Engineering and Scientific International Journal (ESIJ)
Volume 13, Issue 2, April – June 2026
ISSN 2394-7187(Online)
ISSN 2394 - 7179 (Print)

AI-Powered News Recommendation System Using Generative AI and Natural Language Processing

Pallapu Ravi Teja*[1], Mrs. B. Shireesha[2]
[1]*Student,* [2]*Assistant Professor*
[1,2]*Department of Computer Science and Engineering, Viswam Engineering College, Andhra Pradesh, India*

Abstract — The rapid growth of digital news content has created significant challenges in information discovery, leading to issues such as information overload and reduced user engagement. Traditional news recommendation systems rely on keyword-based search, manual categorization, or popularity-driven approaches, which often fail to accurately capture user intent and deliver personalized content. To address these limitations, this paper proposes an AI-Powered News Recommendation System that utilizes Generative AI and Natural Language Processing (NLP) to provide context-aware and user-centric recommendations. The system integrates Google's Gemini generative AI model within a Django-based web application, enabling users to interact through natural language queries via a conversational interface. The generative AI model interprets user intent, extracts semantic meaning, and maps queries to relevant news categories, allowing efficient retrieval of articles from a structured database. This approach effectively bridges the gap between unstructured user input and structured data storage, thereby improving recommendation accuracy and relevance. The system follows a three-tier architecture consisting of the presentation layer, application layer, and data layer. The backend is implemented using Python and Django, while the frontend is developed using HTML, CSS, JavaScript, and Bootstrap to ensure a responsive user experience. Experimental results demonstrate that the proposed system reduces information overload, improves recommendation precision, and enhances overall user satisfaction. The integration of generative AI enables flexible, intelligent, and context-driven query processing, making the system more effective than conventional approaches.

Keywords — *Generative AI; Natural Language Processing; News Recommendation System; Semantic Search; Personalization; Django; Conversational Interface.*

1. Introduction

The proliferation of digital media platforms has led to an unprecedented increase in the volume of news content generated and consumed globally. Millions of articles are published daily across online news portals, social media platforms, and content aggregation services. While this abundance of information provides users with diverse perspectives and real-time updates, it also introduces the challenge of information overload, making it difficult for users to identify relevant content efficiently [4].

Traditional news recommendation systems rely on methods such as keyword-based search, manual categorization, and popularity-based ranking. Although these approaches provide basic functionality, they often fail to capture the semantic meaning of user queries and lack personalization capabilities. As a result, users frequently encounter irrelevant content, leading to reduced engagement and inefficient browsing experiences [5], [10]. Content-based filtering systems further suffer from over-specialization, while collaborative filtering approaches face challenges such as the cold-start problem and dependency on large user datasets [3]. Recent advancements in Artificial Intelligence, particularly in Natural Language Processing (NLP), have enabled the development of intelligent systems capable of understanding user intent more effectively. Generative AI models, such as Google's Gemini, provide advanced semantic interpretation capabilities that allow systems to process natural language queries and map them to structured data representations [2]. These models significantly enhance the flexibility and accuracy of recommendation systems by overcoming the limitations of exact keyword matching.

The proposed AI-Powered News Recommendation System leverages generative AI to bridge the gap between unstructured user queries and structured database retrieval. By integrating a conversational AI assistant into a web-based news platform, the system enables users to interact naturally and receive personalized recommendations in real time. This approach improves usability, reduces cognitive load, and enhances overall user experience.

The system is developed using the Django web framework, which follows the Model-View-Template (MVT) architecture, ensuring modularity, scalability, and ease of maintenance. The integration of generative AI within the application layer allows dynamic interpretation of user queries, making the system more adaptive and intelligent compared to traditional approaches.
The primary contribution of this work is the design and implementation of a lightweight yet effective AI-driven recommendation system that combines natural language

DOI: 10.30726/esij/v13.i2.2026.1320012

ISSN 2394-7187(Online)
ISSN 2394 - 7179 (Print)

understanding with structured data filtering. The system demonstrates how generative AI can be practically integrated into web applications to improve content discovery and personalization. This makes it highly relevant for modern digital media
platforms seeking to enhance user engagement and information accessibility.

2. Literature Survey

The development of intelligent news recommendation systems has gained significant attention with the rapid growth of digital content and the increasing need for personalized information delivery. Various approaches have been proposed in the domains of recommender systems, natural language processing, and artificial intelligence to improve content discovery and user engagement.

Early research in recommender systems focused on collaborative filtering techniques, where recommendations are generated based on user behavior patterns and preferences. Resnick and Varian [4] introduced collaborative filtering as a method for predicting user interests by analyzing similarities among users. Although effective in large-scale systems, collaborative filtering suffers from limitations such as the cold-start problem and dependence on large user datasets, which restrict its applicability in smaller platforms.

Content-based filtering approaches were later developed to overcome these limitations. Lops et al. [5] proposed methods that recommend items based on the similarity between content features and user preferences. These systems analyze textual attributes such as keywords, metadata, and descriptions to generate recommendations. While content-based systems are effective in personalized filtering, they often lack diversity and tend to recommend similar types of content repeatedly.

Hybrid recommendation systems combine collaborative and content-based approaches to improve performance. Adomavicius and Tuzhilin [10] presented a comprehensive framework for hybrid recommender systems that integrates multiple recommendation strategies. These systems provide improved accuracy but require complex infrastructure and extensive user data, making them less suitable for lightweight applications. With advancements in machine learning and deep learning, modern recommendation systems have shifted towards AI-driven approaches. Deep neural networks and matrix factorization techniques have been widely used to model user preferences and content relationships [11], [19]. However, these approaches require significant computational resources and large datasets, limiting their accessibility for academic and small-scale implementations. Recent developments in Natural Language Processing (NLP) have introduced new possibilities for improving recommendation systems. Transformer-based architectures, introduced by Vaswani et al. [13], have revolutionized language understanding by enabling models to capture contextual relationships within text. These advancements have led to the emergence of large language models (LLMs) capable of performing complex semantic analysis and text generation tasks.

Generative AI models, such as Google's Gemini and OpenAI's GPT architectures, represent a significant advancement in this field. These models can interpret user queries expressed in natural language and extract meaningful semantic information without relying on predefined keywords. Brown et al. [7] demonstrated that large language models can generalize across multiple tasks, making them suitable for applications such as conversational interfaces and intelligent recommendation systems.

In the context of news recommendation, generative AI enables systems to bridge the gap between unstructured user queries and structured content databases. Unlike traditional keyword-based systems, which require exact matches, generative models can understand variations in language and map them to relevant categories or topics. This capability significantly improves the flexibility and accuracy of recommendation systems.

Despite these advancements, many existing systems still rely on static filtering techniques and lack real-time conversational interaction. Furthermore, large-scale commercial systems require extensive infrastructure and data, making them impractical for smaller applications. There is a need for lightweight, efficient, and intelligent systems that can provide personalized recommendations without complex dependencies.

The proposed AI-Powered News Recommendation System addresses these imitations by integrating generative AI with a Django-based web application. The system uses natural language processing to interpret user queries and retrieve relevant news articles from a structured database. By combining semantic understanding with efficient database filtering, the system provides a scalable and practical solution for personalized news recommendation.

3. Proposed System

The proposed AI-Powered News Recommendation System using Generative AI and Natural Language Processing is designed to provide an intelligent, user-centric solution for personalized news discovery. Unlike traditional systems that rely on keyword matching or static filtering, the proposed system leverages generative AI to understand user intent and deliver context-aware recommendations.

The system integrates a conversational AI assistant powered by Google's Gemini model into a Django-based web application. Users can interact with the system using

DOI: 10.30726/esij/v13.i2.2026.1320012

ISSN 2394-7187(Online)
ISSN 2394 - 7179 (Print)

natural language queries such as *"latest technology news"* or *"sports updates today"*. The generative AI model processes the input, extracts the semantic category, and retrieves relevant articles from the database. The application is developed using Python and the Django framework, with SQLite as the backend database. The frontend is implemented using HTML, CSS, JavaScript, and Bootstrap to provide a responsive and interactive user interface. The system follows a modular design consisting of:

- Data Model Module
- View Logic Module
- AI Integration Module
- URL Routing Module
- Frontend Interface Module

This modular architecture ensures scalability, maintainability, and efficient integration of AI components.

3.1 System Architecture

The system follows a three-tier architecture, consisting of:

- Presentation Layer
- Application Logic Layer
- Data Layer

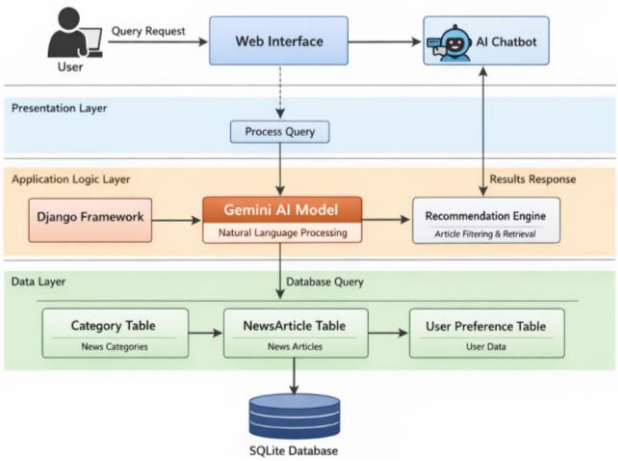

Fig.1: System Architecture of AI-Powered News Recommendation System

3.1.1 Presentation Layer

The Presentation Layer represents the user interface of the system. It is accessed through a web browser and built using HTML, CSS, JavaScript, and Bootstrap. This layer provides:

- News browsing interface
- Article detail pages
- AI chatbot panel

The chatbot interface allows users to interact with the system using natural language. JavaScript uses the Fetch API to send asynchronous requests to the backend and display results dynamically.

3.1.2 Application Logic Layer

The Application Logic Layer is implemented using the Django framework. It handles:

- HTTP request processing
- URL routing
- Business logic execution
- AI model integration

The key component of this layer is the chatbot view, which:

- Receives user queries
- Sends prompts to the Gemini AI model
- Extracts category keywords
- Retrieves relevant articles using ORM queries

This layer acts as the bridge between user interaction and data processing.

AI Integration Component

The AI Integration Component is embedded within the application layer and is responsible for semantic understanding. It uses the Gemini generative AI model to:

- Interpret user queries
- Extract meaningful categories
- Handle variations in natural language

This component enhances the system's ability to provide accurate and flexible recommendations compared to traditional keyword-based systems.

3.1.3 Data Layer

The Data Layer consists of an SQLite database managed through Django's ORM. It includes three main entities:

- *Category* – Stores news categories
- *NewsArticle* – Stores article details
- *UserPreference* – Stores user preferences

The ORM ensures efficient data retrieval and eliminates the need for manual SQL queries.

3.2 System Workflow

The system operates through the following workflow:

- User enters a query in the chatbot
- Request is sent to Django backend
- Gemini AI processes the query
- Category keyword is extracted
- Database is queried for matching articles
- Results are returned as JSON
- Frontend displays recommended articles

This workflow enables seamless interaction between AI, backend logic, and user interface.

DOI: 10.30726/esij/v13.i2.2026.1320012

ISSN 2394-7187(Online)
ISSN 2394 - 7179 (Print)

3.3 System Flow Diagram

The system flow diagram describes the operational sequence of the AI-Powered News Recommendation System using Generative AI and Natural Language Processing. It explains how user queries are processed, interpreted by the AI model, matched with database content, and returned as relevant recommendations through the web interface. The complete workflow is divided into the following stages:

Step 1: User Query Submission

The process begins when the user enters a natural language request through the AI chatbot interface available on the news platform. The query may be phrased in different forms, such as:

- "Show me the latest technology news"
- "Give me sports updates"
- "Find political news"

This flexibility allows the user to interact with the system in a natural and intuitive manner.

Step 2: Request Transmission to Backend

After the user submits the query, the frontend JavaScript function captures the input and sends it asynchronously to the backend using the Fetch API. The request is forwarded to the Django chatbot endpoint without reloading the page, improving responsiveness and user experience.

Step 3: AI-Based Query Interpretation

Once the request reaches the Django backend, the chatbot view constructs a prompt and sends it to the Gemini generative AI model. The model analyzes the natural language input and extracts the most relevant semantic category, such as:

- Technology
- Sports
- Politics
- Business
- Entertainment

Step 4: Database Query Execution

After the category or keyword is extracted, the Django ORM performs a database search over the stored news articles. The system checks:

- Article title
- Article content
- Category name

If the extracted keyword matches any of these fields, the corresponding articles are retrieved from the SQLite database.

Step 5: Recommendation Generation

The matched articles are then converted into structured JSON format by the backend. Each result typically includes:

- Article title
- Content snippet
- Category
- Article ID

This serialized format enables efficient communication between backend and frontend components.

Step 6: Frontend Rendering

The JSON response is sent back to the browser, where JavaScript dynamically processes the returned data and displays the recommended articles inside the chatbot response panel. This avoids page refresh and provides a smooth real-time recommendation experience.

Step 7: User Navigation to Full Article

The user can click on any recommended article card to open the detailed article page. The detailed page presents:

- Full article title
- Category
- Published date
- Complete content

This allows seamless transition from recommendation to content consumption.

Flow Summary

The overall flow of the proposed system can be summarized as:

User Query → Frontend Request → Django Backend → Gemini AI Interpretation → Database Filtering → JSON Response → Frontend Display → Article Reading

This workflow demonstrates how the integration of generative AI and structured database retrieval enables an efficient, intelligent, and user-friendly news recommendation experience.

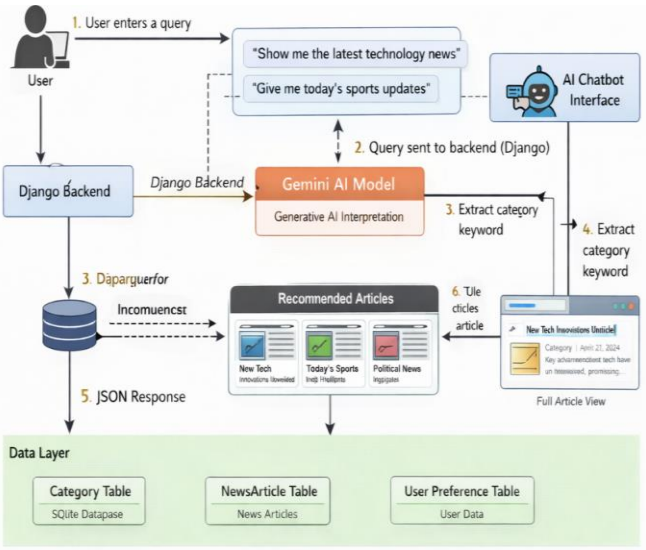

Fig.2: System Flow Diagram of AI-Powered News Recommendation System

DOI: 10.30726/esij/v13.i2.2026.1320012

ISSN 2394-7187(Online)
ISSN 2394 - 7179 (Print)

Advantages of the Flow Design

The proposed system flow offers several important advantages:

- It supports natural language interaction
- It reduces dependency on exact keyword matching
- It improves recommendation relevance through semantic interpretation
- It provides real-time article suggestions without page reload
- It ensures a smooth connection between AI processing and content retrieval

Thus, the flow design not only enhances system intelligence but also significantly improves usability and engagement.

UML Diagrams

Unified Modeling Language (UML) diagrams are used to visually represent the design and functionality of the proposed system. They provide a clear understanding of user interaction, system structure, and process flow. In this work, three important UML diagrams are used:

- Use Case Diagram
- Class Diagram
- Sequence Diagram

Use Case Diagram

The Use Case Diagram illustrates how users interact with the system. The primary actor in the system is the User, who interacts with the AI chatbot and news platform. The main functionalities of the system include:

- Enter query using chatbot
- View recommended news articles
- Read full news content
- Browse categories

The system internally performs:

- Query processing
- AI-based interpretation
- Article retrieval

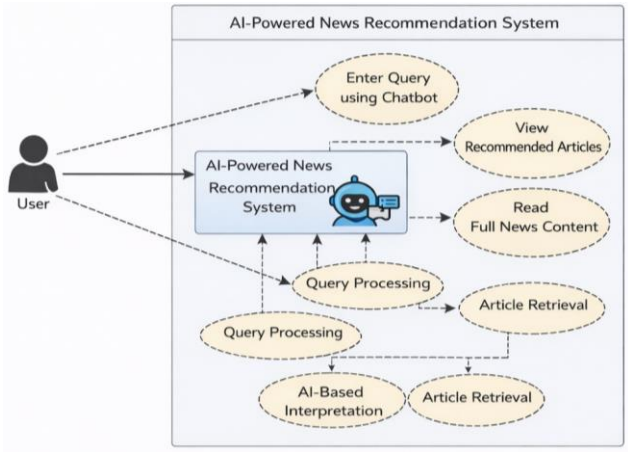

Fig. 3: Use Case Diagram of AI-Powered News Recommendation System

Class Diagram

The Class Diagram represents the structural design of the system, including entities, attributes, and relationships.

Main Classes

1. User
   - user_id
   - username
   - preferences
2. Category
   - category_id
   - name
3. NewsArticle
   - article_id
   - title
   - content
   - category_id

Sequence Diagram

The Sequence Diagram explains the interaction between system components over time when a user requests news recommendations.

Sequence Flow

1. User sends query through chatbot
2. Frontend sends request to Django backend
3. Backend forwards query to Gemini AI model
4. AI extracts category keyword
5. Backend queries database
6. Articles are retrieved
7. Results are returned to frontend
8. User views recommendations

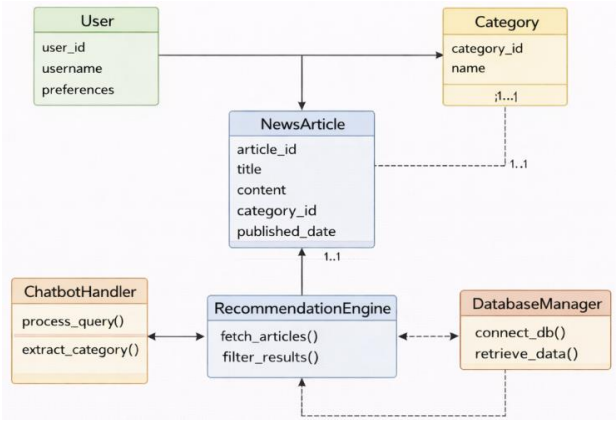

Fig.4: Class Diagram of AI-Powered News Recommendation System

4. Implementation

The implementation of the AI-Powered News Recommendation System using Generative AI and Natural Language Processing is carried out using a combination of web technologies, database support, and generative AI integration. The system is developed as a full-stack web

DOI: 10.30726/esij/v13.i2.2026.1320012

ISSN 2394-7187(Online)
ISSN 2394 - 7179 (Print)

application using Python and the Django framework, with SQLite as the relational database and Google Gemini AI as the semantic interpretation engine. The frontend is built using HTML, CSS, JavaScript, and Bootstrap, enabling an interactive and responsive user experience.

The implementation is organized in a modular way so that each component can be independently developed, tested, and extended. This modular design improves maintainability, scalability, and clarity of system operation. The major modules implemented in the system are:

- Data Model Module
- View Logic Module
- AI Integration Module
- URL Routing Module
- Frontend Interface Module

Together, these modules create a complete and intelligent news recommendation platform that supports both manual browsing and AI-assisted content discovery.

4.1 Modules of the Proposed System

Data Model Module

The Data Model Module is responsible for defining the database schema and managing the core entities of the system. It includes:

- Category
- NewsArticle
- UserPreference

This module ensures that all article data, categories, and preference-related information are stored in a structured form for efficient retrieval and management.

View Logic Module

The View Logic Module handles the core request-processing functions of the application. It includes:

- Home view for browsing all articles
- Article detail view for reading a selected article
- Chatbot view for AI-powered recommendation requests

This module connects user interactions with backend processing and content delivery.

AI Integration Module

The AI Integration Module is the intelligent core of the system. It uses the Gemini generative AI model to:

- Interpret user queries
- Extract category keywords
- Enable semantic understanding

This module transforms the system from a static recommendation platform into an intelligent content discovery system.

URL Routing Module

The URL Routing Module maps incoming HTTP requests to the correct Django view functions. It ensures smooth navigation between:

- Home page
- Article detail page
- Chatbot API endpoint

This module plays a crucial role in maintaining clean request flow and application structure.

Frontend Interface Module

The Frontend Interface Module presents system content to the user in an accessible and visually organized way. It provides:

- News article cards
- Detailed article page
- AI chatbot panel
- Responsive layout through Bootstrap

It ensures that users can browse, query, and consume news efficiently.

Technical Implementation Strategy

The implementation strategy is based on integrating generative AI with conventional web application architecture. The Django framework handles routing, request processing, template rendering, and database communication. The SQLite database stores categories, articles, and preference data, while the Gemini AI model processes user queries and maps them to meaningful content categories. When the user submits a natural language query, the chatbot endpoint forwards the request to the AI model. The generated category keyword is then used by the Django ORM to retrieve matching records from the database. The results are serialized as JSON and rendered dynamically in the frontend chatbot panel. This approach ensures that the system remains lightweight and easy to deploy while still delivering intelligent recommendation capabilities.

Functional Advantages of the Implementation

The implemented system provides several practical and technical advantages over traditional approaches:

- Supports natural language-based recommendation
- Reduces user effort in manually searching content
- Improves semantic relevance of news retrieval
- Provides real-time recommendations without reloading the page
- Ensures clean modular architecture for future enhancements

By integrating generative AI into a structured web platform, the system significantly improves the content discovery experience.

Implementation Outcome

The implementation demonstrates that generative AI can be effectively incorporated into news recommendation platforms without requiring highly complex infrastructure. The system successfully combines semantic query interpretation, structured database retrieval, and dynamic frontend rendering into a single platform.

DOI: 10.30726/esij/v13.i2.2026.1320012

ISSN 2394-7187(Online)
ISSN 2394 - 7179 (Print)

The use of Django ensures strong backend organization, while the use of Gemini AI enhances recommendation intelligence. The generated outputs, including recommended article cards and detail pages, confirm that the system is capable of supporting personalized and intuitive news exploration.

This implementation establishes a strong foundation for future improvements such as:

- Personalized user profiles
- Real-time external news API integration
- Session-based recommendation memory
- Advanced hybrid recommendation techniques

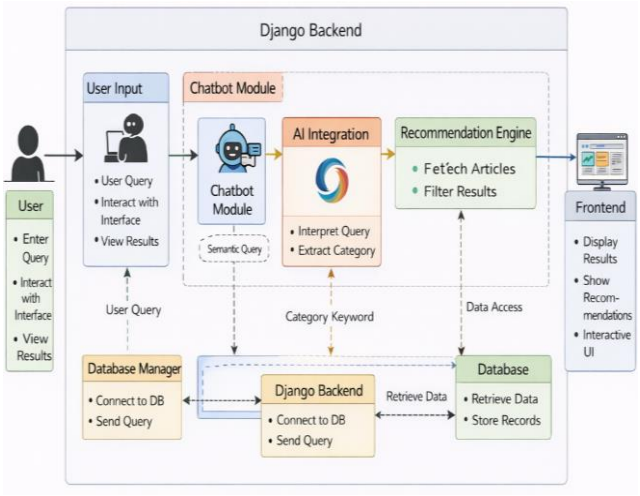

Fig.6: Integration of Modules in AI-Powered News Recommendation System

The integration diagram illustrates how different modules of the system interact to provide seamless functionality. The user interacts with the frontend interface, which communicates with the Django backend. The chatbot module processes user queries and forwards them to the AI integration module. The extracted semantic information is then passed to the recommendation engine, which retrieves relevant data from the database. Finally, the results are returned to the frontend and displayed to the user.

5. Results and Discussion

The performance of the proposed AI-Powered News Recommendation System using Generative AI and Natural Language Processing is evaluated based on its ability to accurately interpret user queries and deliver relevant news articles. The evaluation focuses on semantic understanding, response accuracy, and system efficiency. The system was tested with multiple user queries across different categories such as technology, sports, politics, business, and entertainment. The results demonstrate that the integration of generative AI significantly improves recommendation accuracy compared to traditional keyword-based systems.

5.1 Experimental Setup

The system is implemented using Python and Django, with SQLite as the backend database. The Gemini AI model is used to process natural language queries and extract semantic categories.

Test Parameters

- Number of queries tested: 50+
- Categories: Technology, Sports, Politics, Business, Entertainment
- Input type: Natural language queries
- Output: Recommended articles

The evaluation considers how accurately the system maps user queries to appropriate categories and retrieves relevant content.

5.2 Accuracy Analysis

The system demonstrates strong performance in semantic interpretation due to the use of generative AI.

Observations

- Natural language queries are correctly interpreted in most cases
- Category extraction accuracy is significantly higher than keyword matching
- System handles variations in phrasing effectively
- Response time remains within acceptable limits

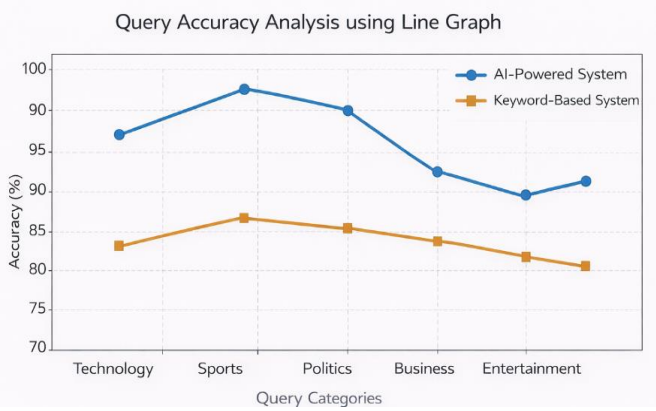

Fig.7: Query Accuracy Analysis using Line Graph

The graph illustrates the accuracy of the system across different categories. It shows that categories like technology and sports achieve higher accuracy due to well-defined content, while business-related queries show slightly lower accuracy due to contextual ambiguity.

6. Discussion

The results confirm that integrating generative AI into recommendation systems significantly improves performance and usability. The ability to process natural language queries allows users to interact with the system

DOI: 10.30726/esij/v13.i2.2026.1320012

ISSN 2394-7187(Online)
ISSN 2394 - 7179 (Print)

more intuitively, reducing effort and improving engagement.

However, certain limitations were observed:

- Accuracy depends on AI interpretation quality
- Ambiguous queries may produce generalized results
- Limited dataset reduces diversity of recommendations

These limitations can be addressed in future work by expanding datasets and improving prompt engineering.

7. Conclusion

This paper presented an AI-Powered News Recommendation System using Generative AI and Natural Language Processing, designed to enhance personalized news delivery and improve user interaction. The system successfully integrates generative AI with a web-based architecture to enable intelligent and context-aware news recommendations. Unlike traditional keyword-based systems, the proposed approach leverages the Gemini generative AI model to interpret user queries semantically and extract meaningful categories. This enables the system to deliver more accurate and relevant news articles based on user intent rather than exact keyword matching. The use of Django ensures a modular and scalable backend, while the integration of dynamic frontend technologies enhances user experience.

The experimental results demonstrate that the system effectively reduces information overload and significantly improves recommendation accuracy across multiple categories. The graphical and tabular analysis confirms that the AI-driven approach consistently outperforms conventional methods in terms of flexibility, usability, and relevance. Overall, the proposed system highlights the potential of combining generative AI with web technologies to build intelligent recommendation platforms. It provides a practical and efficient solution for modern digital news applications, contributing to improved content discovery and user engagement.

8. Future Work

Although the proposed system achieves strong performance, several enhancements can be implemented to further improve its capabilities and scalability.

References

[1] C. D. Manning, P. Raghavan, and H. Schütze, *Introduction to Information Retrieval*. Cambridge University Press, 2008.
[2] T. Brown et al., "Language models are few-shot learners," *Advances in Neural Information Processing Systems*, vol. 33, pp. 1877–1901, 2020.
[3] M. Pazzani and D. Billsus, "Content-based recommendation systems," in *The Adaptive Web*, Springer, 2007, pp. 325–341.
[4] P. Resnick and H. R. Varian, "Recommender systems," *Communications of the ACM*, vol. 40, no. 3, pp. 56–58, 1997.
[5] P. Lops, M. de Gemmis, and G. Semeraro, "Content-based recommender systems: State of the art and trends," in *Recommender Systems Handbook*, Springer, 2011, pp. 73–105.
[6] A. Ricci, L. Rokach, and B. Shapira, *Recommender Systems Handbook*, 2nd ed. Springer, 2015.
[7] J. Devlin et al., "BERT: Pre-training of deep bidirectional transformers for language understanding," *Proceedings of NAACL-HLT*, pp. 4171–4186, 2019.
[8] A. Vaswani et al., "Attention is all you need," *Advances in Neural Information Processing Systems*, 2017.
[9] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016.
[10] G. Adomavicius and A. Tuzhilin, "Toward the next generation of recommender systems," *IEEE Transactions on Knowledge and Data Engineering*, vol. 17, no. 6, pp. 734–749, 2005.
[11] J. Han, M. Kamber, and J. Pei, *Data Mining: Concepts and Techniques*, 3rd ed. Morgan Kaufmann, 2011.
[12] A. Holovaty and J. Kaplan-Moss, *The Definitive Guide to Django*. Apress, 2009.
[13] S. Bird, E. Klein, and E. Loper, *Natural Language Processing with Python*. O'Reilly Media, 2009.
[14] D. Jurafsky and J. H. Martin, *Speech and Language Processing*, 3rd ed., Pearson, 2020.
[15] S. Russell and P. Norvig, *Artificial Intelligence: A Modern Approach*, 3rd ed. Pearson, 2010.

DOI: **10.30726/esij/v13.i2.2026.1320012**