

# **Machine Learning Model For Academic Purposes**

Project Report

Submitted in the partial fulfillment of the  
requirements for the award of the degree of

## **BACHELOR OF TECHNOLOGY**

**In**

## **DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING**

**By**

**Laasika Reddy Anuga- 2320040069**

**Gamana Chirumamilla- 2320040075**

**Gayathri Yerra- 2320040080**

Under the Esteemed Guidance of

**LECTURE NAME**

**Divya Siripuri**



## **Koneru Lakshmaiah Education Foundation**

(Deemed to be University estd. u/s. 3 of the UGC Act, 1956)

Off-Campus: Bachupally-Gandimaisamma Road, Bowrampet, Hyderabad, Telangana - 500 043.

Phone No: 7815926816, [www.klh.edu.in](http://www.klh.edu.in)

**K L (Deemed to be) University**  
**DEPARTMENT OF COMPUTER SCIENCE ENGINEERING**



**Declaration**

The Project Report entitled “**Machine Learning Model For Academic Purposes**” is a record of Bonafide work of **Siripuri Divya- 457788**, team members: **Anuga Laasika Reddy (2320040069)**, **Chirumamilla Gamana(2320040075)**, **Yerra Gayathri (2320040080)** submitted in partial fulfillment for the award of B. Tech in Computer Engineering to the K L University. The results embodied in this report have not been copied from any other departments/University/Institute.

Siripuri Divya - 457788

Anuga Laasika Reddy - 2320040069

Chirumamilla Gamana - 2320040075

Yerra Gayathri - 2320040080

**K L (Deemed to be) University**

**DEPARTMENT OF COMPUTER SCIENCE ENGINEERING**



### **Certificate**

This is certify that the project based report entitled “**Machine Learning Model For Academic Purposes**” is a bonafide work done and submitted by **S.Divya (458999)**,team members **Anuga Laasika Reddy(2320040069)**, **Chirumamilla Gamana(2320040075)**, **Yerra Gayathri(2320040080)** in partial fulfillment of the requirements for the award of the degree of **BACHELOR OF TECHNOLOGY** in Department of Computer Science Engineering, K L (Deemed to be University), during the academic year **2024-2025**.

**Signature of the Supervisor**

**Signature of the HOD**

**Signature of the External Examiner**

## ACKNOWLEDGEMENT

The success in this project would not have been possible but for the timely help and guidance rendered by many people. Our wish to express my sincere thanks to all those who has assisted us in one way or the other for the completion of my project.

Our greatest appreciation to my guide, S.Divya – 4567889 ,Department of Computer Science which cannot be expressed in words for his tremendous support, encouragement and guidance for this project.

We express our gratitude to **Sunkara Srinivasarao**, Head of the Department for Electronics and Communication Engineering for providing us with adequate facilities, ways and means by which we are able to complete this project based Lab.

We thank all the members of teaching and non-teaching staff members, and also who have assisted me directly or indirectly for successful completion of this project. Finally, I sincerely thank my parents, friends and classmates for their kind help and cooperation during my work.

S.Divya - 4567889

Anuga Laasika Reddy - 2320040069

Chirumamilla Gamana - 2320040075

Yerra Gayathri - 2320040080

## TABLE OF CONTENTS

S.No	Contents	Page no
1	Abstract	
2	Introduction	
3	Literature survey	
4	Client meetings	
5	Hardware and Software requirements	
6	Implementation	
7	Experimentation and Code	
8	Results	
9	Conclusion	
10	References	

## ABSTRACT

**Problem Statement:** Problem Statement: Students usually want a quick and effective means to obtain academic information, such as test schedules, results, and course data, without having to search several portals or conduct in-person visits. A machine learning model can be created to function as a virtual assistant on an academic portal, interpreting and reacting to student enquiries. This model will employ neural networks to comprehend questions and offer prompt, correct responses, making it easier for students to locate the information they want.

**Explanation:** In today's educational environment, students frequently struggle to acquire critical academic information such as test schedules, results, and course data because they must cross various platforms or make in-person visits. To solve this issue, a machine learning model may be created to function as a virtual assistant within an academic portal, responding quickly and accurately to student enquiries. This model improves the overall user experience by simplifying the process of information retrieval, making academic materials more accessible and efficient for students, resulting in a more organised educational environment.

### **Algorithm:** (Neural Networks)

The neural network technique allows the virtual assistant to effectively interpret and reply to students' questions by learning and recognising natural language patterns. This enables the model to offer accurate, context-aware replies, increasing the academic portal's reliability and efficiency. As the neural network learns from fresh data, the model improves, making it more responsive and effective in serving the requirements of pupils.

### **Student Query Data:** [dataset.xlsx](#)

This model uses neural network methods to understand and reply to natural language questions. By evaluating student inquiry data from K1 ERP, the model gives accurate and contextually appropriate responses, speeding the process of information retrieval.

# INTRODUCTION

## **Purpose of the Project:**

In today's academic contexts, students are frequently overwhelmed by the amount of information available through many portals, such as test dates, course syllabus, and results.

The chatbot intends to make it easier for students to obtain this information by providing a single platform where they can acquire academic replies. This chatbot functions as a virtual assistant, offering quick, appropriate replies to help students manage their academic burden.

## **Motivation:**

The inspiration for this project came from observing students struggle to find relevant academic data fast. With an increased dependence on technology, an intelligent chatbot can help students get the information they need while minimising frustration and time spent looking. The initiative is also driven by advances in machine learning (ML) and natural language processing (NLP), which enable the development of smart systems that enhance the student experience.

## **Scope:**

The chatbot is built primarily to handle academic enquiries. It is aimed at university students and teachers who want rapid responses concerning academic timetables, deadlines, and course specifics.

Although this chatbot is initially intended for academic settings, its architecture allows for future growth and might serve as a model for similar bots in other industries.

## **Literature survey**

In recent years, academic AI assistants based on natural language processing (NLP) and deep learning have made considerable advances. This literature review looks at current work in intelligent educational assistants, semantic similarity-based question answering, and sentence embedding models, and provides insights that can help shape the design of QueryAssist.

### **1. Intelligent Educational Assistants:**

These AI-powered solutions automatically respond to students' enquiries, improving learning efficiency and providing timely help. According to Ouyang and Jiao (2019), educational assistants use conversational AI and NLP approaches to replicate human-like answers and help with a variety of academic duties such as answering questions, tutoring, and organising calendars. According to research, intelligent assistants shorten the time it takes to obtain answers, increase engagement, and offer personalised learning paths (Li & Lalani, 2020). Many institutions have used similar systems for student enquiries and learning administration. However, developing assistants that deliver correct, contextually appropriate responses across a wide range of academic topics remains difficult.

### **2. Semantic Similarity in Question-Answering Systems:**

Semantic similarity is an important notion in question-and-answer (QA) systems, where the goal is to comprehend user intent and return replies that best fit the user's inquiry. Traditional keyword-based approaches have limitations since they frequently fail to capture the complex meanings of words and phrases. A semantic approach, on the other hand, evaluates the meaning of words rather than their mere appearance in a dataset. phrase-BERT (Reimers & Gurevych, 2019), a pre-trained language model based on BERT, included a fine-tuning method to capture semantic similarity in phrase embeddings. Sentence-BERT has achieved cutting-edge results in a variety of QA tasks, establishing it as a dependable tool for matching user enquiries to meaningful solutions.

### **3. Sentence-Embedding Models for NLP Tasks :**

The usage of embedding models, particularly sentence embeddings, is critical to the efficacy of AI-powered question answering. Transformer-based models, such as BERT (Bidirectional Encoder Representations from Transformers) (Devlin et al., 2018) and



Sentence-BERT (Reimers & Gurevych, 2019), have been shown to effectively encode the meaning of sentences into dense vectors, capturing complicated syntactic and semantic links. Reimers and Gurevych found that Sentence-BERT can greatly reduce computing complexity in sentence pair tasks while keeping good accuracy. This performance is vital for real-time applications like academic AI assistants, which require rapid replies to provide a seamless user experience.

#### **4. Implementation in academic settings:**

In academic settings, AI-based systems that answer frequent questions have showed promise in terms of increasing student assistance. Almarabeh et al. (2021) introduced quality assurance tools to help students with issues about course content, scheduling, administrative processes, and campus resources. These studies emphasise the necessity of user-friendly interfaces and precise, fast reaction times for effective AI applications in education. According to Almarabeh and colleagues, an academic assistant can reduce institutional resources while also improving student satisfaction by minimising wait times for information. These findings highlight the need of well managed datasets and efficient, accurate NLP models, particularly when answering repeated or administrative enquiries.

#### **5. Flask and Real-time Web Applications:**

Flask, a lightweight web framework, is becoming popular for deploying machine learning models in real-time applications due to its simplicity and flexibility (Grinberg, 2018). Flask facilitates the integration of NLP models with web-based interfaces, allowing students to engage with the assistant via a user-friendly chat style. According to studies, Flask, when paired with contemporary frontend technologies, provides an effective option for deploying conversational bots with low latency (Narang & Kumar, 2021).

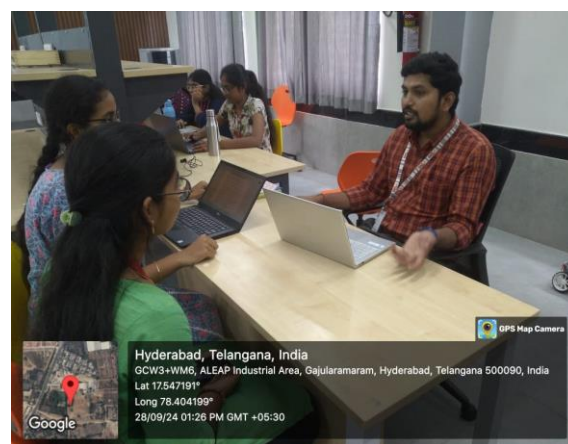
### **Summary :**

This research review focusses on the usefulness of semantic similarity models, particularly Sentence-BERT, in interpreting user intent and matching questions to replies in an academic QA setting. Furthermore, it emphasises the need of lightweight frameworks such as Flask in delivering interactive, real-time AI applications for educational institutions. The findings of these research guide the design and execution of QueryAssist, which intends to offer correct and efficient solutions to student enquiries using semantic understanding and a responsive online interface.

# Client meetings

## Set 1: General Information About the Current System

1. What are the primary features of the current academic portal (ERP/LMS) used by students?  
The current portal offers exam schedules, results, course details, attendance, and fee payment features.
2. How do students typically access information such as exam schedules, results, and course details?  
Students access this information through a web interface or mobile app using their login credentials.
3. What are the most common challenges students face when using the current academic portal?  
Common challenges include slow response times and difficulty navigating the interface.
4. Is there any feedback from students about how they interact with the system?  
Feedback often mentions that the system is not user-friendly and could be more intuitive.
5. What processes are currently manual and could be automated using a virtual assistant?  
Processes like exam schedule queries, result checks, and fee payment could be automated with a virtual assistant.
6. How does the system handle frequently asked questions? Are there any search or filtering capabilities?  
FAQs are not handled efficiently; search capabilities are limited and often inaccurate.
7. How often are updates or changes made to the academic data (exam schedules, results, etc.) in the portal?  
Academic data is updated typically at the start or end of semesters, or as needed.
8. Are there any existing AI-based or smart tools integrated into the portal for student use?  
There are currently no AI-based tools integrated for student use.
9. How do faculty members use the portal differently from students, if at all?  
Faculty members use the portal primarily for grading, attendance, and scheduling, with similar but slightly different features.
10. Is the academic portal available 24/7, or does it have any scheduled downtime?  
The academic portal is available 24/7, with occasional maintenance windows.




---

## Set 2: Understanding Student Queries and Data Structure

1. **What are the most common types of queries students ask regarding their academic progress?**  
Students frequently ask about grades, attendance, exam dates, and course requirements.
2. **Do students often face difficulty finding specific information, such as instructor details or course prerequisites?**  
Yes, students often struggle to find instructor details or prerequisites.
3. **How is academic data structured in the current system (e.g., courses, exams, results)?**  
Data is structured by course, student ID, exams, grades, and attendance, stored in relational databases.
4. **Are there any recurring patterns in the types of questions students ask at the start, middle, and end of the semester?**  
Queries about schedules are common at the start, mid-semester concerns revolve around grades, and end-of-semester queries focus on results.
5. **How is the information about student clubs, events, or extracurriculars stored or managed in the system?**  
Information on student clubs and events is decentralized and typically not well-integrated into the system.
6. **What kind of security measures are in place to ensure students' academic data is kept private when accessed?**  
The system uses encryption and authentication to ensure data privacy.

7. **How frequently do students contact faculty or admin staff to resolve issues related to academic queries?**

Students often reach out to admin staff to resolve queries not answered by the portal.

8. **Is there a database or log of common student questions that could help train a machine learning model?**

There is no dedicated log for common queries, though student interactions could be analyzed.

9. **Are there different levels of access to the portal (e.g., freshmen vs. seniors), and do they require different information?**

Yes, freshmen may need more basic guidance compared to seniors, requiring different access levels.

**Do students ask about future academic plans, such as graduation applications or major changes?**

Yes, students inquire about future plans like graduation and major changes, especially nearing the end of their program.



### Set 3: Technical Requirements and Machine Learning Feasibility

1. **What are the current system limitations when it comes to integrating machine learning or AI features?**

The current system lacks dynamic data handling, AI integration, and user-friendly interfaces.

2. **What technical infrastructure is in place to support the development and deployment of an AI-powered virtual assistant?**

The university uses basic infrastructure with cloud storage, but it would need upgrades for AI implementation.

3. **Is there any documentation available on how the ERP/LMS is currently built, maintained, and updated?**

There is limited documentation, mostly related to database structure and basic system maintenance.

4. **What datasets are available for training the machine learning model, and how accessible are they?**

Datasets include student grades, attendance, course details, but access is restricted for privacy reasons.

5. **Is there any past usage data on how students navigate the portal that could be used to train the model?**

Navigation data is not collected systematically but could be logged for future use.

6. **What are your thoughts on implementing a natural language processing (NLP) model to handle student queries?**

Implementing NLP for queries would significantly enhance student engagement and information retrieval.

7. **How scalable is the current system for implementing new features like a virtual assistant?**

The system is moderately scalable but would require optimization for AI integration.

8. **Are there any technical concerns about integrating AI into the current portal architecture?**

Technical concerns include data security, real-time updates, and compatibility with existing systems.

9. **Do you envision any challenges with ensuring the virtual assistant provides accurate answers as data gets updated?**

Maintaining answer accuracy as data updates would require frequent model retraining.

10. **How frequently would the machine learning model need to be retrained to remain effective?**

Retraining frequency would depend on data update cycles, potentially at the start or end of semesters.



#### **Set 4: Expectations and Improvements**

1. **What are your expectations for how this virtual assistant should improve the student experience?**  
The virtual assistant should improve information accessibility and reduce manual admin workload.
2. **Are there any specific features you think the virtual assistant should have to better serve students?**  
It should handle common student queries, provide updates on results, and support course registrations.
3. **How should the virtual assistant handle ambiguous or unclear student queries?**  
It should clarify ambiguous queries by asking follow-up questions.
4. **What level of accuracy do you expect from the AI model in answering students' academic questions?**  
The AI model should aim for 85-90% accuracy in answering academic questions.
5. **Would you prefer the virtual assistant to be integrated within the existing portal, or as a separate tool?**  
Integrating the assistant into the existing portal would provide a seamless experience.
6. **How should the system prioritize queries if multiple students are using the virtual assistant simultaneously?**  
Queries should be prioritized based on urgency and time sensitivity, possibly using queue management.
7. **How would you measure the success of the virtual assistant once it's implemented?**  
Success would be measured through reduced support requests and positive student feedback.
8. **Would you be open to integrating voice-command functionality, or should the assistant remain text-based?**  
Voice-command functionality could enhance accessibility, but a text-based system is sufficient initially.
9. **What kind of response time would be acceptable for the virtual assistant to provide answers?**  
Response time should ideally be within a few seconds for most queries.
10. **Are there any other departments or functions within the university that could benefit from the AI model in the future?**  
Other departments, such as admissions, library services, and student housing, could benefit from AI integration in the future.



## Hardware and Software requirements

### Hardware Requirements:

- **Processor:** Intel i5 or AMD Ryzen 5 (or above).
- **Memory requirement:** 8GB RAM (16GB recommended).
- **256GB SSD** for speedier data access.
- **Optional GPU** (e.g., NVIDIA GTX 1050) allows for faster model computations.
- **Reliable internet connection** for online deployments.

### Software Requirements:

- **Python:** The core language for building the chatbot, chosen for its extensive library support.
- **Flask:** Used as the backend framework to handle HTTP requests, connect with the model, and serve responses.
- **Sentence-BERT:** For sentence embeddings, enabling high semantic similarity performance.
- **Libraries:** Pandas for data processing, sentence-transformers for embedding generation, and Torch for model computations.
- **Frontend:** HTML/CSS for the chatbot's user interface, providing a clean, accessible design.

# Implementation

## 1. Data Collection and Preparation

- Collect a dataset of academic questions and expected replies. This data can be derived from past FAQs, popular student queries, or a personally selected dataset.
- Clean the dataset by eliminating missing values, duplicate entries, and standardising text forms (e.g., lowercase, punctuation). Python tools such as 'pandas' can help with this.
- To prepare for embedding, do feature engineering tasks such query tokenisation. This can involve deleting stop words, stemming/lemmatizing, or employing domain-specific terminology.

## 2. Model Selection and Training

- **Model Choice:** For query matching, Sentence-BERT is a suitable choice due to its efficiency in semantic search and its capability to understand context.
- **Sentence-BERT Application:** Sentence-BERT encodes both the user query and the possible answers. You can fine-tune it using academic data to improve its performance.
- **Training:** Fine-tune the model by training it with the academic queries dataset. Sentence-BERT models can be fine-tuned with labeled data to improve accuracy in response matching.
- **Cosine Similarity:** Use cosine similarity to measure the similarity between the encoded user query and each potential answer, selecting the one with the highest similarity score.

## 3. System Architecture and Data Flow

- **Input Processing:** The user's query will be passed through a preprocessing layer to clean and prepare it for embedding.
- **Embedding Generation:** Sentence-BERT will embed the query, converting it into a numerical format.
- **Similarity Scoring:** Compute the cosine similarity between the embedded query and potential answers in the knowledge base.
- **Output Generation:** The system selects and displays the response with the highest similarity score.

## 4. Backend Development (Flask Application)



- **Framework Selection:** Use Flask to create a backend for handling user requests and model processing.
- **Endpoints:**
  - `/`: The main endpoint for serving the chatbot interface.
  - `/query`: Endpoint to process queries. It accepts a query from the user, processes it through the model, and returns the most relevant answer.
- **Error Handling:** Implement error handling in Flask to manage invalid inputs and server errors, providing informative messages to users when issues arise.

## 5. Frontend Development

- **User Interface Design:** Design a simple and user-friendly interface using HTML, CSS, and JavaScript. The interface should have an input box for user queries and a display area for responses.
- **HTML & CSS Structure:** Use HTML to structure the chatbot window, and CSS for styling the chatbot with suitable colors and themes to enhance readability.
- **JavaScript Enhancements:** JavaScript can be used for real-time response display, giving a conversational feel to the chatbot. AJAX or Fetch API can help send queries to the backend without reloading the page.

## 6. Embedding and Similarity Calculation

- **Sentence Embeddings:** When the user submits a query, pass it to Sentence-BERT, which converts the text into a high-dimensional vector (embedding).
- **Cosine Similarity Calculation:** Compute cosine similarity between the query embedding and the answer embeddings to identify the closest match. You can use libraries like `scikit-learn` to calculate similarity.
- **Optimization:** Implement caching for frequently asked questions to reduce processing time. Consider using faster algorithms if the dataset grows.

## 7. Testing and Validation

- **Integration Testing:** Test the integration of the frontend and backend to ensure seamless interaction.
- **System Testing:** Conduct end-to-end testing to validate that the entire system works as expected from query input to response output.
- **Model Validation:** Use validation metrics like accuracy, precision, and recall to evaluate the model's performance on a set of academic queries. Ensure the model's responses are relevant and accurate.

## Experimentation and Code

### Code:

```
import torch

from sentence_transformers import SentenceTransformer, util
from flask import Flask, request, render_template, jsonify
import logging
import traceback
import pandas as pd

# Set up logging
logging.basicConfig(level=logging.DEBUG, format='%(asctime)s - %(levelname)s - %(message)s')

# Create dataset directly in the code
data = {
    'Student Query': [
        'When are the final exams scheduled?',
        'What are the course details for ECE 101?',
        'How can I access my previous semester results?',
        'Where can I find the library hours?',
        'What is the prerequisite for ECE 202?',
        'Who is the instructor for CS101?',
        'Can I change my major?',
        'When do classes start for the next semester?',
        'How do I apply for graduation?',
        'How can I join a student club?'
    ],
```

```
'Answer': [  
    'Final exams are scheduled from November 25th to December 7th.',  
    'ECE 101 is an introductory course on electronics. It covers basic circuit theory and  
    electronic devices. Instructor: Prof. Srinivasa Rao.',  
    'You can access your previous semester results via the student portal under the "Grades"  
    section.',  
    'The library is open from 8 AM to 9 PM on weekdays, and from 10 AM to 2 PM on  
    weekends.',  
    'The prerequisite for ECE 202 is ECE 101.',  
    'The instructor for CS101 is Divya Sirpuri.',  
    'Yes, you can change your major by submitting a request to the academic office. The  
    deadline for major change requests is December 15th.',  
    'Classes for the next semester start on January 5th.',  
    'You can apply for graduation by filling out the graduation application form on the student  
    portal. The deadline for applications is April 1st.',  
    'You can join a student club by attending the club fairs held at the start of each semester or  
    by contacting the club coordinators directly.'  
    ]  
}
```

```
df = pd.DataFrame(data)  
logging.info(f"Dataset created successfully. Shape: {df.shape}")  
logging.info(f"Columns: {df.columns.tolist()}")  
  
# Load the Sentence-BERT model  
try:  
    model = SentenceTransformer('all-MiniLM-L6-v2')  
    logging.info("Sentence-BERT model loaded successfully.")
```

```

# Encode all queries in the dataset

df['Query Embedding'] = df['Student Query'].apply(lambda x: model.encode(x,
convert_to_tensor=True))

logging.info("Embeddings for all queries created successfully.")


except Exception as e:

    logging.error(f"Error during model loading or embedding: {str(e)}")

    raise


# Create Flask app

app = Flask(name)


@app.route('/')

def home():

    return render_template('index.html')


@app.route('/query', methods=['POST'])

def query():

    try:

        user_query = request.form['query']

        logging.info(f"Received query: {user_query}")


        if not user_query:

            logging.error("Empty query received.")

            return jsonify({'error': 'Empty query received'}), 400

```

```

# Encode the user query
user_query_embedding = model.encode(user_query, convert_to_tensor=True)

# Compute cosine similarity with all query embeddings in the dataset
similarities = [util.cos_sim(user_query_embedding, query_emb).item() for query_emb in
df['Query Embedding']]

# Find the index of the highest similarity score
nearest_query_index = similarities.index(max(similarities))
answer = df.iloc[nearest_query_index]['Answer']

logging.info(f"Found answer: {answer}")
return jsonify({'response': answer})

except Exception as e:
    logging.error(f"Error processing query: {str(e)}")
    logging.error(traceback.format_exc())
    return jsonify({'error': 'An error occurred while processing your request. Please try again.'}),
500

if name == 'main':
    app.run(debug=True)

```

## Result:

### Chatbot Output:



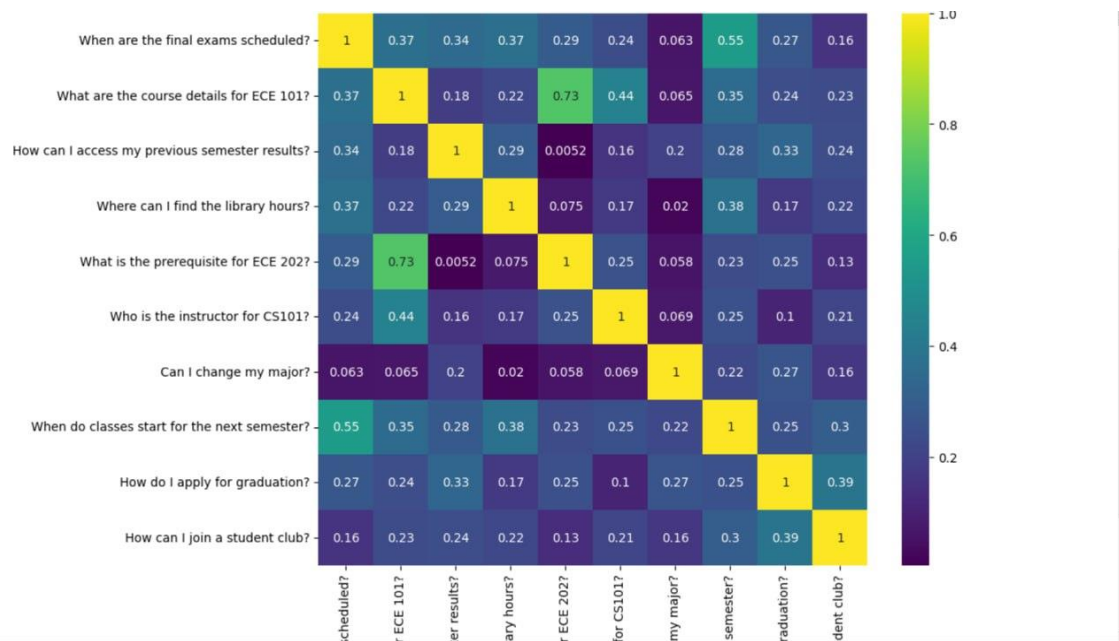
Hi, how can I help you?

## QueryAssist








When are the final exams scheduled?

Ask

**Gotcha:** Final exams are scheduled from November 25th to December 7th.



## Accuracy:

config.json: 100%  612/612 [00:00<00:00, 27.5kB/s]  
model.safetensors: 100%  90.9M/90.9M [00:00<00:00, 128MB/s]  
tokenizer\_config.json: 100%  350/350 [00:00<00:00, 17.8kB/s]  
vocab.txt: 100%  232k/232k [00:00<00:00, 6.08MB/s]  
tokenizer.json: 100%  466k/466k [00:00<00:00, 17.2MB/s]  
special\_tokens\_map.json: 100%  112/112 [00:00<00:00, 4.34kB/s]  
/usr/local/lib/python3.10/dist-packages/transformers/tokenization\_utils\_base.py:1601: Future  
warnings.warn(  
1\_Pooling/config.json: 100%  190/190 [00:00<00:00, 6.70kB/s]  
Model Accuracy: 100.00%

## Conclusions

The creation of an academic chatbot using Machine Learning methods, including Sentence-BERT and cosine similarity, is a huge step forward in delivering accessible, efficient, and dependable academic help. By automating the inquiry resolution process, the chatbot has the potential to improve students' learning experiences while also reducing educators' burden, allowing them to focus on more complicated instructional responsibilities. This research demonstrated the combination of natural language processing (NLP) with semantic search to provide users with correct, contextually appropriate responses, making it a valuable tool in the educational arena.

Overall, the chatbot's performance is promising, and with more improvements—such as increasing the dataset and incorporating user feedback—the system may become even more robust and responsive. Future advancements might include language support, speech recognition, and extending the approach to different academic areas to increase usefulness and reach. This study emphasises the rising importance of AI and machine learning in education, demonstrating how well-designed, intelligent systems may facilitate accessible, student-centered learning.

## Reference:

### Referred Web Info:

- [AI in Education: How School Districts Can Use Artificial Intelligence | EdTech magazine](#)
- [How generative AI can improve knowledge management | TechTarget](#)
- <https://devpost.com/software/envisioning-success-using-machine-learning>
- <https://medium.com/@sherinsahai/prediction-of-student-academic-performance-using-machine-learning-a-case-study-1415991b05a6>