- Large Language Models (LLMs) are advanced neural networks trained on vast amounts of text to understand and generate human-like language. Models like GPT-2 are capable of completing sentences, answering questions, and mimicking conversational patterns. Fine-tuning is the process of adapting a pretrained LLM to a specific domain or task by training it further on a smaller, specialized dataset. This allows the model to produce more relevant and accurate outputs in a targeted context. Fine-tuning bridges the gap between general language understanding and domain-specific expertise.

- How do I make my fine-tuned chatbot generalize to new, unseen questions?

1) expand and diversify the dataset
2) use a hybrid approach : GPT-2 + GPT-3.5 API fallback
3) embed retrieval-augmented generation (RAG) let the bot pull from a text file or knowledge base

- Can't I implement an RNN to predict answers smartly?

RNNs are outdated for natural language tasks. Transformers (like GPT, BERT) have fully replaced RNNs in NLP for good reasons

Feature	RNN	Transformer
handles long context	poorly	excellent
training speed	slow	fast
accuracy	limited	state of the art (the latest)
generalization	manual	emergent with pretraining
pretrained models	rare now	everywhere (HuggingFace, OpenAI)

- best and easiest option

Fine-tune a small Transformer model (DialoGPT) using the Q&A dataset

why this is the best move?

1) much smarter than RNNs
2) ease of use : just 20-30 lines of Hugging Face code
3) already understands language well
4) local training possible
5) scalable with data (more Q&A = smarter bot)
6) open source

DialoGPT : pretrained on conversations

smart, natural, already knows how to chat

- Insights on DialoGPT : identifying the real limitations of using a locally fine-tuned DialoGPT model without full pretraining power

1) answers unseen questions with incorrect made-up answers

I thought DialoGPT would make my model know how to answer basic input like "how are you". I didn't think it would need this kind of input to be included along with its answer in my dataset for it to be able to answer correctly

so why am I not getting those smart responses yet?

what DialoGPT should do by default

is - pretrained on millions of conversations from Reddit

- very good at small talk, greetings, and casual dialogue
- already knows how to answer "Hi" "How are you?" "Thanks" "Goodbye"

this means that including "Hi" and its variations is unnecessary in the dataset - unless DialoGPT was fine-tuned so heavily that it forgot its base knowledge.

in my case: I overwrote its pretraining with only narrow bakery data. my dataset only contains Q&A about cake, cookies, location ... and I haven't provided any general or chitchat examples. the model's personality becomes overly narrow and forgets its conversational instincts (a known issue in fine-tuning called catastrophic forgetting)

how to fix it

option 1: mix small talk into the dataset

option 2: fine-tune less aggressively by using fewer epochs or less bakery-specific data, to preserve general

knowledge

insights after implementing option 1 : continuation of inconsistent answers, delay in response time, and no instruction to stop generation at the end of an answer. it's interpreting "Q:" in the dataset as a continuation signal and keeps generating new Qs

2) takes a while to respond.

I'm using full chat_history_ids, which grows with every turn, eventually hits the 1024 token limit and slows down

how to fix it

only keep the last 1 or 2 turns. this prevents memory overflow and speeds up generation

3) sometimes doesn't respond at all / cuts off randomly

model hits max_length mid_sentence and generation parameters need tuning

- Insights after combining these fixes (truncated history to prevent memory overflow, faster response time, proper generation stop, no repeat Q: Q: Q: hallucinations

hallucinated generation spiraling

same issues as before

- observations after yet another attempt at finding a solution to these limitations : no more repeated hallucinations, faster response time, and the inability to map user input with the correct output as specified in the dataset

that is because I'm training with too little data. my dataset is less than 1000 examples, so the model learns surface-level word patterns, overfits, and loses generalization

fix

I rebuilt the dataset with 3-5 rephrasings per Q&A, converted all Q&A to User: and Bot: format and retrained

- observations : the model got worse from the previous step. it answered only 2 identical input questions exactly like how their answers are written in the dataset. it also shipped answering a few questions.

why does DialoGPT fail to memorize exact Q&A pairs?

DialoGPT is not designed to memorize specific prompt-answer mappings. it's pretrained on dialog flow, not direct factual retrieval

this means
even when fine-tuning on
Q: What are your hours on Friday?
A: We're open from 8 am to 8 pm every Friday.
DialoGPT still might generate
A: We're open from 8 am to 8 pm every Sunday.
because it's trying to chat naturally, not recall facts precisely
solution: the need for a retrieval-augmented system.
to get precise answers from my dataset - especially when the input exactly matches. the best option is to combine
1) intent matching (semantic similarity using sentence transformers): finds the closest match to the user's question from the dataset
2) generate the response using a language model
so I implemented code that does the following
1) loads all Q&A pairs from the dataset
2) embeds the questions using sentence-transformers
3) when a user asks a question
finds the closest matching question

returns the matched answer

4) works instantly and always gives back what's in the dataset

- result : the bot now answers with accuracy and speed. it is able to map to the correct answers based on a one-word only input

Limitations of the code

doesn't perform training

the code doesn't train a model. instead, it uses a pretrained sentence-transformer model (all-MiniLM-L6-v2) to convert sentences into embeddings (numerical vectors). these embeddings capture the semantic meaning of the question. "What are your opening hours?" and "When do you open?" will have similar embeddings

it encodes all the dataset questions. now it has a list of vector representations of the questions. no training is done here - this is just preprocessing.

at runtime, when the user types, it is encoded just like the dataset questions. the script compares the input embedding to every stored question. this tells it

which question is closest - even if the wording is different

why this is not fine-tuning

1) it doesn't modify the transformer model's weights

2) it doesn't create a new model

3) it doesn't use a trainer

the code didn't do any of that. I was just using a frozen, pretrained model just to encode and compare - not to learn

Retrieval vs. Fine-tuning

feature	Retrieval	Fine-tuning
learns from examples	no - searches existing examples	yes - model weights are updated
returns known answers	yes - finds and returns exact matches	not guaranteed
speed to deploy	instant (no training)	requires training time
accuracy	perfect	can vary, prone to hallucination
custom training data	reads it and searches through it	uses it to train the model
model file saved	no new model created	saves a new model checkpoint

- so anyways I got back to the code I had prior and decided to try improving my fine-tuning setup while continuing to use DialoGPT because of how dumb, inconsistent, and inaccurate my model was.

what's improved

1) used LineByLineTextDataset for proper turn-by-turn tokenization
2) configured with 5 training epochs for better learning
3) added learning rate, weight decay, and warmup setups for smoother optimization
4) ensures consistent formatting via pad_token

- observations: the bot was getting progressively worse and dumber

this is called catastrophic forgetting - and it's very common when fine-tuning small models like DialoGPT with small or unbalanced datasets

what's going wrong

1) model is overfitting on small dataset

DialoGPT was pretrained on massive Reddit-style conversations as I mentioned earlier. by fine-tuning it on a narrow, repititive, and small bakery dataset,

I'm making the model forget how to chat properly and spit out gibberish, especially related to repeated keywords like "red velvet", "order", "cake"

2) my dataset is too small

164 lines is way too tiny for DialoGPT. worse, if the same patterns are repeated, it hallucinates fragments, blends phrases, and loops

- I decided to switch from DialoGPT to my original GPT-2 because GPT-2 offers a more stable and controlled foundation for my specific use case. Although the retrieval model exhibited 100% accuracy and smarter responses, only that I couldn't rely on it because it defied the project's objectives regarding fine-tuning. While DialoGPT is optimized for free-flowing, Reddit-style conversations, it struggles with precision, often producing unpredictable and incoherent responses when fine-tuned on small, structured datasets like my bakery Q&A. GPT-2, on the other hand, is a general purpose language model that performs better when direct, factual, and template-based responses are needed. It's more consistent in generating outputs that mirror the structure of the dataset, making

it a better fit for a task that demands clarity, accuracy, and minimal hallucination. By choosing GPT-2, I'm prioritizing reliability, maintainability, and output quality - all of which are critical for delivering a dependable bakery assistant

- The code I ended up with fine-tunes a pretrained GPT-2 language model using my custom bakery dataset. It begins by loading the GPT-2 model and tokenizer from Hugging Face's model hub. The tokenizer is configured to use the end-of-sequence token as the padding token to maintain compatibility during training. The dataset is then loaded using TextDataset, which tokenizes the content and splits it into fixed-length blocks of 128 tokens for model input. A DataCollatorForLanguageModeling is used to prepare the data for casual language modeling, where the model learns to predict the next token in a sequence. Training parameters such as number of epochs, batch size, logging frequency, and model save path are defined using TrainingArguments. The model is fine-tuned using the Trainer class, which handles the training loop, logging,

and checkpointing. After training, the fine-tuned model and tokenizer are saved to the specified output directory for future use. Overall, this code customizes GPT-2 to generate bakery-specific responses based on the patterns learned from the provided dataset

- My overall reflection

Over the course of this project, I explored multiple approaches to building an intelligent, locally hosted chatbot tailored for a bakery setting. After taking the initiative of making my first GPT-2 model smarter, I began by fine-tuning DialoGPT on a custom Q&A dataset, aiming to create a conversational assistant capable of handling customer inquiries. While DialoGPT initially showed potential, I encountered critical limitations - including hallucinated responses, overfitting, and degradation in output quality due to the model's sensitivity to small datasets and its conversational training bias. To address these issues, I resorted to a retrieval-based approach using sentence-transformers, which significantly improved accuracy by semantically matching user input to pre-defined Q&A pairs in my dataset. This method

proved to be fast, reliable, and ideal for delivering factual, context-specific answers - making it a strong candidate for production use. Despite these successes, I ultimately decided to continue development with GPT-2. Compared to DialoGPT, GPT-2 provides more structured and predictable behavior when fine-tuned on clearly formatted prompt-response data. My choice reflects a shift toward consistency and control, with GPT-2 better suited for delivering direct, formatted answers rather than managing open-ended dialogue. In summary, I explored fine-tuning, retrieval-augmented models, and made an informed decision to move forward with GPT-2 - a choice grounded in my goal of building a bakery chatbot that prioritizes stability, clarity, and customer relevance.