Follow-up Answers to Professor's Questions

Following our presentation, here are the answers to the questions you raised:

Srinu Babu Rai:

1. How is the NLP engine related to your sentiment analysis?

Answer :

BY doing the reverse engineering with project code

I found that in my Project, when a user types a question, the model first checks the tone of that question, if it sounds negative or about a crisis, it stops them going further and immediately provide info for human help.

If the question isn't negative, then it passes through the NLP engine, which reads the question, searches the knowledge base and gives a helpful answer and even suggests follow-up questions
So the sentimental analysis decides whether the NLP engine should work on the question at all or whether to forward for supporting staff.
here I am attaching the reference from the Chatgpt

1️⃣ Sentiment analysis happens first

File: `controller.py`

- **Line ~76–93 →** `_load_sentiment_pipeline()`

  python Copy Edit

  ```python
  self.sentiment_pipeline = pipeline(
      "sentiment-analysis",
      model="cardiffnlp/twitter-roberta-base-sentiment-latest"
  )
  ```

  This loads the **RoBERTa** sentiment model.

- **Line ~95–130 →** `check_escalation_needed(text)`

  python Copy Edit

  ```python
  results = self.sentiment_pipeline(text)
  if results[0]['label'] == 'NEGATIVE' and results[0]['score'] >= 0.80:
      return True
  ```

  If a message is strongly negative (≥ 80% confidence), the function returns `True` (escalation needed).

2️⃣ NLP engine runs only if sentiment check is OK

File: `controller.py`

- **Line ~132–170 →** `ask_and_respond(user_text)`

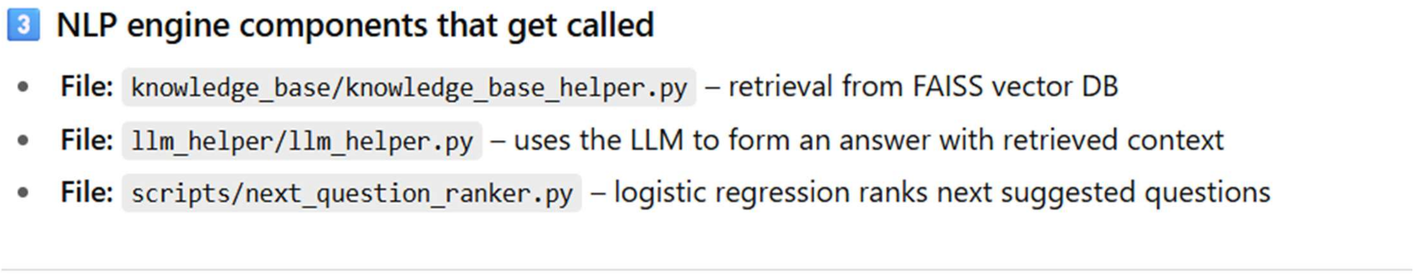
  python Copy Edit

  ```python
  if self.check_escalation_needed(user_text):
      return {"answer": "Escalated to human support"}

  answer = self.vector_db_helper.handle_query(user_text)
  final_answer = self.llm_helper.ask_llm(answer)
  ```

  Here:

  - First, `check_escalation_needed()` is called.
  - If escalation is **not** needed, it passes the text to:
    - `vector_db_helper.handle_query(...)` → **retrieves matching info** from the knowledge base
    - `llm_helper.ask_llm(...)` → **creates the final answer**
    - Then `next_question_ranker` suggests follow-up questions.



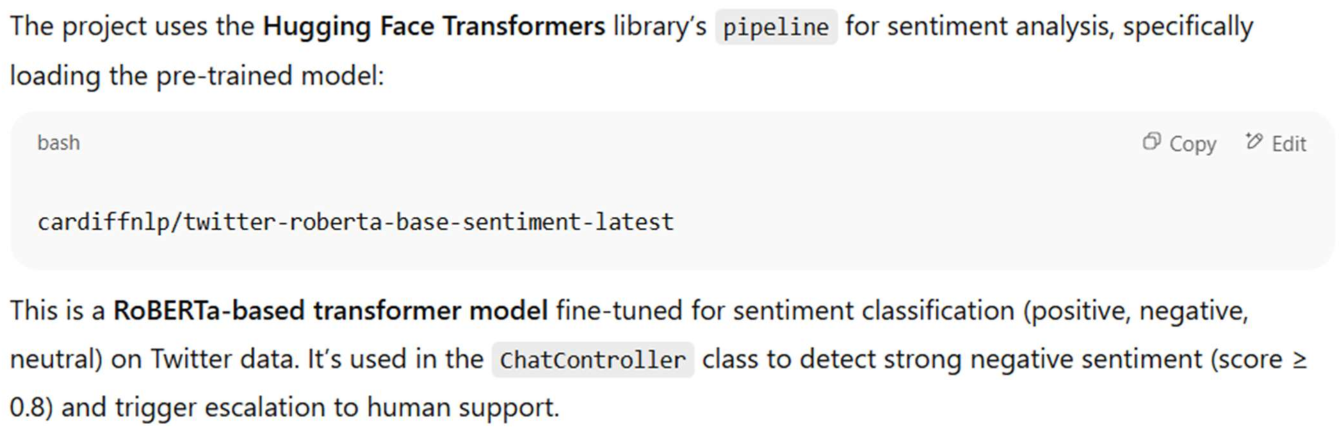
3️⃣ NLP engine components that get called

- **File:** `knowledge_base/knowledge_base_helper.py` – retrieval from FAISS vector DB
- **File:** `llm_helper/llm_helper.py` – uses the LLM to form an answer with retrieved context
- **File:** `scripts/next_question_ranker.py` – logistic regression ranks next suggested questions

2. What algorithm is used for sentiment analysis?

Answer :

In my project It is using a RoBERTa-based AI model from Hugging Face, trained to detect positive, negative, or neutral tone in tweets. If it finds a very negative tone (score ≥ 0.8), it skips the chatbot and sends the case to a human for help.



The project uses the **Hugging Face Transformers** library's `pipeline` for sentiment analysis, specifically loading the pre-trained model:

```bash
cardiffnlp/twitter-roberta-base-sentiment-latest
```

This is a **RoBERTa-based transformer model** fine-tuned for sentiment classification (positive, negative, neutral) on Twitter data. It's used in the `ChatController` class to detect strong negative sentiment (score ≥ 0.8) and trigger escalation to human support.

Fasalu:

3. Why are epochs used in Logistic Regression and what do they do? How does it work?

Answer :

We train Logistic Regression through Stochastic Gradient Descent (SGD), which is iterative. One pass over the train set is called an epoch. Multiple epochs allow the model to learn gradually, reduce error step by step until the performance cannot be further improved. In our case we have done 60 epochs and it took like maximum 20 epochs to get its maximum accuracy

---

Christo:

4. What are the independent and dependent variables in your model?

Answer : This is the feedback from chat GPT.

Prompt: We have a set of questions and corresponding answers in a dataset. we have also vectorized this dataset. I have done a logistic regression model for this chatbot. What we are trying to predict is the next questions which is related to the current asked question. In this regression model what will be the in depended and depended variable?

Response:

In your case:

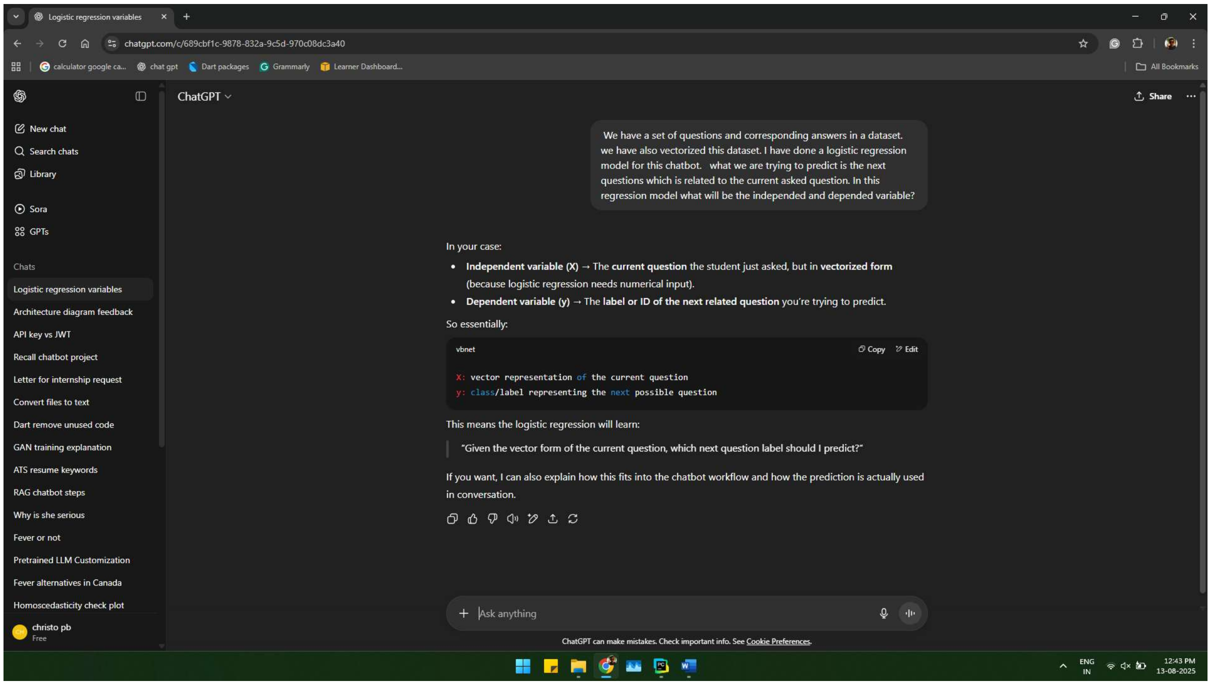
- **Independent variable (X)** → The **current question** the student just asked, but in **vectorized form** (because logistic regression needs numerical input).
- **Dependent variable (y)** → The **label or ID of the next related question** you're trying to predict.

X: vector representation of the current question

y: class/label representing the next possible question

This means the logistic regression will learn:

"Given the vector form of the current question, which next question label should I predict?"



We have a set of questions and corresponding answers in a dataset. we have also vectorized this dataset. I have done a logistic regression model for this chatbot.   what we are trying to predict is the next questions which is related to the current asked question. In this regression model what will be the independed and depended variable?

In your case:

- **Independent variable (X)** → The **current question** the student just asked, but in **vectorized form** (because logistic regression needs numerical input).
- **Dependent variable (y)** → The **label or ID of the next related question** you're trying to predict.

So essentially:

```vbnet
X: vector representation of the current question
y: class/label representing the next possible question
```

This means the logistic regression will learn:

> "Given the vector form of the current question, which next question label should I predict?"

If you want, I can also explain how this fits into the chatbot workflow and how the prediction is actually used in conversation.

ChatGPT can make mistakes. Check important info. See Cookie Preferences.

The next prompt I have to the LLM was to check my understanding about its working.

Prompt: So let me explain its working. you correct me if I am working. The regression model is actually comparing current question with all other question and it says whether is related or not. and this happens till the end of all question we have.

Response: Yes — you're basically right, but let's tighten up the picture so it's exactly what's happening in logistic regression.

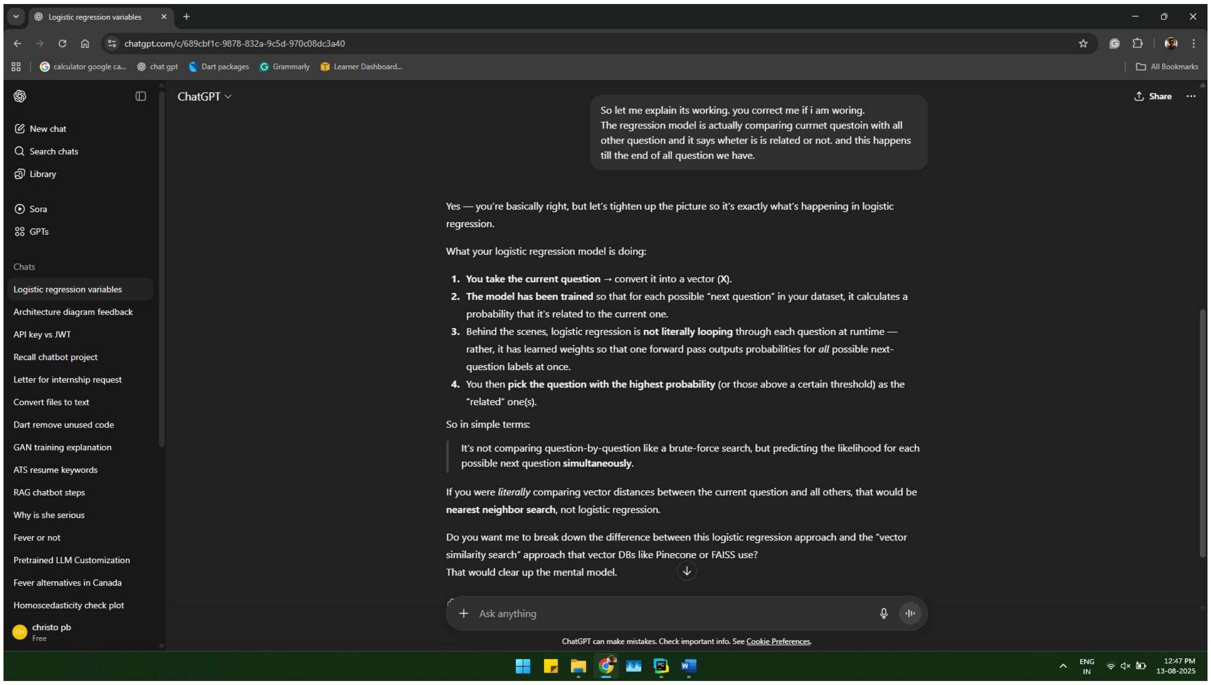
What your logistic regression model is doing:

1. **You take the current question** → convert it into a vector (**X**).
2. **The model has been trained** so that for each possible "next question" in your dataset, it calculates a probability that it's related to the current one.
3. Behind the scenes, logistic regression is **not literally looping** through each question at runtime — rather, it has learned weights so that one forward pass outputs probabilities for *all* possible next-question labels at once.
4. You then **pick the question with the highest probability** (or those above a certain threshold) as the "related" one(s).

So in simple terms:

It's not comparing question-by-question like a brute-force search, but predicting the likelihood for each possible next question **simultaneously**.

If you were *literally* comparing vector distances between the current question and all others, that would be **nearest neighbor search**, not logistic regression.



So let me explain its working. you correct me if i am woring.
The regression model is actually comparing currnet questoin with all other question and it says wheter is is related or not. and this happens till the end of all question we have.

Yes — you're basically right, but let's tighten up the picture so it's exactly what's happening in logistic regression.

What your logistic regression model is doing:

1. **You take the current question** → convert it into a vector (**X**).
2. **The model has been trained** so that for each possible "next question" in your dataset, it calculates a probability that it's related to the current one.
3. Behind the scenes, logistic regression is **not literally looping** through each question at runtime — rather, it has learned weights so that one forward pass outputs probabilities for *all* possible next-question labels at once.
4. You then **pick the question with the highest probability** (or those above a certain threshold) as the "related" one(s).

So in simple terms:

> It's not comparing question-by-question like a brute-force search, but predicting the likelihood for each possible next question **simultaneously**.

If you were *literally* comparing vector distances between the current question and all others, that would be **nearest neighbor search**, not logistic regression.

Do you want me to break down the difference between this logistic regression approach and the "vector similarity search" approach that vector DBs like Pinecone or FAISS use?
That would clear up the mental model.