RIPHAH INTERNATIONAL UNIVERSITY, ISLAMABAD

BSAI-3

(Fall 2025)

Faculty of Computing

Subject: AI

Members

Name: Muhammad Afaq Shoaib, Saghir Ali

Sap ID: 66285, 67005

Project Documentation: Riphah University Intelligent Chatbot

1. Project Overview

This project implements an intelligent Question-Answering (QA) chatbot designed for Riphah International University. It utilizes a hybrid approach combining **semantic search** (Deep Learning), **fuzzy matching** (String Similarity), and **machine learning classification** (Voting Classifier) to provide accurate answers to administrative and academic inquiries.

2. Technical Workflow

The system operates through a linear pipeline:

1. **Data Ingestion:** Loads and merges JSON datasets containing Question-Answer pairs.
2. **Preprocessing:** Cleans raw text using spelling correction, normalization, and lemmatization.
3. **Feature Engineering:**
    - **TF-IDF:** Converts text to sparse vectors (Word-level & Character-level).
    - **Embeddings:** Generates dense vector representations using a Sentence Transformer model.
4. **Model Training:** Trains an ensemble Voting Classifier (Naive Bayes + Logistic Regression + SVM) on the processed data.
5. **Inference Engine (Hybrid Logic):**
    - Checks **Semantic Similarity** (Understanding meaning).
    - Checks **Fuzzy Match** (Handling typos).
    - Checks **Classifier Prediction** (Intent detection).
    - Selects the best answer based on confidence thresholds.

6. **User Interface:** A Gradio web app accepts user input and displays the selected answer.

3. Libraries & Dependencies

The project relies on the following key Python libraries:

Library	Purpose
Pandas / NumPy	Data manipulation and numerical operations.
Scikit-learn	Machine Learning algorithms (TF-IDF, Naive Bayes, SVM, Logistic Regression, Voting Classifier).
Spacy	Natural Language Processing (Lemmatization).
Sentence-Transformers	Deep Learning for semantic text embeddings (multi-qa-mpnet-base-dot-v1).
Rapidfuzz	Fast string matching (Fuzzy logic).
Pyspellchecker	Automatic spelling correction.
Imbalanced-learn	Handling class imbalance via SMOTE.
Gradio	Web interface creation.

4. Key Functions

A. Data & Preprocessing

load_dataset_safe(url)

- **Input:** URL string pointing to a raw JSON file.
- **Output:** Pandas DataFrame.
- **Logic:** Fetches data via HTTP, parses diverse JSON structures (dict or list), and normalizes nested fields like entities.

normalize_fast(text)

- **Input:** Raw text string.
- **Output:** Cleaned, lemmatized string.
- **Logic:**
    1. Converts to lowercase.
    2. Removes non-alphanumeric characters.
    3. Expands synonyms (e.g., "cs" $\to$ "computer science").

4. Corrects spelling using pyspellchecker.
5. Lemmatizes tokens using spaCy to reduce words to their root form.

B. Inference & Logic

fuzzy_match_answer(q_proc)

- **Input:** Preprocessed query string.
- **Output:** Tuple (answer, score).
- **Logic:** Uses rapidfuzz.process.extractOne to find the most similar question in the dataset based on token sort ratio.

semantic_rerank(q_proc, top_k=5)

- **Input:** Preprocessed query string.
- **Output:** Best index, similarity score, and answer text.
- **Logic:** Encodes the input query into a vector and calculates Cosine Similarity against all pre-computed dataset embeddings.

answer_hybrid(raw_query)

- **Input:** User's raw question string.
- **Output:** Final Answer string (and optional debug dictionary).
- **Logic (Decision Tree):**
    1. **Semantic Check:** If similarity score > 0.66, return semantic match.
    2. **Fuzzy Check:** If fuzzy score > 72, return fuzzy match.
    3. **ML Check:** If classifier confidence > 0.60, return answer from predicted category.
    4. **Fallback:** If semantic confidence is moderate, return semantic match; otherwise, return a "Don't know" message.

5. Inputs and Outputs

System Input

- **User Query:** A natural language string (e.g., *"What is the fee for BSCS?"* or *"When are exams?"*).

System Output

- **Primary Output:** A text string containing the specific answer retrieved from the dataset.
- **Debug Output (Optional):** A dictionary containing:

- nb_proba: Classifier confidence score.
- sem_score: Semantic similarity score.
- fuzz_score: String matching score.
- reason: Which logic path was chosen (e.g., 'semantic', 'fuzzy', 'nb_cat_match').

6. Execution Steps

To run this project from a fresh environment (e.g., Google Colab):

1. **Install Libraries:** Run the pip install command for scikit-learn, gradio, rapidfuzz, pyspellchecker, sentence-transformers, spacy, and imbalanced-learn.
   - *Command:* !pip install -q ...
2. **Download Resources:** Download the English language model for Spacy.
   - *Command:* !python -m spacy download en_core_web_sm
3. **Load & Process Data:** Execute the cells that fetch the JSON data from GitHub and apply normalize_fast to create the question_proc column.
4. **Train Models:** Run the training cells to generate TF-IDF matrices and train the VotingClassifier. This step also saves models to disk (.joblib files).
5. **Generate Embeddings:** Run the cell that loads multi-qa-mpnet-base-dot-v1 and encodes the entire dataset.
6. **Launch Interface:** Execute the final cell containing the gradio code.
   - *Action:* Click the public URL generated (e.g., https://xxxx.gradio.live) to interact with the bot.
7. **GitHub Link:**

https://github.com/saghiralich/AI_Semester_Project_Saghir_and_Afaq