencies into a resolved requirements.txt:**

**pip-compile requirements.in**

**pip install -r requirements.txt**

**Optional: Use Conda for Simplified Management**

**If you prefer Conda, it can simplify installing dependencies with proper versioning.**

**Install PyTorch with GPU (Conda)**

**conda install pytorch torchvision torchaudio pytorch-cuda=11.8 -c pytorch -c nvidia**

**Add Other Dependencies**

**conda install -c conda-forge transformers sentence-transformers faiss-cpu numpy pandas scikit-learn**

**These steps will ensure you have all dependencies installed without conflicts!**

**4o**

**You said:**

**How to install al above if we don't have internet**

**ChatGPT said:**

**ChatGPT**

**If you don't have internet access on the target machine, you can still install all the required Python packages by following these steps:**

**1. Prepare on a Machine with Internet Access**

**Use a machine with internet access to download all the required packages and dependencies.**

**Step 1: Create a requirements.txt File**

**Prepare a list of dependencies for your project:**

**txt**

**torch**

**torchvision**

**torchaudio**

**transformers**

**sentence-transformers**

**faiss-cpu**

**datasets**

**numpy**

**pandas**

**scikit-learn**

**Step 2: Download Packages**

**Use pip to download all the required packages, including their dependencies, to a local folder.**

**mkdir offline\_packages**

**pip download -r requirements.txt -d offline\_packages**

**This will download all the specified packages and their dependencies into the offline\_packages folder.**

**Step 3: Include CUDA-Enabled PyTorch (Optional)**

**If you need GPU-enabled PyTorch, download the appropriate wheel file from the PyTorch website manually:**

1. **Visit PyTorch Get Started Page.**
2. **Select your PyTorch version, OS, and CUDA version.**
3. **Download the .whl file (e.g., torch-2.0.1+cu118-cp39-cp39-linux\_x86\_64.whl).**
4. **Save it into the offline\_packages folder.**

**2. Transfer Packages to the Offline Machine**

**Copy the offline\_packages folder to the offline machine using a USB drive, external hard drive, or local network transfer.**

**3. Install Packages on the Offline Machine**

**On the offline machine:**

**Step 1: Create a Virtual Environment**

**To isolate dependencies:**

**python -m venv rag\_env**

**source rag\_env/bin/activate # Linux/Mac**

**rag\_env\Scripts\activate # Windows**

**Step 2: Install Packages Locally**

**Install all the downloaded packages from the local folder:**

**pip install --no-index --find-links=offline\_packages -r requirements.txt**

**This tells pip to only look for packages in the offline\_packages folder.**

**4. Verify Installation**

**Once installed, test your setup by verifying key packages like torch and transformers:**

**python**

**import torch**

**from transformers import AutoTokenizer, AutoModel**

**print(torch.\_\_version\_\_) # Should print the PyTorch version**

**print(torch.cuda.is\_available()) # Check for GPU availability**

**5. Optional: Bundle All Dependencies**

**If the offline machine will repeatedly install the same setup, consider creating a wheel or archive of your environment:**

1. **On an internet-connected machine:**

**pip freeze > requirements\_full.txt**

**pip wheel -r requirements\_full.txt -w offline\_packages\_full**

1. **Transfer offline\_packages\_full to the offline machine and install as before:**

**pip install --no-index --find-links=offline\_packages\_full -r requirements\_full.txt**

**By downloading all dependencies beforehand and transferring them to the offline machine, you can install everything without requiring internet access.**