**Use Case: "FreshBite Organics"**

**Business:**  
FreshBite Organics is a small D2C business selling organic packaged foods online and via local stores. The team has 12 employees and limited tech expertise.

**Current Challenge:**  
Customers email and message product-related questions (e.g., ingredients, shipping time, shelf life, allergens). The small support team is overwhelmed.

**Goal: Automate First-Level Customer Support**

We want to **deploy an AI assistant trained to answer product FAQs** based on our own documentation and FAQs.

**Approach**

**1. Define the Problem Clearly**

* **Problem**: Manual handling of repetitive customer questions wastes 4–5 hours daily.
* **Solution**: Deploy an AI FAQ bot trained on internal knowledge (website FAQs, manuals).
* **Desired Outcome**: Cut support time by 60%, improve response consistency.

**2. Paid subscription like ChatGPT or open source model like Hugging face?**

| **Factor** | **Hugging Face Model** | **ChatGPT/Copilot** |
| --- | --- | --- |
| **Cost Control** | Run on local/own server | Paid per API usage/token |
| **Customization** | Fine-tune models on your data | No fine-tuning (ChatGPT) |
| **Data Privacy** | Data never leaves your infra | Data sent to external API |
| **Interface Options** | Easily integrate with apps | Copilot is IDE-focused |

**Decision:** Use Hugging Face (a QnA model).

**3. Select the Right Model**

**Model:** [deepset/roberta-base-squad2](https://huggingface.co/deepset/roberta-base-squad2)  
**Why:** It's built for question-answering over your own documents (SQuAD-style QA).

**4. Prepare the Knowledge Base**

* Collect internal docs to provide as context: product descriptions, FAQs, shipping policy, ingredients list.

**5. Write Code to Set Up the Model**

| **Code Section** | **Purpose** |
| --- | --- |
| pipeline("question-answering", model="deepset/roberta-base-squad2") | Loads a pretrained Hugging Face QA model for answering questions based on a given context **The model is downloaded once from Hugging Face**  * Location: ~/.cache/huggingface/transformers/ * This includes:   + pytorch\_model.bin → the pretrained model weights   + config.json, tokenizer\_config.json, etc.   These are cached locally and **not trained or modified**. |
| load\_context("freshbite\_faq.txt") | Reads internal business knowledge from a plain text file (e.g., product FAQs) |
| class QnAHandler | Tornado request handler that listens for /ask POST requests, runs the model, and returns the answer |
| app.listen(8686) | Starts a local web server to expose the model via an HTTP API |
| tornado.web.Application([...]) | Registers URL routes (e.g., /ask, /) and maps them to appropriate handlers |

**7. Operationalize It for Your Team**

* Host on internal server or low-cost cloud (e.g., Render, Railway, AWS Lightsail).
* Frontend via chatbot widget, internal Slack bot, or web interface.
* Update context file (freshbite\_faq.txt) as product line grows.

**📊 Business Impact**

| **Metric** | **Before** | **After** |
| --- | --- | --- |
| Time spent on routine queries | 4–5 hrs/day | < 1 hr/day |
| First-response time | 3–12 hrs | < 5 seconds |
| Support consistency | Manual | Standardized |

What we accomplished here :

* Load a general-purpose QA model (e.g., roberta-base-squad2)
* Provide context dynamically at **inference time**
* Get an answer — but the model is **not "learning"** from this context.

## **What if you want the model to learn from your context (not just read it):**

You have **two major options**, depending on your goal and scale:

### **Option 1: Fine-Tune the Model on Your Context**

Train the Hugging Face model using your data so that it learns domain-specific language, patterns, and answers.

#### What it requires:

* Convert your FAQs into **SQuAD-style format**:

{

"data": [

{

"title": "FreshBite FAQs",

"paragraphs": [

{

"context": "FreshBite Organics ships orders within 48 hours.",

"qas": [

{

"question": "How long does shipping take?",

"id": "001",

"answers": [

{"text": "within 48 hours", "answer\_start": 21}

],

"is\_impossible": false

}

]

}

]

}

]

}

* Use Trainer from Hugging Face transformers to fine-tune:

from transformers import Trainer, TrainingArguments, AutoModelForQuestionAnswering, AutoTokenizer

* You’ll need:
  + A GPU (or Colab)
  + Clean, structured training data
  + ~15–100 Q&A examples (small-scale fine-tuning)

#### Result:

* A **custom model** that doesn’t need external context — it learns your domain.
* Slightly slower to set up, but gives **faster inference and better alignment**.

### **Option 2: Use Embedding + Vector Search (Retrieval-Augmented QA)**

This is the smarter, more scalable way.

Instead of fine-tuning the model, **convert your FAQs into embeddings**, store them in a vector database (like FAISS), and retrieve relevant passages for each question before feeding to the model.

#### 🧩 Architecture:

1. **Use sentence transformers** to embed each paragraph of your FAQ:

from sentence\_transformers import SentenceTransformer

model = SentenceTransformer("all-MiniLM-L6-v2")

faq\_embeddings = model.encode(faq\_chunks)

1. **Store in FAISS (or ChromaDB, Weaviate, etc.)**
2. At runtime:
   * Convert the user question to an embedding
   * Find the most relevant chunk(s)
   * Feed that **as context** into your existing Hugging Face QA pipeline

#### Advantages:

* **No fine-tuning needed**
* **Fast updates** — just re-index the text if content changes
* Works well with **larger unstructured knowledge bases**
* Scales to **hundreds or thousands** of documents