Yarmouk Assistant Bot

Abstract

Yarmouk Assistant Bot is an intelligent conversational agent designed to help students and staff at Yarmouk University by answering university-related queries in natural language. The chatbot processes queries in both Arabic and English, automatically correcting typos and interpreting free-form questions to identify the user's intent. It is trained on an internally constructed FAQ dataset (4,917 questions across 236 intents) and uses a simple feed-forward neural network to classify intents, backed by a fallback to a large-language model (DeepSeek API) To overcome the limitations of fixed-response systems by generating natural, context-aware, and professional replies, especially in complex or open-ended scenarios. The system achieves high classification accuracy (≈92%) and provides predefined answers for known intents, enriched with an LLM-generated elaboration. Visual interfaces are provided via both a Telegram bot and a web-based chat UI. Key benefits include bilingual support, spelling-error tolerance, and a friendly interactive interface.

Introduction

Modern universities increasingly employ chatbots to provide timely information and guidance to students and faculty. A well-designed chatbot can answer routine questions instantly, improving user experience and reducing staff workload. In higher education contexts, student expectations for quick responses are high – for example, 72% of users expect replies within one hour – which chatbots can satisfy by responding instantly. Yarmouk Assistant Bot was developed to serve Yarmouk University's community, offering information about courses, schedules, faculty, and facilities. Crucially, it supports both Arabic and English, understanding questions phrased in either language (often with transliterated names) and replying in the same language. Prior work on educational chatbots has shown that bilingual support greatly improves user acceptance, and our system follows this insight.

Chatbots operate by classifying a user's input with a predefined intent and producing a canned or computed response. Unlike generative dialogue systems (which require massive training data to produce fluent free-form responses, our design is primarily retrieval-based: we match the user's query to one of the known intents in our FAQ and return the associated answer. For questions outside our intent set (low confidence), we fall back on a semantic search / LLM answer generation approach, blending rule-based and generative methods. This hybrid design leverages the reliability of fixed answers for common queries while still handling novel questions. The rest of the report details our data sources, processing pipeline, model design, integration, user interfaces, and evaluation.

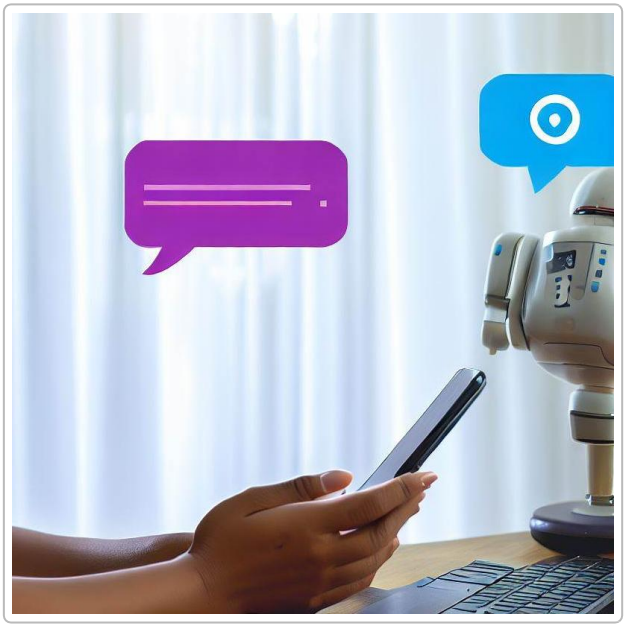

Figure 1: Example of a user interacting with the chatbot on a smartphone. The interface allows students to type free-form questions and receive responses from the assistant.

Methodology

- **Datasets:** We created a **FAQ dataset** specific to Yarmouk University by web-scraping the university website and manually collecting question-answer pairs. This resulted in 4,917 example questions covering 236 distinct intents (e.g. "library hours," "study plans," "professor information", etc.). For each intent, we curated a predefined answer (the **Response dataset**). Building a high-quality data set is challenging; in Arabic-language chatbots this scarcity of structured data is well-known. Our approach was to carefully collect and label each question manually, following best practices. For reproducibility, all responses were written by team members knowledgeable about Yarmouk's academic programs and policies.

- **Preprocessing (Arabic):** We anticipate users typing queries in colloquial or Modern Standard Arabic. To reuse English NLP tools, each Arabic query is first translated to English (using an online translation API). Then we apply a standard text cleaning pipeline: convert to lowercase, fix punctuation spacing via regex (ensuring punctuation marks attach correctly to words), and normalize any mis-encoded characters. The text is tokenized into words, and we remove Arabic stop words and punctuation tokens. We apply part-of-speech tagging and lemmatization (using spaCy/Python libraries) to reduce each word to its root form. Finally, we collapse extra whitespace. This pipeline helps normalize different word forms and prepares the text as input to the classifier. (Arabic's rich morphology makes this step important.)

- **Preprocessing (English):** For English queries, we skip the translation and instead start by correcting spelling mistakes to improve intent matching. We implemented a custom spell corrector (based on edit distance) that ignores known professor names and other key entities, so it does not auto-correct proper nouns erroneously. After spelling correction, the same steps are followed: lowercase, fix punctuation, tokenize, remove stop words, part-of-speech tag, and lemmatize. Finally, we apply a specialized name matcher: every token is compared (via levenshtein distance) to a list of known professor names and technical terms, and close matches (edit distance $\leq$ 2) are corrected. This ensures that queries mentioning faculty or courses by name will be recognized correctly (for example, "Dr. Ale Youssef" corrected to "Dr. Ali Yousef"). Handling

misspellings are important in practice, especially in Arabic, since users often type quickly or phonetically. Prior research notes that user inputs frequently contain errors and nonstandard spelling, posing a major challenge for Arabic chatbots; our pipeline addresses this by correction and normalization steps.

- **Feature Encoding:** After preprocessing, each query (now a list of lemmatized English tokens) is vectorized into numerical features. We use a TF-IDF representation (term-frequency inverse-document-frequency) over the entire training corpus. TF-IDF is effective at capturing the importance of words (giving more weight to rare but discriminative terms). We chose TF-IDF over simple bag-of-words so that unique keywords (like specific location names or course codes) stand out. (Using TF-IDF helped preserve key proper nouns in a high-dimensional sparse vector.) Word embeddings were considered, but because our dataset is relatively small and domain- specific, training robust embeddings was impractical; hence we rely on TF-IDF features for classification.

- **Model Training:** We encode each intent label as a one-hot vector and split our data into 80% training and 20% test sets. Various classifiers were evaluated. A feed-forward neural network (FNN) with three hidden layers (each with ReLU activations and dropout for regularization) achieved the best results. This model, trained on the TF-IDF vectors, reached about 92% accuracy and loss $\approx$ 0.27 on the test set. Classical methods like Support Vector Machines and Naive Bayes were also tried: SVM gave reasonable accuracy, but because we have 236 classes, it often returned very low confidence scores and was sensitive to parameter choices. Naive Bayes performed similarly, but neither matched the FNN in stability. It is not uncommon for Naive Bayes to prefer raw count vectors over TF-IDF in some cases, yet our TF-IDF FNN outperformed NB in practice. We also considered Recurrent Neural Networks (RNN/LSTM), but training these effectively typically requires pre-trained word embeddings and much larger datasets. Our dataset ($\approx$5k examples) is too small to learn a good sequence model, and no off-the-shelf embedding fits the domain well. Moreover, generative sequence models require huge data to avoid erratic outputs. Thus, a simpler FNN on TF-IDF features was the pragmatic choice.

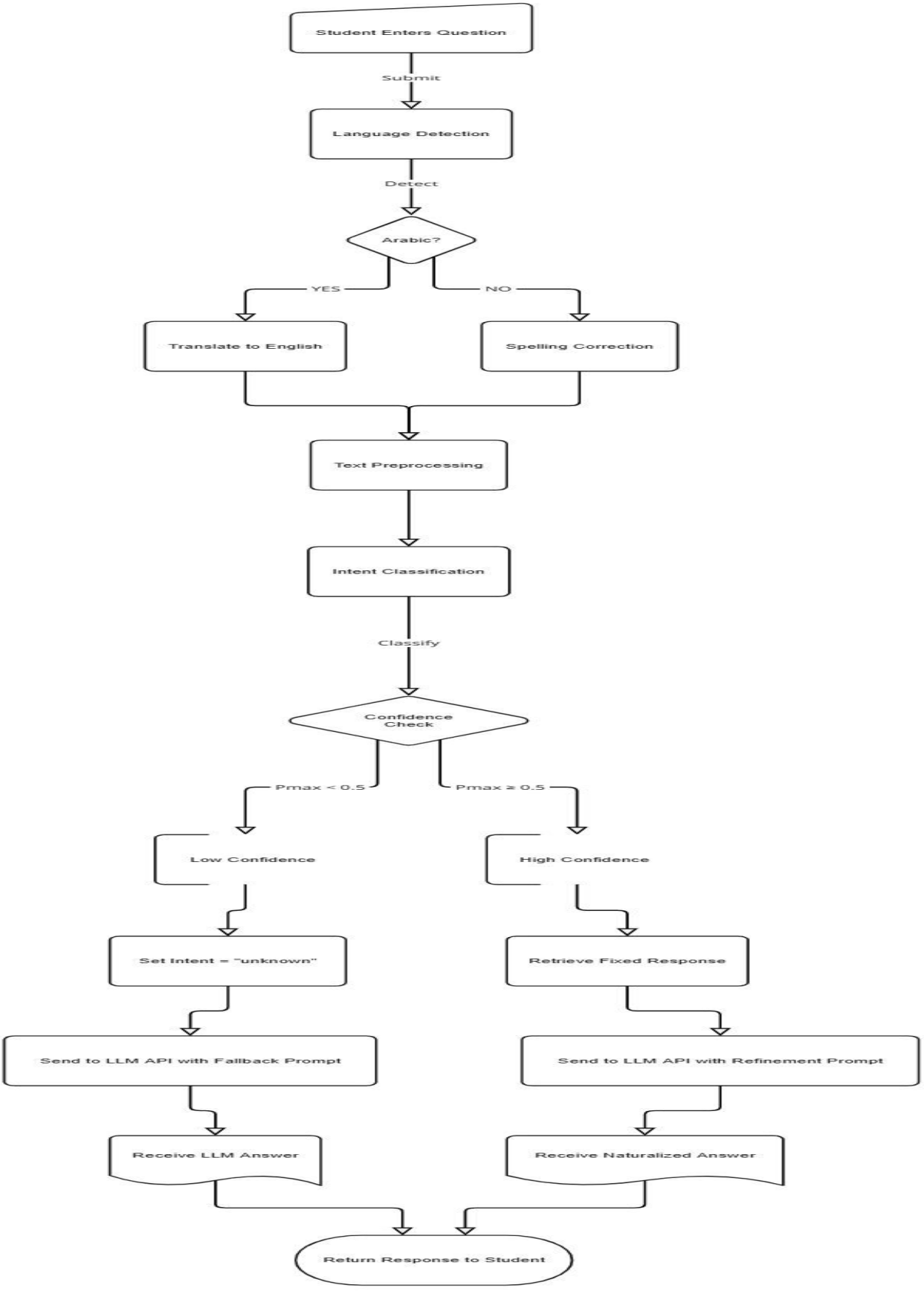

Figure *2:* Flowchart of the chatbot pipeline used by the Yarmouk University Assistant, showing the steps from student question input to final response generation.

Integration Pipeline

The complete question-answering pipeline combines several steps in sequence:

1. **Language Detection:** The system first detects if the input is Arabic or English (using a simple language-detection library).
2. **Name Correction:** If in English, apply the name-matching step described above; if in Arabic, translate first then apply name matching on the English output.
3. **Preprocessing:** Perform all text cleaning (lowercasing, punctuation fix, tokenization, stopword removal, POS-tagging, lemmatization).
4. **Intent Prediction:** Transform the cleaned text into a TF-IDF vector and feed it to the FNN intent classifier. The model outputs a probability distribution over the 236 intents.
5. **Confidence Check:** If the top intent's probability is below 0.5, we treat the query as "unknown." In that case, we do *not* trust the fixed-intent answer. Instead, the question is forwarded to a DeepSeek API, which essentially invokes a search or LLM prompt with context (e.g. "DeepSeek, please answer: <original query>"). This acts like a fallback handler – conceptually similar to Rasa's fallback concept [8]. If confidence is ≥ 0.5, we assume the intent match is valid.
6. **Response Generation:** For high-confidence intents, we return the predefined answer from our Response dataset. Optionally, we can further enrich this answer by running the question (and the fixed answer) through a small language model prompt to make the phrasing more conversational. For low-confidence queries answered by DeepSeek, the response is generated directly from that API.

This hybrid approach ensures that well-covered questions get fast, accurate replies, while unusual questions are still answered by "searching" in the knowledge base.

Implementation

The system is implemented in Python and deployed with two user-facing interfaces:

- **Telegram Bot:** We created a Telegram chatbot named "Yarmouk Assistant" via the BotFather registration tool. The bot token is used with the python-telegram-bot library to connect to Telegram's API. Incoming messages trigger our pipeline: each message is processed server-side, and the reply is sent back via the Telegram API. The interface supports rich text and emojis.

Figure *3:* Sample conversation with the Yarmouk Assistant Bot in the Telegram mobile app. The user first asks for the electronic student information system website in Arabic, and the bot replies with a clickable link to the system. Then, the user requests the Computer Science study plan in English, and the bot responds instantly with the program name, study plan year, and a downloadable PDF link for the 2024 Bachelor's curriculum. The conversation demonstrates bilingual interaction and quick access to official university documents through Telegram.

- **Web Chat Interface:** We also built a web-based chat UI using Flask. The Flask server hosts HTML templates with a chat window. User messages are sent to the backend via an AJAX call, processed by the same NLP pipeline, and the response is displayed in the browser. The web chat mimics a typical messenger layout, ensuring accessibility for users who prefer the desktop or browser interface. (This web service can be run locally or hosted on a campus server).

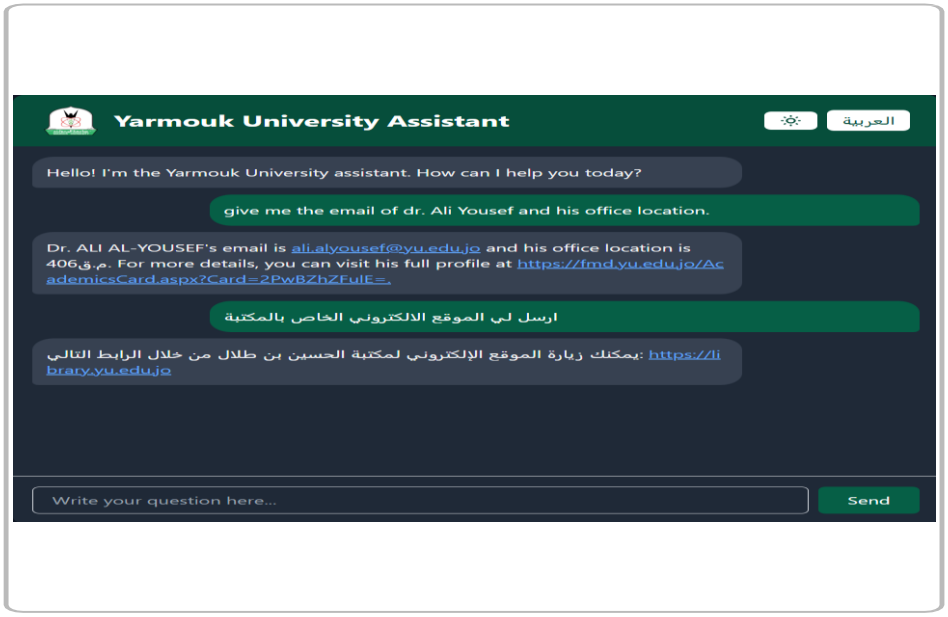

Figure 4: Sample interaction with the Yarmouk University Assistant web chatbot. The user requests the email and office location of Dr. Ali Yousef in English, and the bot responds with his email address, office number, and a link to his academic profile. Then, the user switches to Arabic, asking for the library's electronic website, and the bot instantly replies with a clickable URL to the Hussein Bin Talal Library website. The interface supports bilingual queries and provides official contact details and links promptly.

Our implementation emphasizes simplicity and user-friendliness. Both interfaces use the same core logic; only the delivery channel differs.

Evaluation

The primary evaluation is the classification accuracy on our test set: the FNN model achieved **≈92% accuracy** with a cross-entropy loss around 0.27. This indicates that, for 9 out of 10 test queries, the system correctly identified the intent. The large number of classes (236) makes this a nontrivial result. In practice, user satisfaction is also important: we plan to pilot the bot with actual students. (Prior work on similar educational chatbots found high student satisfaction when bilingual support was provided). For queries that fall outside the known intents, our fallback to DeepSeek has not been formally evaluated

here, but conceptually it is like retrieval-augmented response systems that have shown promise in other domains. Future work includes logging real user queries to measure how often intents are matched vs. handled by fallback.

Conclusion

The Yarmouk Assistant Bot provides an AI-powered, bilingual conversational interface for Yarmouk University information. By combining a traditional intent-classification model with a generative fallback, it ensures broad coverage: known questions get precise canned answers, while novel questions still receive informative responses. Key advantages of the system include:

- Natural language support: Users can phrase questions in everyday language without following rigid menus.
- Bilingual functionality: The bot understands both Arabic and English queries and responds in the same language, increasing accessibility.
- Spelling and transliteration correction: Misspellings and variant spellings (especially of names) are auto corrected, making the system robust to typos.
- Accessible UIs: Available on popular platforms (Telegram and web), the bot is easy and familiar to use.

In summary, the project meets its goals of a friendly, accessible campus assistant. Future improvements could include expanding the intent set, refining responses with richer prompts, and integrating campus databases for dynamic information (e.g. live course schedules).