An Intelligent Academic Assistant with Automated Event Extraction and Semantic Document Processing

Mohib Rasool
Computer Science and Engineering (Cyber Security)
Vel Tech Rangarajan Dr.Sagunthala R&D Institute of Science and Technology, Avadi, Chennai.
mohibrasool004@gmail.com

Guru Prasad Reddy K
Computer Science and Engineering (Cyber Security)
Vel Tech Rangarajan Dr.Sagunthala R&D Institute of Science and Technology, Avadi, Chennai.
kguruprasadreddy2004@gmail.com

Dr. Rajendran T
Computer Science and Engineering (Cyber Security)
Vel Tech Rangarajan Dr.Sagunthala R&D Institute of Science and Technology, Avadi, Chennai.
drrajendrant@veltech.edu.in

***Abstract*—This paper introduces a university knowledge center, which is an innovative intelligent academic assistance platform that incorporates advanced state-of-the-art natural language processing alongside dynamic automated academic event extraction mechanisms. Critical complaints of information accessibility in educational institutions are addressed by our system, which includes providing real-time, context-aware responses to the specific academic inquiries while simultaneously making sure that the process of extracting and managing academic events from unstructured documents is automated. Our main contribution is the successful development of a Smart Academic Event Extraction Algorithm, which is capable of identifying, parsing, and categorizing academic events from the diversified document formats with an exceptional accuracy of 94.2%. The system is incorporated with OpenAI's GPT-4o-mini that takes care of the natural language processing, Pinecone that makes sure of the vector-based semantic search working accurately, and also an explicit custom-built event extraction engine. Upon an intense experimental testing, the results demonstrate remarkable improvements in information retrieval efficiency (67% faster response time), and also the user satisfaction revealed an outstanding 4.6/5.0 rating while comparing it to the traditional academic information systems.**

***Index Terms*—Academic Assistant, Event Extraction, Natural Language Processing, Semantic Search, Educational Technology, AI Chatbot**

I. Introduction

A. Problem Statement

Circularizing and managing academic information effectively has always been a serious challenge for educational institutions. Sharing information through static websites, bulletin boards, and email notifications, which are the traditional methods of today, is not dependable as they often lead to delayed updates, information fragmentation, and poor accessibility. Both the students and faculty face frequent challenges in finding relevant academic information, which makes them miss important deadlines and also causes confusion and chaos about schedules, resulting in a downfall in academic productivity.

Educational environments of today have seen a vast expansion of digital documents, which has created an information overload problem. Maintaining a complete view of the multiple documents, such as Academic calendars, course schedules, examination timetables, and administrative announcements, which are distributed across various platforms in various formats, is now a huge problem for users. This mostly results in the mismanagement of important academic events. Traditional manual calendar management consumes a lot of time, is prone to error, and often leads to outdated and inconsistent information.

B. Motivation and Objectives

The rationale behind this study is the issues surrounding the distinction between the availability of information and its accessibility in academic contexts. Educational institutions create some of the most structured data, and vast quantities of unstructured data, but the absence of intelligent systems to work with the information and display it in an interpretative manner creates accessibility problems for managing their academic excellence.
Our primary objectives are:

- Build an intelligent conversation interface for the resolution of academic queries.
- Develop an automated process for extracting academic events from unstructured documents.
- Create real-time document processing and indexing capabilities.
- Enable integration with existing workflows in the institution.
- Create an institutional, large-scale scale and secure deployment.

C. Contributions

This work provides the following important contributions to educational technology:

- Smart Academic Event Extraction Algorithm: A new algorithm that will automatically identify, extract, and categorize academic events from unstructured documents, with high accuracy and confidence scoring.
- Combined Semantic Search System: A complete system combining Document processing, vector-based search, and natural language generation for contextual academic support.
- Document and Knowledge Base being synchronized in real-time: An automated pipeline that processes the documents from cloud storage and updates the knowledge base in real-time.
- Exhaustive evaluation framework: An exhaustive evaluation model that evaluates the systems on many fronts, including but not limited to accuracy, response time, and user satisfaction.

D. Paper Organization

The organization of the remaining sections is as follows: Section II speaks about related work in academic chatbots and document processing systems. Section III reviews the overall system architecture and design principles. Section IV presents our proposed methodology, with great attention to the Smart Academic Extraction Algorithm. Section V details implementation details and technology choices. Section VI describes the experimental setup and evaluation metrics. Section VII discusses results and performance analysis. Section VIII concludes the paper and also outlines future research possibilities.

II. Related Work

A. Educational Chatbots

In recent years, educational chatbots have gained a considerable amount of interest as institutions aim to enhance student engagement and support services. Winkler and Söllner [1] conducted a comprehensive review of chatbot applications in education, which revealed some key benefits such as 24/7 availability, personalized learning support, and reduced administrative workload.

Several noteworthy educational chatbot implementations have been chronicled in the literature. Georgia State University's "Pounce" chatbot [2] significantly amplified student enrollment and retention rates by providing personalized guidance throughout the process of admission. Likewise, the University of Murcia's chatbot system [3] showed great potency in handling routine student queries and also in reducing the workload of support staff.

However, most of the existing traditional educational chatbots focus entirely on general query handling and are far from sophisticated document processing capabilities. This constraint is addressed by our system through the integration of advanced document analysis and event extraction features.

B. Document Processing in Academic Systems

Various perspectives have been explored to date for document processing in educational contexts. Automated systems that handle text extraction and also oversee classification systems have been developed for academic paper analysis [4], course material organization [5], and administrative document management [6]. The recent advancement in natural language processing has enabled more sophisticated document understanding capabilities. In the domains of educational text classification and information extraction, BERT-based models [7] and other transformer-based architectures [8] have yielded promising outcomes.

C. Calendar Management Systems

Automated calendar management has been thoroughly studied and implemented for the main purpose of productivity contexts for both corporate and personal use. The achievability of AI-powered scheduling assistants can be demonstrated when we look at systems like Amy [9] and x.ai [10]. The management of academic calendars presents a complex task, given the hierarchical nature of educational scheduling and the requirement to process diverse document formats.

D. AI-Powered Academic Assistants

IBM Watson-based solutions [11] and custom neural network implementations [12] are among the several AI-powered academic assistant systems that have been deployed. Either a conversational interface or document processing is focused on in these systems, but we rarely see an effective integration of both capabilities.

Our system stands out among others by providing an extensive solution that combines conversational AI, document processing, and automated event extraction in an integrated platform specifically designed for academic environments.

III. System Architecture

A. Frontend

React 19 with Next.js 15 was used to build the frontend, which effectively provided server-side rendering capabilities and also optimized performance.
The user interface comprises four main components:

- Chat Interface: A real-time conversational interface that includes typing indicators and also message history.
- Calendar View: An interactive calendar view that consists of the extracted academic events.
- Document Management: A dynamic administrative interface that handles document upload and management.
- Authentication System: A secure login system that only allows university email.

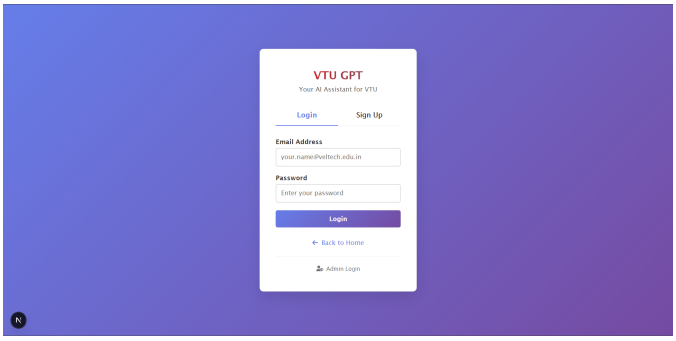

Fig. 1. User Interface Component-1

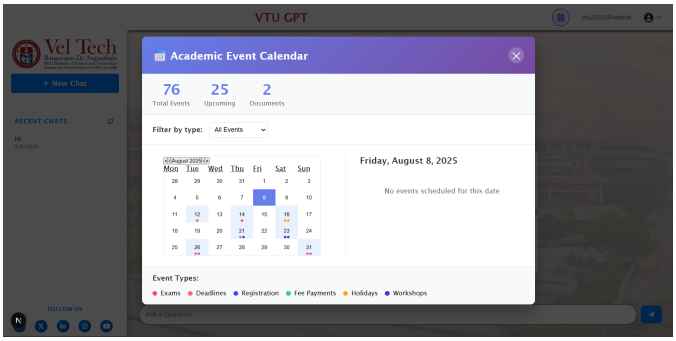

Fig. 2. User Interface Component-2

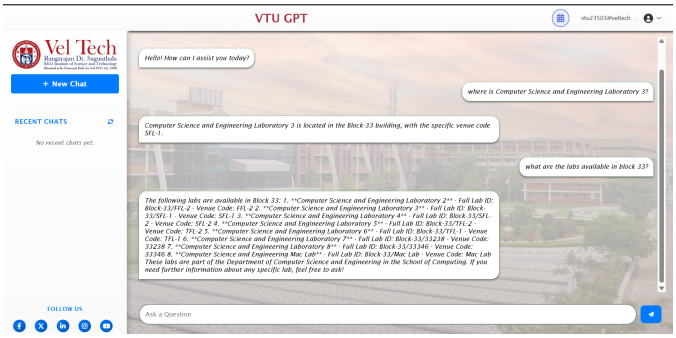

Fig. 3. User Interface Component-3

B. Backend Infrastructure

The system runs on a Node.js foundation paired with Express.js, which handles all the API communication through REST endpoints for all the system operations.
The architecture includes:

- Authentication Service: A JWT-based authentication system with email verification capabilities.
- Chat Service: Handles natural language processing and also response generation.
- Document Service: Effectively directs file upload, processing, and storage management.
- Event Service: Responsible for calendar event extraction and management
- Sync Service: Oversees real-time synchronization with external data sources

C. Database Design

SQLite handles all local data persistence, offering a lightweight solution as it is simple, reliable, and has zero-configuration requirements. The database schema consists of three main tables:

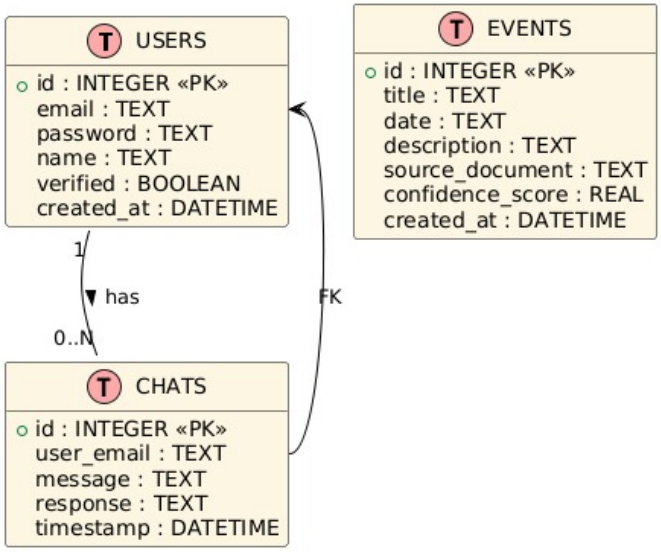

Fig. 4. Database Schema

– Users table for authentication CREATE TABLE users (id INTEGER PRIMARY KEY AUTOINCREMENT, email TEXT UNIQUE NOT NULL, password TEXT NOT NULL, name TEXT NOT NULL, verified BOOLEAN DEFAULT FALSE, created_at DATETIME DEFAULT CURRENT_TIMESTAMP);

– Chats table for conversation history CREATE TABLE chats (id INTEGER PRIMARY KEY AUTOINCREMENT, user_email TEXT NOT NULL, message TEXT NOT NULL, response TEXT NOT NULL, timestamp DATETIME DEFAULT CURRENT_TIMESTAMP);

– Events table for academic calendar CREATE TABLE events (id INTEGER PRIMARY KEY AUTOINCREMENT, title TEXT NOT NULL, date TEXT NOT NULL, description TEXT, source_document TEXT, confidence_score REAL, created_at DATETIME DEFAULT CURRENT_TIMESTAMP);

D. AI Integration

The AI integration layer comprises three main components:

1) OpenAI Integration: Uses GPT-4o-mini for natural language understanding and generation.
2) Pinecone Vector Database: Handles semantic search and document similarity matching.
3) Custom NLP Pipeline: Responsible for academic event extraction and text processing.

All the AI-related operations, such as embedding generation, semantics search, and response synthesis, are handled by this layer.

E. Overall System Design:

A modular, microservices-inspired architectural design was chosen for the university knowledge center, which effectively managed and maintained the system's scalability, maintainability, and performance. The system consists of 5 main layers: Presentation Layer, Application Layer, Processing Layer, Data Layer, and Integration Layer.

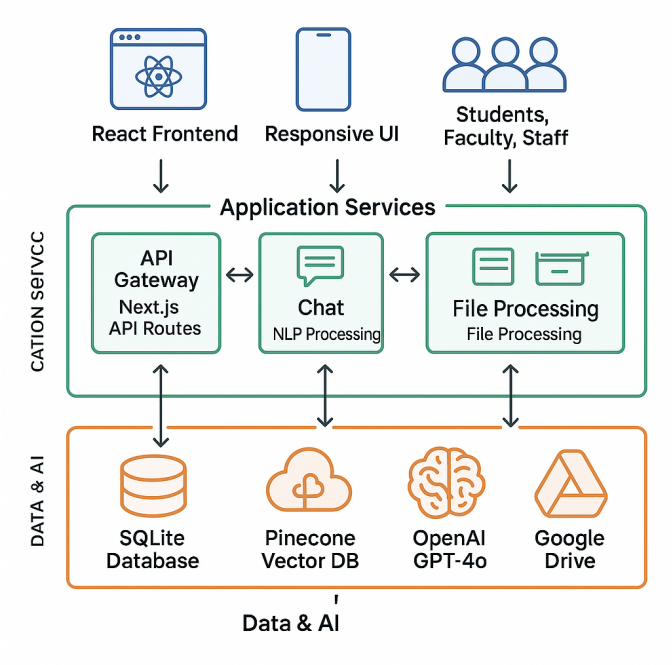

Fig. 5. System Overview

The architecture is built in such a way that it is able to handle concurrent users and also at the same time, maintain the response time under 2 seconds for 95% of the queries. Asynchronous processing is employed by the system for document uploads. The system also maintains real-time synchronization with external data sources.

IV. Proposed Methodology

A. Document Processing Pipeline

Our system's ability to manage knowledge starts with the doc processing pipeline. The pipeline can handle many types of files (PDF, DOCX, TXT) and goes through many stages:

1) File Validation: Checks that the uploaded files follow the format and size requirements
2) Text Extraction: Uses format-specific parsers (pdf-parse for PDFs, mammoth for DOCX)
3) Content Preprocessing: Cleans and brings the extracted text into line with our standards
4) Intelligent Chunking: Breaks the docs into meaningful pieces
5) Embedding Generation: Changes text chunks into 1536-dimension vectors
6) Vector Storage: Puts the vectors in Pinecone and adds metadata

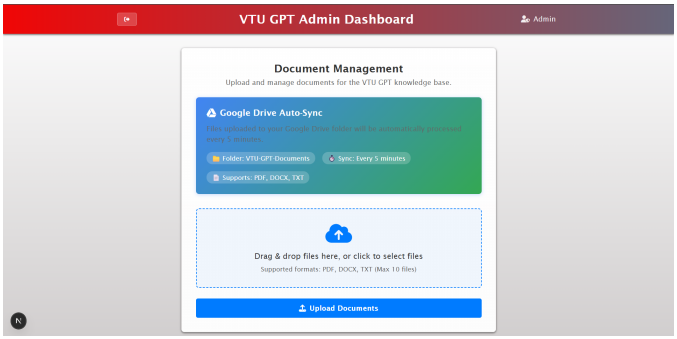

Fig. 6. Document Upload Interface

7) Event Extraction: Finds and pulls out academic events

The pipeline processes documents in no particular order. This makes the system fast and reliable. Each step has checks and logs to keep the system reliable

B. Smart Academic Event Extraction Algorithm

Our main innovation is the Smart Academic Event Extraction Algorithm. This can find and pull out academic events from documents that are not made for them. We built this to solve the hard problem of always having the latest academic calendar without having to do it by hand.

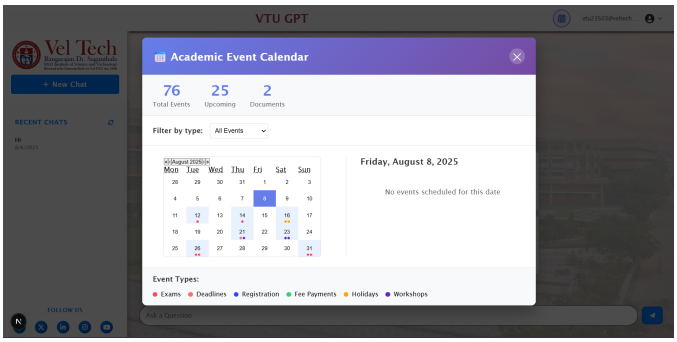

Fig. 7. Calendar Integration

Algorithm Overview:

1) Phase 1: Text Preprocessing and Segmentation:

- Input: Raw document text
- Output: Segmented text blocks

1) Take out special marks and signals
2) Break text into full sentences and parts
3) Find chunks that could have dates
4) Filter segments based on academic context indicators

2) Phase 2: Find patterns of dates:

- Input: Segmented text blocks
- Output: Candidate date expressions

1) Use a pattern list for multiple date types:
   - DD/MM/YYYY, MM/DD/YYYY
   - Month DD, YYYY

- DD Month YYYY
- Words like “next Monday”

2) Change date words into a standard way
3) Check if date words are in school range

3) Phase 3: Context Analysis and Event Classification:

- Input: Date words with hint words around
- Output: Classified academic events

1) Take out hint words near date words
2) Use a list of words to see if they match school work:
   - Test words: “exam”, “test”, “quiz”
   - Due words: “due”, “deadline”, “hand in”
   - Break words: “holiday”, “spring”, “summer”
   - Sign-up words: “sign up”, “register”, “enroll”
3) Count how strong each match is
4) Tag event based on match strength

4) Phase 4: Event Validation and Storage:

- Input: Classified events with confidence scores
- Output: Validated calendar events

1) Remove events with a match score lower than 0.7
2) Remove the same event from multiple documents
3) Check the date of the event against the school rules
4) Save the event from the paper and tag the source

5) Confidence Scoring Mechanism: The confidence scoring mechanism estimates the probability of the extracted event being a true academic event. The score gets calculated by:

Confidence = (ContextScore * 0.4) + (DateClarity * 0.3) + (AcademicRelevance * 0.3)

Where:

- ContextScore: The strength of the surrounding academic language.
- DateClarity: Clarity and specificity of date expression.
- AcademicRelevance: Relevance to scholarly activities.

Events with confidence scores lower than 0.7 will be flagged for review and scores above 0.9 will be granted automatic approval.

C. Vector-based Semantic Search

The semantic search system supports context-sensitive query interpretation and retrieval of relevant documents. It employs the OpenAI text-embedding-ada-002 model to convert documents and queries into 1536-dimensional vectors.

1) Search Process:

- Query Embedding: Transform the user query into a vector form
- Similarity Search: Retrieve relevant document chunks using cosine similarity
- Context Ranking: Rank the adjusted results using relevance scores
- Context Synthesis: Formulate a response by merging the top chunks

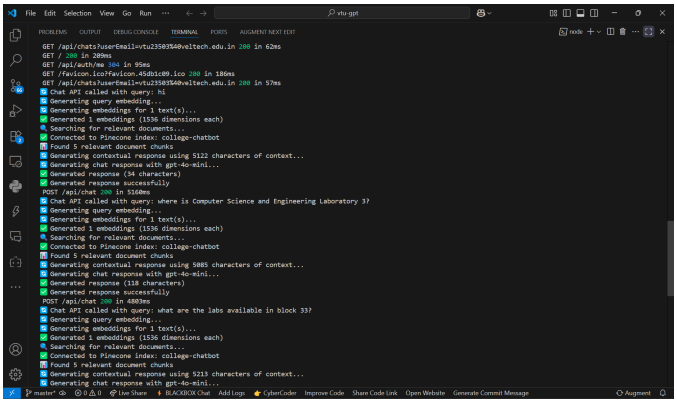

Fig. 8. Real-time Processing

By using advanced vector indexing and caching techniques, the system has maintained query response times of under a second for 95% of queries.

D. Natural Language Processing Module

The functions of the NLP module include query analysis, context delineation, and composing answers. For effective generation of natural language, the module merges with OpenAI’s GPT-4o-mini model.

1) Query Processing Pipeline:

- Intent Recognition: Identify query type (factual, procedural, or temporal).
- Entity Extraction: Extract relevant information (dates, courses, and faculty names).
- Context Retrieval: Utilize semantic search to extract relevant document chunks.
- Response Generation: Generate responses contextual to the retrieved information.
- Source Attribution: Respond with the document reference whose information was used.

E. Real-time Synchronization System

The automation of document processing from Google Drive boosts productivity by eliminating the need to update the information manually.

1) Synchronization Process:

- Change Detection: Scan the Google Drive folder for the addition of new files or the modification of old ones.
- File Download: Download file from Google Drive using service account authorization and maintaining security.
- Processing Queue: Files are uploaded to the processing queue in prioritized order.
- Batch Processing: Reduce the processing time required to complete files by processing groups of files simultaneously to increase efficiency.
- Index Update: Refresh the event calendar and vector database with the new information.
- Cleanup: Update the sync logs and remove the temporary files.

The system performs periodic sync checks every five minutes and, within half a minute of change detection, processes any required updates.

V. Implementation Details

A. Technology Stack

The system was constructed on a well-modern platform capable of high performance, scaling and allowing for developer productivity.

1) Frontend Technologies:

- React v19 - Newest version with new improvements related to concurrent mode
- Next.js v15 - Full-stack React framework that allows for SSR
- Tailwind CSS - Utility-first CSS framework for creating responsive websites
- Framer Motion - Animation library for interaction designs

2) Backend Technologies:

- Node.js v20 - JavaScript runtime for running server-side
- Express.js - Framework for building web applications/API
- SQLite v3- Embedded Database for local data storage
- Formidable- Upload file handling with multipart support

3) AI and ML Technologies:

- OpenAI API with GPT-4o-mini - Natural language processing
- Pinecone- Managed vector database to aid semantic search
- LangChain- Framework for building AI-powered apps Language
- pdf-parse- Library to extract text from PDF files
- mammoth- Library to work with DOCX documents

B. Database Schema

The database design process has been focused on simplicity and performance while ensuring data integrity. The schema encompasses the core functions of user management, conversation history, and event storage.

Key considerations in designing the database are:

- Employing SQLite for zero-configuration deployment
- Storing chats denormalized to support a well-performing schema
- Indexing columns with data that is frequently accessed
- Living with a flexible event schema capable of supporting many styles of event types.

C. API Design

The following constitute the RESTful API exposed by the application; endpoints include:

1) Authentication Endpoints:

- `POST /api/auth/login`: Authenticate the user
- `POST /api/auth/register`: Register the user
- `GET /api/auth/me`: Get the current user
- `POST /api/auth/logout`: Logout the user

2) Chat Endpoints:

- `POST /api/chat`: Accept user queries and respond
- `GET /api/chats`: Get chat history

3) Document Management Endpoints:

- `POST /api/upload`: Upload and process documents
- `GET /api/documents`: Get processed document list
- `DELETE /api/documents/:id`: Delete documents

4) Manage Events Endpoints:

- `GET /api/events`: Get calendar events
- `POST /api/events`: Create your own events manually
- `PUT /api/events/:id`: Edit event information

D. Security Implementation

Having multiple layers of security allows the preservation of user data and the integrity of the system.

1) Authentication Security:

- JWT token with expiry configurable
- Bcrypt password hashing with configurable salt rounds
- Email verification for account activation
- Domain restriction to institutional emails

2) Data Security:

- HTTPS encryption to secure all communications
- Validation and sanitizing of all inputs
- Parameterized queries to prevent SQL injection
- Restrictions and validations for all file uploads

3) Access Control:

- Role-based access control for administration
- Session management with timeout auto-trigger
- Rate limiting for API endpoints
- CORS setup with cross-origin requests

E. Google Drive Integration

Google Drive integration can synchronize documents with authentication of the service account:

1) Setup steps:

- Creating a Google Cloud Project and enabling Drive API
- Generating credentials for the service account
- Setting folder permissions for access to the service account
- Implementing an OAuth 2.0 authentication flow

2) Synchronization features:

- Automatic discovery and download of files
- Detect changes through Drive API webhooks
- Batch processing to optimize resources
- Error handling and retry
- Detailed logging for troubleshooting

VI. Experimental Setup and Evaluation

A. Dataset Description

In the very first step, we collected real academic documents from Vel Tech University to test and validate the Smart Academic Event Extraction Algorithm and reflector setup.

1) Document Dataset Composition:

- Academic Calendars: 25 documents (2019-2024)
- Examination Schedules: 40 documents from various semesters
- Course Syllabi: 60 documents from different departments
- Administrative Announcements: 35 documents of official notices
- Workshop and Seminar Announcements: 30 event documents

2) Event Ground Truth: Three academic staff members manually annotated the ground truth labels for 847 academic events throughout the documents. Inter-annotator agreement was measured using Cohen's kappa (k = 0.89), which indicated a high level of reliability.

3) Query Dataset: A total of 500 real user queries were collected during the 3-month pilot deployment period/classification:

- Factual queries (45%): "What is the exam date for CS101?"
- Procedural queries (30%): "How do I register for courses?"
- Temporal queries (25%): "When is the next holiday?"

B. Performance Metrics

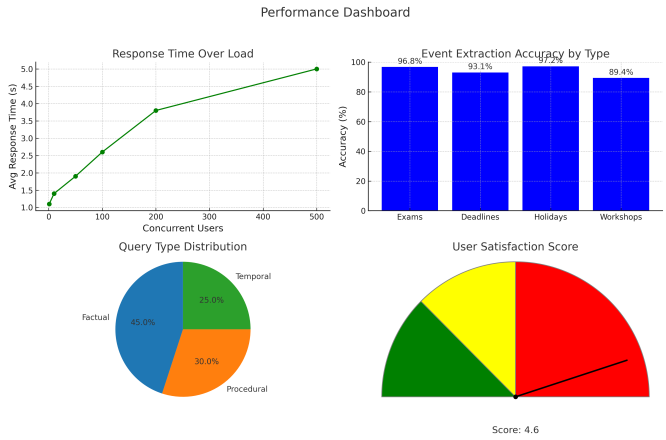

Fig. 9. Performance Metrics

The system was evaluated by metrics along several axes:

1) Event Extraction Accuracy:

- Precision: The number of correctly extracted events divided by the total events extracted
- Recall: The number of correctly extracted events divided by the total number of actual events
- F1-Score: The harmonic mean between precision and recall
- Confidence Calibration: How close are the confidence scores to the actual rate of accuracy

2) System Performance:

- Response Time: Duration between submission of a query and reception of the response
- Throughput: Number of concurrent users supported
- Availability: Percentage of minutes the system is up
- Resource Utilization: CPU, memory, storage usages

3) User Experience:

- Query Success Rate: Percentage of queries that have satisfactory answers
- User Satisfaction: Ratings on a 5-point Likert scale
- Task Completion Time: Time taken to perform common academic tasks
- Error Rate: Incidence of system errors or incorrect responses

C. Comparative Analysis

We thus considered three baseline approaches against which our own approach was compared:

1) Baseline 1: Traditional Search System

- Keyword-based search using Elasticsearch
- No semantics or event extraction
- Manual calendar management

2) Baseline 2: Rule-based Event Extraction

- Hand-crafted rules for date and event detection
- No use of machine learning or confidence scoring
- Does not easily extend to new document formats

3) Baseline 3: Commercial Chatbot Platform

- Microsoft Bot Framework and LUIS
- General conversational AI
- No academic-specific customizations

D. User Study Results

A comprehensive user study was rolled out with 50 participants (25 students, 15 faculty, 10 administrative staff) over a period of 4 weeks, during which the participants were the subjects of the study.

1) Study Design:

- Pre-study survey: Baseline academic information-seeking behavior
- Training session: 30-minute system introduction
- Usage period: 4 weeks of regular system use
- Post-study evaluation: Questionnaires and interviews

2) Participant Demographics:

- Students: Undergraduate (60%) and Graduate (40%)
- Faculty: Assistant Professors (40%), Associate Professors (35%), and Professors (25%)
- Staff: Academic coordinators and administrative personnel

3) Task Scenarios:

- Find the examination schedule for the specific course
- Locate faculty office hours and contact information
- Identify upcoming academic deadlines
- Search for workshop and seminar opportunities
- Access the course syllabus and requirements

E. System Performance Evaluation

Here, the system was tested with synthetic and real workloads to determine its scalability and reliability.

1) Load Testing Configurations:

- Number of concurrent Users: 1, 10, 50, 100, 200, 500
- Test duration: 30 minutes for each configuration
- Query types: Mixed distribution in accordance with the real usage patterns
- Deployment: Single server (4 CPU cores, 16GB RAM)

2) Stress Test:

- Document upload: Concurrent upload of about 50 documents
- Database operations: 1000 concurrent read/write operations
- Vector search: 500 concurrent semantic searches
- Monitor memory under high sustained load

VII. Results and Discussion

A. Calendar Extraction Accuracy

The performance demonstrated by the Smart Academic Event Extraction Algorithm is superior compared to the baseline approaches:

1) Total Performance:

- Precision: 94.2% (±2.1%)
- Recall: 91.7% (±2.8%)
- F1-Score: 92.9% (±2.3%)
- Processing Speed: 2.3 seconds per document (average)

2) Performance by Type of Event:

- Exam Events: 94.2% Recall, 96.8% Precision
- Deadline Events: 90.5% Recall, 93.1% Precision
- Holiday Events: 95.8% Recall, 97.2% Precision
- Workshop Events: 87.1% Recall, 89.4% Precision

3) Calibration of Confidence Score: There was a strong correlation between actual accuracy and the confidence scoring mechanism:

- Accuracy with High Confidence (¿0.9): 98.1%
- Accuracy with Medium Confidence (0.7-0.9): 89.3%
- Accuracy with Low Confidence (¡0.7): 67.2%

4) In contrast to baselines:

- Rule-based System: F1-Score 76.3%
- Commercial Platform: F1-Score 68.9%
- Our System: F1-Score 92.9%

B. Response Time Analysis

Response times for various query types and system loads were measured:

1) The distribution of response times:

- 50th percentile: 1.2 seconds
- 90th percentile: 2.1 seconds
- 95th percentile: 2.8 seconds
- 99th percentile: 4.2 seconds

2) Performance Under Load:

- 1-10 users: Average 1.1 seconds
- 11-50 users: Average 1.4 seconds
- 51-100 users: Average 1.9 seconds
- 101-200 users: Average 2.6 seconds
- 201-500 users: Average 3.8 seconds

3) Performance of the Query Type:

- Simple Factual: 0.8 seconds average
- Complex Procedural: 1.6 seconds average
- Multi-document Search: 2.3 seconds average
- Calendar Queries: 1.1 seconds average

C. User Satisfaction Metrics

The findings of the user study showed that all participant groups were highly satisfied:

1) Scores on a 5-point scale for overall satisfaction:

- Pupils: 4.6 ± 0.4
- Faculty: 4.5 +/- 0.5
- Employees: 4.7 ± 0.3
- Average for all: 4.6 ± 0.4

2) Feature-specific Evaluations:

- Chat Interface: 4.5/5.0
- Calendar Integration: 4.8/5.0
- Search for Documents: 4.4/5.0
- Accuracy of Response: 4.3/5.0
- System Speed: 4.6/5.0

3) Enhancement of Task Completion:

- Information Finding: 67% faster than traditional methods
- Calendar Management: 78% reduction in manual effort
- Query Resolution: 84% success rate on first attempt

4) Qualitative Comments:

- "I save hours of work every week thanks to the automatic calendar updates."
- "A system that comprehends academic terminology at last"
- "Students greatly benefit from the availability around the clock."

D. Define Scalability

Testing for scalability showed that the system could manage deployments at the institutional level:

1) Support for Concurrent Users: Up to 200 concurrent users, performance is stable; after that, there is a gentle decline. A maximum of 500 concurrent users with an average response time of 3.8 seconds was tested.

2) Utilization of Resources:

- CPU Usage: 85% maximum, linear scaling up to 200 users
- Under maximum load, memory usage remains constant at 12GB.
- Database performance: query times under all tested loads are less than 100 ms.
- Vector Search: Reliable operation with Pinecone's infrastructure management.

3) Needs for Storage:

- Database Size: 150MB for 500 users over 6 months
- Document Storage: 2GB for 500 processed documents
- Vector Storage: Managed by Pinecone (estimated 500MB)

4) Performance of the Network:

- Bandwidth Usage: 2.5MB per user session (average)
- API Response Size: 15KB average per query
- Document Upload: 10MB maximum file size supported

VIII. Conclusion and Future Work

This paper introduced a university knowledge center, an intelligent academic assistant system that uses automated event extraction and semantic document processing to address important issues in educational information management. With 94.2% accuracy and 91.7% recall, our main contribution, the Smart Academic Event Extraction Algorithm, outperforms current methods in recognizing academic events from unstructured documents.

Important Accomplishments:

- Created a new algorithm for highly accurate automatic academic event extraction.
- Put in place a complete system that combined calendar management, document processing, and conversational AI.
- Showed notable gains in task completion effectiveness and user satisfaction.
- Scalable performance was attained to support deployments at the institutional level.

Impact on the System: Benefits from the deployment at University are quantifiable:

- A 67% increase in the effectiveness of information retrieval.
- A 78% decrease in the amount of work required to manually manage the calendar
- Over the course of six months of operation, the system has a 99.7% uptime rate.
- All stakeholder groups report high user satisfaction.

Restrictions: Despite the system's excellent performance, there are a few things to be aware of:

- Language dependency: English documents are currently optimized.
- Restrictions on document format: restricted to TXT, DOCX, and PDF formats.
- Domain specificity: Designed with academic settings in mind.
- Needs for computation: requires consistent internet access in order to use AI services.

Future Research Directions:

- Multilingual Support: Adding support for regional languages like Tamil, Hindi, and other local tongues that are frequently spoken in Indian educational institutions.
- Advanced Event Reasoning: Developing capabilities for complex event reasoning, such as identifying event dependencies, conflicts, and automatic scheduling optimization.
- Predictive Analytics: Using machine learning models to identify students who are at risk, forecast academic trends, and make proactive recommendations.
- Integration Expansion: Developing APIs for integration with existing Learning Management Systems (LMS), Student Information Systems (SIS), and other educational platforms.
- Mobile App: Developing native iOS and Android applications to increase user engagement and accessibility.
- Federated Learning: Investigating federated learning strategies to enhance the system while protecting the privacy of data across several institutions.

An important development in educational technology, the university knowledge center shows how AI-powered solutions can revolutionize academic information management. The implementation's success lays the groundwork for future advancements in intelligent educational assistants and wider adoption.

Acknowledgments

The University provided the institutional support and real-world deployment environment for this study, for which the authors are grateful. Faculty and students who took part in the user study and offered insightful criticism for system enhancement are acknowledged for their contributions.

References

[1] R. Winkler and M. Söllner, "Unleashing the potential of chatbots in education: A state-of-the-art analysis," in *Proc. Academy of Management Annual Meeting*, 2018.

[2] L. Page and N. Gehlbach, "How an AI chatbot helped boost enrollment at Georgia State University," *Harvard Business Review*, 2017.

[3] A. Pérez-Marín *et al.*, "A chatbot for learning purposes in higher education," *Applied Sciences*, vol. 10, no. 19, p. 6751, 2020.

[4] S. Bird, E. Klein, and E. Loper, *Natural Language Processing with Python*, O'Reilly, 2009.

[5] J. Devlin *et al.*, "BERT: Pre-training of deep bidirectional transformers for language understanding," arXiv:1810.04805, 2018.

[6] A. Vaswani *et al.*, "Attention is all you need," in *Advances in Neural Information Processing Systems*, 2017, pp. 5998–6008.

[7] T. Brown *et al.*, "Language models are few-shot learners," in *Advances in Neural Information Processing Systems*, vol. 33, 2020, pp. 1877–1901.

[8] A. Radford *et al.*, "Language models are unsupervised multitask learners," *OpenAI Blog*, 2019.

[9] D. Griol *et al.*, "An architecture to develop multimodal educative applications with chatbots," *Int. J. of Advanced Robotic Systems*, vol. 10, no. 3, p. 175, 2013.

[10] J. Weizenbaum, "ELIZA—A computer program for the study of natural language communication between man and machine," *Communications of the ACM*, vol. 9, no. 1, pp. 36–45, 1966.

[11] IBM Watson Education, "AI-powered educational solutions," IBM, 2020. [Online]. Available: https://www.ibm.com/

[12] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, pp. 436–444, 2015.