1. How does chatgpt work

a) how to build a chatbot - custom knowledge chatbot?

- Scrape the internet for f1 related data
- convert text into vector embeddings [ex: ‘cat’ into number rep that computer understand], then dia buat similarity search [cosine similarity] (distance between this vectors [coordinate in a multidimension])
- each LLM will have different number representing the words, so we cannot compare vector embedding for the word ‘cat’ between open ai and other LLM make
- store in a vector database

[original text -> text splitter into paragraphs -> chunks of texts -> vector embeddings]

RAG: Retrievel Augmented Generation (when you change the output of LLM by provding the model more context alongside a user’s input) - so that model can use its ability to generate text along with extra context to answer questions correctly…

Since LLM is trained on billions of parameters to answer question, we can use RAG to leverage on the power of LLM without needing to train it. Usually LLM have a cutoff date to the training data, so we will need to top the knowledge up. Sometimes the information is not known by the LLM at all.

For RAG chatbot, We need:

1. Free database (nodev20)
2. open ai account
3. basic of vector embeddings
4. Go to datastax - vector database management system - create database (vector database), choose provider. Can create collections in here
5. Set openai account - api login - create api key, use api to communicate with the LLM —> gpt4 and text-embeddings
6. build chatbot

Steps:

1. Prepare the data that we want to use (in a folder), use langchain to load the files into documents (will contain all the metadata)
2. Need to split each document if they’re too long into smaller chunks like paragraphs - use text splitter, they will store the metadata or paths of the chunks
3. to query each chunks, we need to turn this into database (chromadb) —> we will use openai embeddings to generate vector embeddings for each chunks —> save into database
4. turn that into vector database
5. query the database —> to find the chunk in the db that most likely contain the answer of the question [query—> convert to vector embedding —> scan through db to find the chunks of info that is closest to our query embedding]
6. load the chroma db path, get embedding function (openai) —> do similarity search in the database, specify number of result we want to retrieve (can set threshold also)
7. Now that we get relevant data chunks for our query, we can feed this into openai to create response using our data.
8. build prompt template (context would be the chunks retrieve from db after doing similarity search), call LLM model to answer the question (model.predict(prompt))

—> How does chatgpt is trained

- the current LLM for chatgpt is gpt-4/gpt 3.5
- talk about the LLM: a type of neural network based model trained on massive amt of internet text data (have 175 billion parameters with 96 layers in the neural network)
- model use training data to learn statistical pattern between words in language, then utilize this knowlege to predict subsequent word
- the input and output to the model use token (numerical rep of the words)
- the model is trained to predict the next token given a sequence of input tokens.
- So it is able to generate text that is structured in a way that is grammatically correct and semantically similar to the internet data it was trained on.
- to make it safer and give answer that is used in chatgpt (style of a chatbot), the model is further fine-tuned using RLHF (reinforcement training from human feedback). Get feedback from user (mutliple options).
- Then reward modelling: model use this feedback to create reward model (like a guide to understand user preference, higher reward, better answer
- Then we train the model with PPO (proximal policy optimization algorithm): model give the answers based on the reward model. The LLM compares their current answer with a slightly different version, see which gets the highest reward, then keep repeating this step (keeps getting updated user feedback). Iteratively using this PPO.
- How chatgpt UI works?
- user give prompt > chatgpt know the context, it feeds the model all the past conversation everytime new prompt entered
- chatgpt also have primary prompt engineering (pieces of instruction injected before and after user's prompt. This will guide the model for conversational tone.
- prompt is also passed to moderation API to warn or block unsafe content before return to the user.

2. walkthrough 1 one of the models, how to explain the model to the stakeholder

yolov5: NRIC landmarks detection

1. Train the yolov5 model on my custom dataset, which is the NRICs
2. need to label the custom datasets with bounding boxes
3. to train model, we need images by batch download, need to label and annotate the images (folder train data, that have folder images and folder labels), inside folder images have folder train and folder val.inside folder labels (folder train and val)
4. annotate the images using 'makesense.ai'. Drop images in train folder. Insert label names (the landmarks).draw the bounding boxes and select the labels. Then export annotations (in YOLO format).
5. extract the labels folder, put into labels folder in the train folder (class and the coordinate of the bounding box). Do the same for val folder.
6. use colab to train the model. Yolov5 github ultralytics.
7. 1. Setup (model yolov5 will be downloaded in the folder with its weights etc) 2. Upload data into colab, drag the zipped image folder into colab 3. Yolov5 folder (data folder ada coco128.yaml - download) 4. Bukak yaml file, remove the command, tukar filepath of train and val folder, number of class, class name. Save then rename as custome data.yaml, letak balik

8. pegi kat train part. (Letak custom yaml). Run with 3 epochs. Decrease batch size if gpu out of memory. Increase epoch (accuracy dekat map). Tgk result dekat test batch
9. pergi dekat inference detect.py (tukar weight dgn source filepath), boleh tukar conf score threshold

-> script:

1.

Context of the Model

"This prediction is part of one of the checks in our e-KYC solution. Specifically, the model is responsible for detecting key visual landmarks on the Malaysian NRIC—such as the MyKad logo, chip, signature, and bunga raya flower. The objective is simple: if all required landmarks are present and correctly detected, the NRIC image passes this check and moves on to the next verification step."

2.

Model Architecture & Setup

"We're using a YOLOv5 object detection model. I customized the YAML file to define our own classes—i.e., the NRIC landmarks. The model was trained using real customer NRIC images that we've collected, so it's optimized for the kind of images we actually receive in production."

3.

Validation Performance & Observations

"During validation, the model performed well for most classes. However, we identified one specific challenge: detecting the 'bunga raya' landmark. The confusion matrix and some sample predictions showed that this class had lower precision and recall than others."

Strong performance: for classes like chip, photo, and MyKad text

Lower performance: for the bunga raya, due to:

- Limited number of training images with bunga raya clearly visible

- Some misclassifications caused by floral or blurry background elements

4.

Model Improvements for Trustworthiness

"To improve prediction quality and reduce false positives for bunga raya, we took two steps:

- Increased confidence threshold to 0.85 — this filters out low-confidence detections, especially those false positives from background noise.
- Plan to expand high-quality training data — specifically more NRICs with clearly visible bunga raya, which will help improve model precision and recall over time."

5.

Why You Can Trust the Prediction

"Our model is deliberately conservative — it will only pass the check if all 6 required landmarks are detected with high confidence.
So if a customer NRIC passes this step, we can be confident that:

- The image is authentic-looking and complete
- None of the critical elements are missing
- The confidence scores are above our strict threshold

In cases where we're not confident — especially for the bunga raya — the system will not allow the image to pass, reducing the risk of accepting tampered or incomplete IDs."

✔

Summary: Why You Can Trust the Model

Point	Explanation
Tailored for NRICs	Custom-trained YOLOv5 model on real Malaysian NRIC samples
Strict confidence rules	Only landmarks detected with ≥85% confidence are accepted
Failsafe check	All 6 landmarks must be present — partial detections don't pass
Continuous improvement	Ongoing data collection to boost weaker class (bunga raya)
Transparent metrics	Confusion matrix and validation samples shown to monitor model performance

Optional (If They Push Further)

If the stakeholder still seems doubtful, you can add:

> "Would you like me to demo some borderline cases and show the bounding box confidence scores? I can walk through examples where detections passed vs failed, and how the threshold ensures robustness."

Your Original Explanation – Feedback

Here's your draft with my technical corrections and improvements:

Original:

> “I use yolov5 model, where i changed the exisiting classed in yaml and trained the model on the nric datasets. So based on the result on validation set, i see that the model is not able to identify bunga raya well, as you can see on a few validation samples and the confusion matrix…”

Corrections:

- “Changed the existing classes in the YAML file”
- “Trained the model on real NRIC images collected from customers”
- “Model struggles to identify ‘bunga raya’ due to background interference and limited training samples”
- “Some images had two ‘bunga raya’ predictions, but one was low confidence and due to background misdetection”
- “To address this, we raised the confidence threshold and plan to add more annotated examples of bunga raya”

If you want, I can help build slides or a live demo script next.