Project Report: Natural Language Processing for Student Support

Topic: Development of an Student Chatbot for Warsaw University of Technology (WUT)

1. Executive Summary

The objective of this project was to design, implement, and evaluate a Proof of Concept (PoC) for an automated chatbot system. This system is designed to assist international and local students at the Warsaw University of Technology (WUT) by answering frequently asked questions (FAQs) about admissions, scholarships, exams, and campus life.

To ensure the best balance between accuracy, cost, and user experience, we implemented and compared two different technical approaches:

1. **A Baseline Model:** A classical Information Retrieval system using **TF-IDF** (Term Frequency-Inverse Document Frequency).
2. **A State-of-the-Art (SOTA) Model:** A **Retrieval-Augmented Generation (RAG)** system utilizing OpenAI's **GPT-4o-mini**.

2. Problem Statement and Motivation

Administrative departments at universities often face a high volume of repetitive queries, especially during admission seasons. Students frequently ask about deadlines, required documents, and tuition fees. Manual responses are time-consuming and can lead to delays.

Our goal was to automate this process using Natural Language Processing (NLP). The solution needed to be:

- **Accurate:** Providing official, verified information.
- **Cost-effective:** Sustainable for a university budget.
- **Reproducible:** Easy for other developers to set up and deploy.

3. State of the Art (SOTA) Review

Before developing our solution, we analyzed the evolution of question-answering systems to justify our choice of architecture.

3.1. Evolution of Chatbots

- **Generation 1: Rule-Based Systems.** These rely on exact keyword matching (e.g., if user types "fee", output "Tuition info"). They are rigid and fail when users use synonyms (e.g., "cost" instead of "fee").
- **Generation 2: Semantic Search (Vector Embeddings).** Systems using models like BERT transform text into mathematical vectors. This allows the bot to understand that "cost" and "fee" have similar meanings. However, they cannot generate natural, conversational answers; they simply return a document link.
- **Generation 3: Retrieval-Augmented Generation (RAG).** This is the current SOTA standard. It combines a search engine (Retriever) with a Generative AI (LLM). The retriever finds the correct data, and the LLM writes a human-like response based *only* on that data.

Decision: We chose the **RAG architecture**. This addresses the main limitation of standard LLMs (hallucinations) by forcing the AI to strictly use our provided university data.

4. Exploratory Data Analysis (EDA)

The foundation of any ML project is data. We performed a comprehensive analysis of the provided dataset (wut_faq_dataset_bilingual.csv) using the notebook 01_eda_notebook.py

4.1. Data Source and Origin

To create a realistic and representative dataset for this Proof of Concept, we researched common administrative FAQ patterns from existing chatbot projects at other universities. We extracted generic question templates (e.g., "How do I apply?", "What are the fees?") and adapted them specifically for the Warsaw University of Technology context. While the question structures are based on common academic inquiries, every answer was manually curated and populated with official, verified information directly from WUT administrative sources to ensure accuracy.

4.2. Dataset Structure

The dataset consists of **44 Question-Answer pairs** with the following key features:

- question: The user query (e.g., "What is the deadline?").
- answer: The official administrative response.
- category: The topic (Admissions, Scholarships, Exams, etc.).
- source_url: A link to the official WUT webpage for verification.

4.3. Statistical Insights

We visualized the data to understand its distribution:

- **Category Balance:** The dataset is well-distributed across 6 categories. "Admissions" (9 items) and "Scholarships" (8 items) are the largest categories, which aligns with the most common student inquiries.
- **Language Distribution:** The data is bilingual but predominantly English (86.4%), with a smaller subset in Polish (13.6%) to demonstrate multilingual capabilities.
- **Content Length:** The average answer length is **371 characters** (approx. 55 words). This indicates that the answers are concise, which is ideal for a chatbot interface where long blocks of text are discouraged.

4.4. Data Quality

We performed a rigorous quality check:

- **Missing Values:** No null values were found in critical columns.
- **Duplicates:** No duplicate questions were detected.
- **Verification:** 100% of answers include a valid source_url, ensuring traceability.

5. Machine Learning Implementation

5.1. Feature Engineering (Preprocessing)

We used the scikit-learn library for text processing. Before feeding data into our models, we applied a **TF-IDF Vectorizer**.

- **N-grams:** We configured the vectorizer to use (1, 2) n-grams. This means the model considers not just single words ("social"), but also pairs ("social scholarship"). This is crucial for distinguishing between different types of scholarships.
- **Vocabulary Pruning:** We limited the vocabulary to the top 1,000 features and removed standard English "stop words" (like "the", "and") to reduce noise.

5.2. Approach 1: The TF-IDF Baseline

We built a modular Python class, FAQChatbot, located in tfidf_chatbot.py.

- **Mechanism:** When a user asks a question, the system calculates the **Cosine Similarity** between the user's vector and every FAQ question in our database.
- **Result:** It returns the answer associated with the mathematically closest question.
- **Limitations:** As seen in our tests, this model struggles with phrasing differences. For example, if a student asks "How does Erasmus work?", the model might fail if the database only contains "How do I apply for Erasmus?".

5.3. Approach 2: The RAG Solution (Final PoC)

Our advanced solution, implemented in rag.ipynb, integrates OpenAI's API.

Architecture:

1. **Retrieval Step:** We re-use the lightweight TF-IDF retriever to find the top 3 most relevant FAQ entries from our CSV.
2. **Context Construction:** We dynamically build a prompt that includes these 3 retrieved FAQs.
3. **Generation Step (The "Brain"):** We send this prompt to **GPT-4o-mini**. We use a specific **System Prompt** to control behavior:*"You are an administrative assistant for Warsaw University of Technology. You must answer student questions using ONLY the information in the FAQ context below."*
4. **Output:** The LLM synthesizes the retrieved facts into a polite, clear response.

Results: In our interactive tests, the RAG model successfully answered vague queries like "Where is the schedule?" by synthesizing information from the "Campus Life" category, whereas the baseline model struggled.

6. Reproducibility Guide

A key requirement was ensuring this PoC is reproducible. We have organized the code to allow any developer to run it immediately.

Repository Structure:

- wut_faq_dataset_bilingual.csv: The clean data source.
- requirements.txt: Lists exact library versions (e.g., pandas==2.1.4, scikit-learn==1.3.2) to prevent compatibility errors.
- tfidf_chatbot.py: A reusable module containing the chatbot logic.
- rag.ipynb: The main notebook to run the advanced model.

Steps to Reproduce:

1. Install dependencies: pip install -r requirements.txt
2. Set up the API key: export OPENAI_API_KEY="your-key"
3. Run rag.ipynb to see the retrieval process and cost analysis in real-time.

7. Conclusions and Future Work

This project successfully demonstrated that a **RAG architecture** significantly outperforms traditional keyword-based search for university administration. It provides the flexibility of a human conversation with the strict accuracy of a database.

Future Improvements:

- **Vector Database:** For larger datasets (200+ FAQs), replace TF-IDF with a vector database like FAISS or ChromaDB for better semantic retrieval.
- **Deployment:** Wrap the rag_answer function in a FastAPI web server to connect it to the university's website.