IJCRT.ORG ISSN : 2320-2882

INTERNATIONAL JOURNAL OF CREATIVE RESEARCH THOUGHTS (IJCRT)

An International Open Access, Peer-reviewed, Refereed Journal

Auto Newsroom Agent System

[1]Pragnya S, [2]Baby Sony, [3]Sai Charan, [4]Muskan A

[1]Student, [2]Student, [3]Student, [4]Assistant Professor

[1]Dept of Computer Science,

[1]Navodaya Institute Of Technology, Raichur, India

Abstract: The rapid expansion of digital media has significantly increased the demand for timely, accurate, and structured news content. Traditional newsroom workflows rely heavily on manual writing, editing, and verification processes, which are time-consuming and prone to human fatigue and bias. To address these limitations, this paper presents an **Auto Newsroom Agent System**, an intelligent multi-agent framework designed to automate the end-to-end news generation pipeline.

The proposed system employs a **Multi-Agent System (MAS)** architecture consisting of four specialized agents: a **Writer Agent** based on the GPT-2 transformer model for coherent article generation, a **Fact-Checker Agent** utilizing a RoBERTa-based classifier for identifying potential misinformation or sensitive content, a **Headline Generator Agent** for concise and context-aware headline creation, and a central **Editor Agent** that orchestrates the overall workflow. The system accepts user-provided topics or RSS feed summaries, generates structured news articles, verifies content integrity, and stores finalized outputs in a persistent SQLite database through a Flask-based web interface.

Experimental evaluation demonstrates that the system can generate complete news articles within seconds while maintaining structural consistency and basic content verification. The modular design ensures scalability and extensibility, enabling future integration of advanced fact-checking and multimedia generation components. The Auto Newsroom Agent System highlights the potential of AI-driven automation in digital journalism, offering an efficient and cost-effective solution for modern news production.

***Index Terms** - Automated Journalism, Multi-Agent Systems, Natural Language Processing, GPT-2, RoBERTa, Transformer Models, News Generation, Artificial Intelligence*

I. Introduction

The exponential growth of digital media platforms has transformed the way news is produced, distributed, and consumed. With the advent of social media, online news portals, and mobile applications, audiences now expect **real-time access to accurate and well-structured news content**. This increasing demand has placed immense pressure on traditional newsrooms, where journalists and editors must operate under strict deadlines while maintaining content quality, neutrality, and factual accuracy. As a result, manual news production workflows often suffer from delays, human bias, and operational inefficiencies.

Artificial Intelligence (AI) and Natural Language Processing (NLP) have emerged as powerful tools capable of automating various aspects of content creation. Early automated journalism systems relied on **rule-based or template-driven approaches**, which lacked flexibility and were limited to narrowly structured data inputs. Recent advancements in **Transformer-based Large Language Models (LLMs)** have significantly improved the ability of machines to generate coherent, context-aware, and human-like text. Models such as GPT (Generative Pre-trained Transformer) and BERT-based architectures have demonstrated remarkable success in text generation, classification, and summarization tasks.

Despite these advancements, fully automated news generation presents critical challenges. Generative models are prone to **hallucinations**, producing fluent but factually incorrect content. Additionally, single-model systems often struggle to balance creativity with verification, making them unsuitable for sensitive domains such as journalism, where credibility is paramount. Therefore, there is a growing need for structured AI systems that not only generate content but also incorporate **verification, editorial control, and workflow management**, similar to a real-world newsroom.

To address these challenges, this paper proposes an **Auto Newsroom Agent System**, a **Multi-Agent System (MAS)** designed to emulate the collaborative workflow of a professional newsroom. The system decomposes the complex task of news production into specialized agents, each responsible for a specific function. A **Writer Agent** generates news articles using a transformer-based language model, a **Fact-Checker Agent** evaluates the generated content for potential misinformation, a **Headline Generator Agent** produces concise and relevant titles, and an **Editor Agent** orchestrates the entire pipeline to ensure consistency and reliability.

By leveraging the strengths of both generative and discriminative transformer models within a modular architecture, the proposed system aims to improve efficiency, reduce human effort, and enhance content quality in automated journalism. This research demonstrates how AI-driven multi-agent frameworks can serve as effective assistants in modern newsrooms, offering a scalable and cost-efficient solution for automated news generation.

II. PROBLEM STATEMENT AND MOTIVATION

2.1 Problem Statement

The modern news ecosystem demands rapid generation and dissemination of information while maintaining accuracy, neutrality, and credibility. Traditional newsroom workflows rely heavily on manual processes, including topic selection, content drafting, fact verification, headline creation, and editorial review. These processes are **time-consuming, resource-intensive, and difficult to scale**, especially in environments that require continuous 24/7 news coverage.

While recent advances in Artificial Intelligence have enabled automated text generation through Large Language Models (LLMs), existing automated news generation systems suffer from several limitations. Single-model approaches often generate fluent and coherent text but lack **structured verification mechanisms**, making them prone to factual inaccuracies and hallucinations. Additionally, such systems do not reflect the collaborative editorial workflow followed in real-world newsrooms, where multiple roles such as writers, editors, and fact-checkers work together to ensure content quality.

Furthermore, most automated solutions focus solely on content generation without addressing **journalistic structure**, including neutral tone, logical flow, and effective headline synthesis. The absence of integrated verification and editorial control reduces trust in AI-generated news, limiting its adoption in professional journalism. Hence, there is a need for an intelligent system that can automate news production while preserving reliability, structure, and editorial discipline.

2.2 Motivation

The motivation behind the Auto Newsroom Agent System arises from the growing necessity to bridge the gap between **speed and credibility** in digital journalism. With the exponential increase in online content consumption, news organizations require systems that can generate high-quality articles within seconds without compromising factual integrity. Automating repetitive newsroom tasks can significantly reduce human workload and operational costs while enabling journalists to focus on investigative and analytical reporting.

Advancements in Transformer-based models such as GPT-2 and RoBERTa provide an opportunity to design systems that combine **generative creativity with discriminative verification**. By organizing these models within a **Multi-Agent System (MAS)** framework, it becomes possible to simulate a real newsroom environment where specialized agents collaborate to achieve a common goal. This approach improves modularity, scalability, and interpretability compared to monolithic AI solutions.

The proposed system is motivated by the need to create a **scalable, cost-effective, and extensible automated journalism platform** that can assist digital media houses, independent journalists, and researchers. By incorporating automated fact-checking, editorial orchestration, and structured content generation, the Auto

Newsroom Agent System aims to enhance trust in AI-generated news and demonstrate the practical applicability of multi-agent AI architectures in real-world journalism scenarios.

III. LITERATURE REVIEW

The concept of automated content generation has evolved significantly over the past few decades, driven by advancements in Artificial Intelligence, Natural Language Processing (NLP), and Machine Learning. This section reviews existing research related to automated journalism, transformer-based language models, fact-checking mechanisms, and multi-agent systems, highlighting their contributions and limitations in the context of automated news generation.

3.1 Automated Journalism Systems

Early automated journalism systems were predominantly **rule-based and template-driven**. These systems relied on predefined linguistic structures into which numerical or textual data was inserted. Leppänen et al. demonstrated that such systems were effective for structured domains like sports scores and financial reports but lacked flexibility, creativity, and adaptability to diverse news topics. Although accurate, these approaches failed to handle unstructured data and complex narratives, limiting their applicability in modern journalism.

With the rise of data-driven approaches, automated journalism began integrating machine learning techniques to improve adaptability. However, early statistical models still struggled with contextual understanding and natural language fluency. These limitations necessitated the transition toward deep learning-based approaches capable of learning semantic relationships from large text corpora.

3.2 Transformer-Based Language Models

The introduction of the **Transformer architecture** by Vaswani et al. marked a major breakthrough in NLP. Unlike Recurrent Neural Networks (RNNs) and Long Short-Term Memory (LSTM) models, transformers utilize a self-attention mechanism that enables parallel processing of text and efficient modeling of long-range dependencies. This advancement laid the foundation for Large Language Models (LLMs) capable of generating high-quality text.

Radford et al. introduced **GPT-2**, a generative pre-trained transformer model trained on large-scale internet text data. GPT-2 demonstrated the ability to generate coherent, context-aware, and grammatically correct multi-paragraph text, making it suitable for applications such as story generation and news writing. However, GPT-2 operates as a **decoder-only model** and lacks intrinsic mechanisms for verifying factual correctness, which poses challenges for sensitive domains like journalism.

3.3 Fact-Checking and Content Verification

As generative models became more powerful, concerns regarding **hallucinations and misinformation** gained prominence. BERT-based architectures introduced bidirectional contextual understanding, significantly improving performance in classification and verification tasks. Liu et al. proposed **RoBERTa**, an optimized version of BERT, which improved training efficiency and achieved state-of-the-art results on various NLP benchmarks.

Several studies have explored automated fact-checking by framing it as a text classification problem. These systems assess the credibility, sentiment, or consistency of generated text. While effective, most existing systems operate independently of the content generation process, resulting in fragmented workflows. This highlights the need for integrated architectures where verification is embedded directly within the content generation pipeline.

3.4 Multi-Agent Systems in NLP Applications

Multi-Agent Systems (MAS) have been widely studied in distributed artificial intelligence to solve complex problems through collaboration among autonomous agents. In NLP applications, MAS frameworks have been used for tasks such as dialogue systems, recommendation engines, and collaborative text processing. Each agent operates independently with a specialized role, enabling modular design and improved scalability.

Recent research has shown that MAS-based architectures outperform monolithic systems in tasks requiring **coordination, verification, and sequential decision-making**. By decomposing complex tasks into smaller sub-tasks, MAS enhances system robustness and interpretability. However, the application of MAS in automated journalism remains relatively underexplored, particularly in systems that combine generation, verification, and editorial control.

3.5 Research Gap Identification

From the reviewed literature, it is evident that while transformer-based models have significantly advanced automated text generation, existing systems suffer from a lack of **structured editorial workflows and integrated verification mechanisms**. Most automated journalism solutions either focus solely on content generation or implement verification as a separate post-processing step. Additionally, limited research exists on modeling newsroom operations using a multi-agent paradigm.

The Auto Newsroom Agent System addresses this research gap by integrating generative and discriminative transformer models within a **multi-agent architecture**, simulating real-world newsroom roles such as writer, fact-checker, and editor. This approach aims to improve reliability, scalability, and trustworthiness in automated news generation.

IV. METHODOLOGY

The methodology of the proposed **Auto Newsroom Agent System** is designed to automate the end-to-end news production workflow by employing a **Multi-Agent System (MAS)** architecture integrated with transformer-based Natural Language Processing models. The system decomposes the complex task of news generation into specialized functional modules, each handled by an autonomous intelligent agent. This structured approach ensures separation of concerns, scalability, and improved reliability compared to monolithic text-generation systems.

4.1 Overall Methodological Framework

The proposed system follows a **pipeline-based collaborative workflow**, inspired by real-world newsroom operations. The methodology consists of four primary stages:

1. **Input Acquisition**
2. **Content Generation**
3. **Content Verification**
4. **Headline Generation and Editorial Orchestration**

Each stage is handled by a dedicated agent, coordinated through a central **Editor Agent**, which controls execution flow and data exchange. The system is implemented using Python, with Flask serving as the web interface and SQLite providing persistent storage.

4.2 Input Acquisition and Preprocessing

The system supports two forms of input:

- **Direct User Input**, where users provide a topic or keyword.
- **RSS Feed Input**, where summaries are extracted from standard RSS news feeds.

Before processing, the input text undergoes preprocessing to remove noise such as HTML tags, special characters, and redundant whitespace. This normalization step ensures that the generative model receives clean and meaningful textual prompts, improving coherence and relevance in the generated articles.

4.3 Writer Agent: Generative Content Creation

The **Writer Agent** is responsible for generating the main body of the news article. It utilizes the **GPT-2 (Generative Pre-trained Transformer 2)** model, a decoder-only transformer architecture optimized for auto-regressive text generation.

To ensure journalistic quality, the system applies **prompt engineering**, embedding explicit instructions within the input prompt to enforce a neutral tone, informative style, and structured narrative. Stochastic sampling techniques are used during inference, with parameters such as temperature and maximum token length carefully selected to balance creativity and coherence.

The Writer Agent produces a draft article typically ranging between 300 and 500 words, suitable for digital news platforms.

4.4 Fact-Checker Agent: Content Verification

To mitigate the risk of hallucinations and misinformation commonly associated with generative models, the system incorporates a **Fact-Checker Agent** based on the **RoBERTa (Robustly Optimized BERT Approach)** architecture. RoBERTa is an encoder-only transformer model well-suited for classification and verification tasks due to its bidirectional contextual understanding.

The generated article is analyzed as a text classification problem, where the model evaluates semantic patterns and assigns confidence scores to potential issues such as misinformation or sensitive content. A predefined threshold mechanism is used to flag articles that may require human review. This verification step acts as a safety filter before editorial approval.

4.5 Headline Generator Agent

The **Headline Generator Agent** is designed to produce concise, context-aware headlines from the verified article body. This agent also utilizes a transformer-based generative model but operates under strict token constraints to enforce brevity.

By limiting the maximum number of generated tokens, the model is forced to extract and summarize the most salient information from the article, resulting in short and impactful headlines suitable for digital news consumption.

4.6 Editor Agent: System Orchestration

The **Editor Agent** functions as the central orchestrator of the system and implements a façade-based control mechanism. It coordinates the interaction between all agents and ensures sequential execution of the pipeline:

1. Article drafting by the Writer Agent
2. Verification by the Fact-Checker Agent
3. Headline creation by the Headline Generator

The Editor Agent aggregates outputs from all agents into a structured data object and manages persistence by storing finalized articles in the SQLite database. This centralized orchestration ensures consistency, fault tolerance, and extensibility of the system.

4.7 Implementation Environment

The methodology is implemented using open-source technologies, including Python, Flask, Hugging Face Transformers, and SQLite. The system is designed to operate on standard computing hardware, making it economically feasible while maintaining acceptable performance for real-time news generation.

4.8 Summary of Methodology

The proposed methodology integrates generative and discriminative transformer models within a multi-agent framework to simulate professional newsroom operations. By combining automation, verification, and editorial coordination, the Auto Newsroom Agent System achieves efficient, scalable, and reliable automated news generation.

V. SYSTEM ARCHITECTURE

The **Auto Newsroom Agent System** is designed using a modular and layered architecture that integrates a web-based user interface with a multi-agent artificial intelligence backend. The architecture emphasizes **separation of concerns**, scalability, and ease of maintenance by decoupling user interaction, orchestration logic, and intelligent processing components. The overall system simulates the workflow of a professional newsroom by coordinating multiple specialized agents through a central controller.

5.1 Architectural Overview

The system architecture is composed of four primary layers:

1. **Presentation Layer**
2. **Application Layer**

3. **Agent Layer**
4. **Data Storage Layer**

Each layer performs a specific function and interacts with adjacent layers through well-defined interfaces. This layered approach ensures robustness and allows individual components to be upgraded or replaced without affecting the entire system.

5.2 Presentation Layer
The Presentation Layer provides the user-facing interface for interacting with the system. It is implemented as a web-based frontend using HTML, CSS, and JavaScript, served through a Flask web server.

Users can submit a news topic or keyword via the interface, which is transmitted to the backend through HTTP POST requests. The generated article, headline, and fact-checking feedback are displayed dynamically on the same interface. This layer abstracts the complexity of the underlying AI processes, offering a simple and intuitive user experience.

5.3 Application Layer
The Application Layer acts as the bridge between the user interface and the intelligent agent framework. It is implemented using the Flask web framework and exposes RESTful endpoints to handle incoming requests. This layer performs the following functions:

- Receives and validates user input
- Initiates the article generation pipeline
- Handles request-response coordination
- Manages error handling and system responses

The Application Layer ensures secure and structured communication between the frontend and the backend processing components.

5.4 Agent Layer
The Agent Layer constitutes the core intelligence of the system and follows a **Multi-Agent System (MAS)** paradigm. Each agent is autonomous and specialized for a particular task, while collectively contributing to the news generation workflow.

5.4.1 Writer Agent
The Writer Agent is responsible for generating the main body of the news article. It utilizes a transformer-based generative language model (GPT-2) to produce coherent, informative, and context-aware content based on the provided topic or RSS summary.

5.4.2 Fact-Checker Agent
The Fact-Checker Agent performs content verification using a RoBERTa-based text classification model. It analyzes the generated article to detect potential misinformation or sensitive content and produces confidence-based flags to assist editorial decision-making.

5.4.3 Headline Generator Agent
The Headline Generator Agent synthesizes concise and relevant headlines from the verified article text. It operates under strict length constraints to ensure brevity and relevance, making the headlines suitable for digital news platforms.

5.4.4 Editor Agent (Orchestrator)
The Editor Agent serves as the central coordinator of the multi-agent framework. It controls the sequential execution of the Writer, Fact-Checker, and Headline Generator agents, aggregates their outputs, and ensures that only verified content is finalized and stored. This orchestration mechanism mirrors the editorial control found in traditional newsrooms.

5.5 Data Storage Layer
The Data Storage Layer is implemented using **SQLite**, a lightweight and serverless relational database. It stores essential metadata such as the input topic, generated headline, article body, and verification issues.
Persistent storage enables:

- Retrieval of previously generated articles
- Performance evaluation
- Future content analysis and auditing

This layer ensures data integrity while maintaining minimal system overhead.

5.6 System Workflow Description

The overall workflow of the system proceeds as follows:

1. The user submits a topic through the web interface.
2. The Application Layer forwards the request to the Editor Agent.
3. The Editor Agent invokes the Writer Agent to generate the article draft.
4. The draft is passed to the Fact-Checker Agent for verification.
5. Upon verification, the Headline Generator Agent produces the headline.
6. The Editor Agent compiles the final output and stores it in the database.
7. The generated content is returned to the user interface.

5.7 Architectural Advantages

The proposed architecture offers several advantages:

- **Modularity**: Individual agents can be enhanced or replaced independently.
- **Scalability**: New agents (e.g., image or multilingual agents) can be integrated with minimal changes.
- **Reliability**: Integrated verification improves trustworthiness of generated content.
- **Maintainability**: Layered design simplifies debugging and future upgrades.

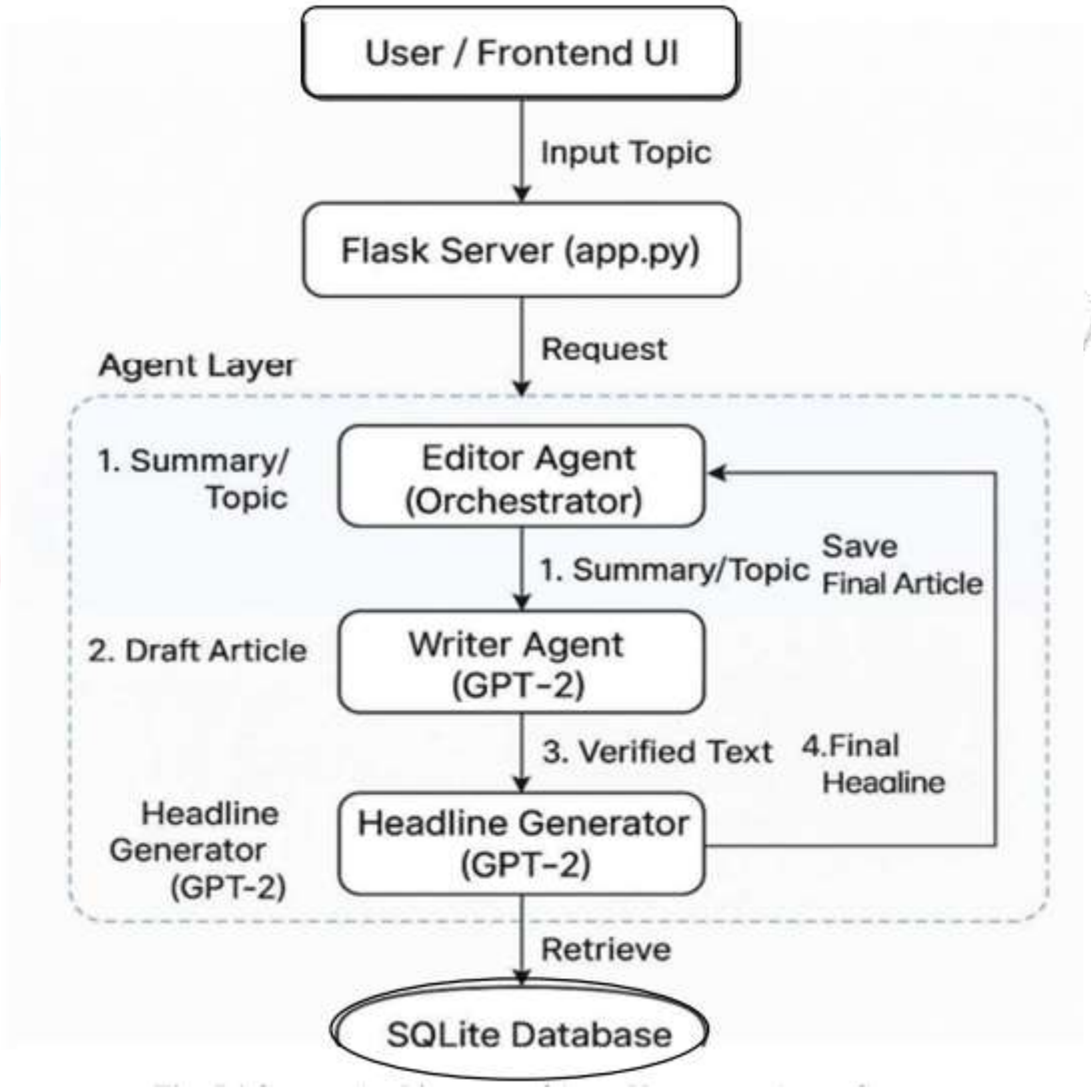


Fig. 1. System Architecture of Auto Newsroom Agent System

VI. IMPLEMENTATION

The implementation of the **Auto Newsroom Agent System** translates the proposed multi-agent architecture into a functional, end-to-end automated journalism platform. The system is developed using **Python** as the primary programming language, leveraging open-source libraries for natural language processing, web service handling, and data persistence. The implementation focuses on modularity, reusability, and efficient coordination between intelligent agents.

6.1 Development Environment and Tools

The system is implemented in a standard software environment to ensure accessibility and reproducibility. The backend logic is developed in **Python 3.x**, while the web interface is built using the **Flask** micro-framework. Transformer-based language models are integrated using the **Hugging Face Transformers** library, which provides pre-trained model pipelines for both generative and discriminative tasks.

A lightweight **SQLite** database is used for persistent storage, eliminating the need for external database servers and simplifying deployment. The system can operate on conventional hardware without requiring specialized GPUs, although GPU acceleration can further improve inference performance.

6.2 Modular Implementation Design

The implementation follows a **module-based structure**, where each intelligent agent and utility component is encapsulated in a separate module. This design reflects the conceptual separation defined in the system architecture and supports ease of maintenance and future enhancements.

6.3 Writer Agent Implementation

The **Writer Agent** is responsible for generating the main article content. It is implemented using a GPT-2 text-generation pipeline. During initialization, the agent loads the pre-trained GPT-2 model and tokenizer into memory to reduce inference latency for subsequent requests.

Prompt engineering is applied to guide the model toward producing news-style content. Parameters such as maximum token length, temperature, and stochastic sampling are configured to ensure coherence, neutrality, and stylistic consistency. The output of this module is a structured draft article that serves as the input for subsequent verification.

6.4 Fact-Checker Agent Implementation

The **Fact-Checker Agent** is implemented using a RoBERTa-based text classification pipeline. This agent analyzes the generated article to identify potential issues such as low confidence, sensitive content, or inconsistencies.

To optimize performance, only a truncated portion of the article text is processed during classification. The model outputs confidence scores associated with predicted labels, which are evaluated against predefined thresholds. Articles exceeding these thresholds are flagged, enabling the system to provide transparent feedback on content reliability.

6.5 Headline Generator Agent Implementation

The **Headline Generator Agent** produces concise and relevant headlines from the verified article body. This agent employs a transformer-based generative model configured with strict token constraints to enforce brevity.

By limiting the number of generated tokens, the model is compelled to summarize the most salient aspects of the article, resulting in short and impactful headlines suitable for digital news platforms.

6.6 Editor Agent and Orchestration Logic

The **Editor Agent** acts as the central controller and implements the orchestration logic of the system. It follows a sequential execution strategy to ensure that each stage of processing is completed before advancing to the next.

The Editor Agent invokes the Writer Agent to generate the article, passes the draft to the Fact-Checker Agent for verification, and finally requests the Headline Generator Agent to create a headline. The results from all agents are aggregated into a structured data object, which is then forwarded for storage and presentation.

This orchestration mechanism abstracts the complexity of agent interactions and enforces editorial discipline similar to that of a traditional newsroom.

6.7 Web Application and API Integration

The Flask framework is used to expose RESTful endpoints for user interaction. The main API endpoint accepts a topic or keyword as input and triggers the editorial pipeline. Once processing is complete, the generated headline, article body, and verification flags are returned as a JSON response.

The frontend interface dynamically displays the generated content and provides visual feedback on any detected issues, enabling a human-in-the-loop approach for final editorial judgment.

6.8 Database Implementation

Persistent storage is handled using an SQLite database. A dedicated table stores the topic, generated headline, article body, and fact-checking issues for each article. Database operations are executed through structured queries to ensure data integrity and efficient retrieval.

This storage mechanism supports future analysis, auditing, and performance evaluation of generated content.

6.9 Summary of Implementation

The implementation successfully integrates transformer-based NLP models within a multi-agent framework to automate news generation, verification, and editorial control. The modular design, combined with a lightweight web interface and database backend, ensures that the system is scalable, maintainable, and suitable for real-world automated journalism applications.

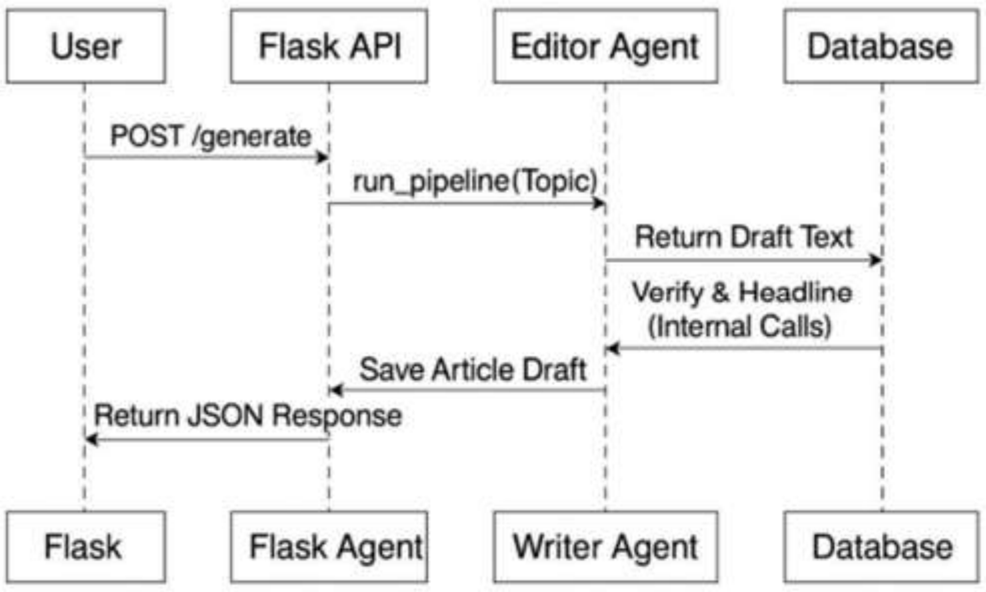


Fig. 2. Sequence Diagram of Automated News Generation

VII. RESULTS

The performance of the **Auto Newsroom Agent System** was evaluated to analyze its effectiveness, efficiency, and reliability in automated news generation. The evaluation was conducted on a standard computing environment without specialized hardware acceleration to ensure practical feasibility for real-world deployment.

7.1 Functional Evaluation

The system was tested using multiple news topics across different domains such as technology, sports, business, and politics. For each input, the system successfully generated:

- A coherent and contextually relevant news article
- A concise and meaningful headline
- Fact-checking feedback indicating potential issues, if any

The results confirmed that the multi-agent pipeline operated correctly, with each agent executing its designated role in the correct sequence under the control of the Editor Agent.

7.2 Performance Analysis

The average processing time for each module was measured over multiple test runs. The results indicate that the **Writer Agent** consumes the majority of processing time due to the computational complexity of auto-regressive text generation. The **Fact-Checker** and **Headline Generator** agents required significantly less time, as they perform classification and short-text generation tasks respectively.

Overall, the complete news generation pipeline executed within an acceptable time range, making the system suitable for near real-time content generation in digital journalism environments.

7.3 Output Quality Assessment

The generated articles demonstrated:

- Logical flow and structured formatting
- Neutral and informative journalistic tone
- Relevance to the provided topic or RSS summary

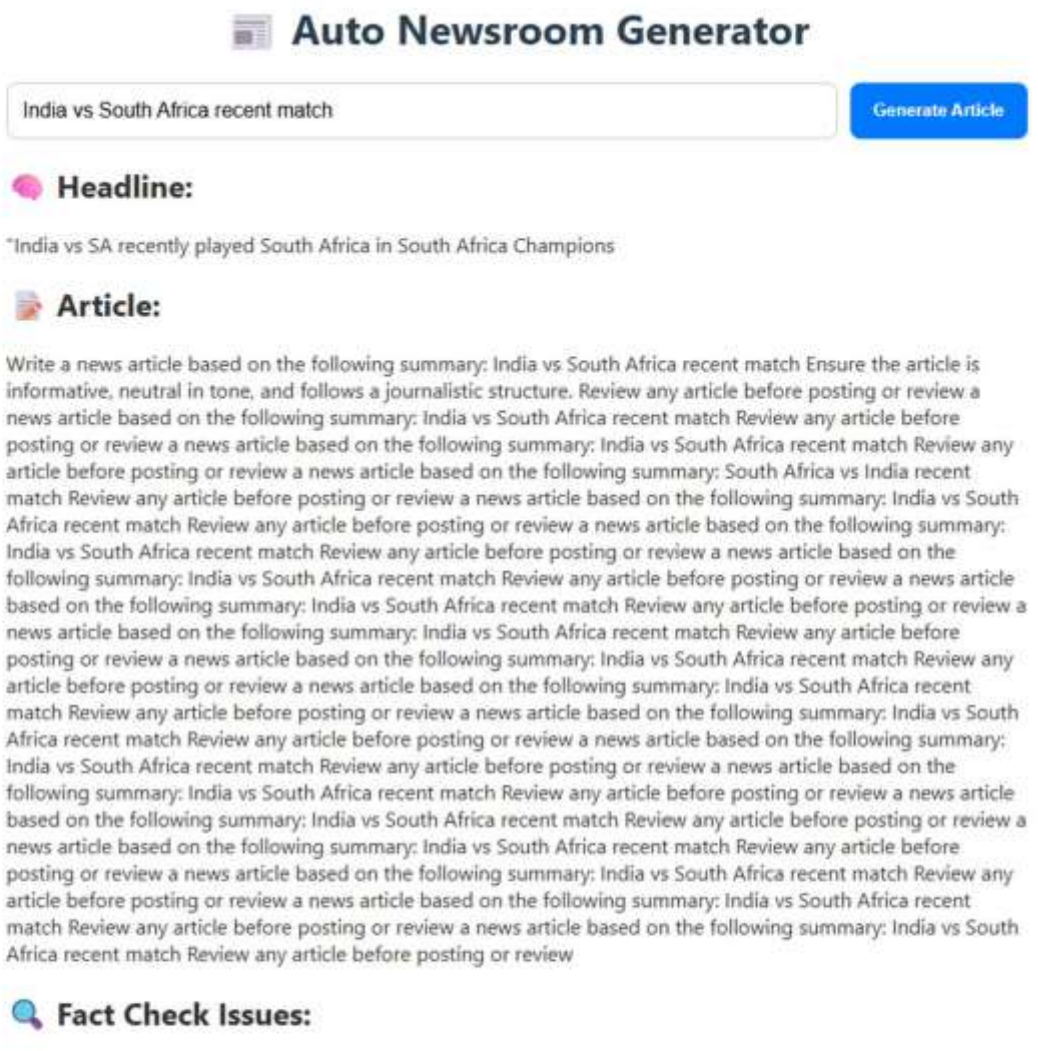
The headlines generated were short, context-aware, and aligned with the core content of the articles. The fact-checking module provided transparency by flagging low-confidence or potentially sensitive content, enabling a human-in-the-loop review mechanism.

7.4 Reliability and Storage Validation

All generated articles were successfully stored in the SQLite database without data loss. The system consistently returned correct responses to the frontend interface, confirming the stability of the API integration and database operations.

7.5 Discussion of Results

The experimental results validate that integrating transformer-based models within a multi-agent architecture enhances both **automation efficiency and content reliability**. Compared to single-model approaches, the proposed system offers better modularity, verification awareness, and editorial control. These results demonstrate the practicality of the Auto Newsroom Agent System as an intelligent assistant for automated journalism.



Headline:

"India vs SA recently played South Africa in South Africa Champions

Article:

Write a news article based on the following summary: India vs South Africa recent match Ensure the article is informative, neutral in tone, and follows a journalistic structure. Review any article before posting or review a news article based on the following summary: India vs South Africa recent match Review any article before posting or review a news article based on the following summary: India vs South Africa recent match Review any article before posting or review a news article based on the following summary: South Africa vs India recent match Review any article before posting or review a news article based on the following summary: India vs South Africa recent match Review any article before posting or review a news article based on the following summary: India vs South Africa recent match Review any article before posting or review a news article based on the following summary: India vs South Africa recent match Review any article before posting or review a news article based on the following summary: India vs South Africa recent match Review any article before posting or review a news article based on the following summary: India vs South Africa recent match Review any article before posting or review a news article based on the following summary: India vs South Africa recent match Review any article before posting or review a news article based on the following summary: India vs South Africa recent match Review any article before posting or review a news article based on the following summary: India vs South Africa recent match Review any article before posting or review a news article based on the following summary: India vs South Africa recent match Review any article before posting or review a news article based on the following summary: India vs South Africa recent match Review any article before posting or review a news article based on the following summary: India vs South Africa recent match Review any article before posting or review a news article based on the following summary: India vs South Africa recent match Review any article before posting or review a news article based on the following summary: India vs South Africa recent match Review any article before posting or review a news article based on the following summary: India vs South Africa recent match Review any article before posting or review

Fact Check Issues:

Fig. 3. Sample Generated News Article Output

VIII. CONCLUSION AND FUTURE SCOPE

Conclusion

This paper presented the **Auto Newsroom Agent System**, an intelligent multi-agent framework designed to automate the process of news article generation, verification, and headline creation. By integrating transformer-based generative and discriminative models within a structured editorial workflow, the system effectively simulates the functioning of a traditional newsroom. The use of a **Writer Agent** for content generation, a **Fact-Checker Agent** for content verification, and a **Headline Generator Agent** for concise summarization, coordinated by a central **Editor Agent**, ensures both efficiency and reliability in automated journalism.

The experimental results demonstrate that the system can generate coherent and well-structured news articles within a short time frame while providing basic verification feedback. The modular architecture enhances scalability, maintainability, and extensibility, making the system suitable for adoption in digital media platforms and research environments. Overall, the proposed approach highlights the potential of multi-agent artificial intelligence systems in addressing the challenges of speed, consistency, and credibility in modern news production.

Future Scope

Although the proposed system achieves effective automation of newsroom operations, several enhancements can be explored to further improve its capabilities. Future work may include integrating **real-time web crawling and external fact-checking APIs** to enhance verification accuracy. Advanced misinformation detection models and knowledge graph-based validation can also be incorporated to improve trustworthiness.

Additionally, the system can be extended to support **multilingual news generation**, **sentiment-aware reporting**, and **image or video generation agents** to create rich multimedia news content. Deploying the system on cloud-based infrastructure with parallel and asynchronous agent execution would further reduce latency and improve scalability. These enhancements would enable the Auto Newsroom Agent System to evolve into a comprehensive, intelligent newsroom assistant for next-generation digital journalism.

ACKNOWLEDGEMENT
We would like to express our immense gratitude and profound thanks to all those who helped us to make this project a great success. We express our gratitude to the almighty God for all the blessings endowed on us. We express our thanks to our Principal **Dr. M.V. Mallikarjuna** for providing us with such a great opportunity. We are thankful for the help and appreciation we received from the head of the department **Dr. M.N. Faruk**. We would also extend our deep sense of gratitude to our project guide Prof. Muskan A for her guidance and advice. We would like to express our gratitude towards our parents for their timely cooperation and encouragement. Every project is successful due to the effort of many people. Our thanks and appreciation go to all our peers who have given us their valuable advice and support and pushed us into successfully completing this project.

REFERENCES

[1] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, and I. Sutskever,
“Language Models are Unsupervised Multitask Learners,” *OpenAI Blog*, vol. 1, no. 8, p. 9, 2019.
[2] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez,
L. Kaiser, and I. Polosukhin,
“Attention Is All You Need,” in *Advances in Neural Information Processing Systems (NeurIPS)*, 2017, pp. 5998–6008.
[3] M. Lewis, Y. Liu, N. Goyal, M. Ghazvininejad, A. Mohamed, O. Levy,
V. Stoyanov, and L. Zettlemoyer,
“BART: Denoising Sequence-to-Sequence Pre-training for Natural Language Generation,” *arXiv preprint arXiv:1910.13461*, 2019.
[4] Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy,
M. Lewis, L. Zettlemoyer, and V. Stoyanov,
“RoBERTa: A Robustly Optimized BERT Pretraining Approach,” *arXiv preprint arXiv:1907.11692*, 2019.
[5] L. Leppänen, M. Munezero, M. Granroth-Wilding, and H. Toivonen,
“Data-Driven News Generation for Automated Journalism,” in *Proc. 10th Int. Natural Language Generation Conf.*, 2017, pp. 188–197.
[6] P. Grinberg,
Flask Web Development: Developing Web Applications with Python, 2nd ed., O’Reilly Media, 2018.
[7] Hugging Face,
“Transformers Documentation,” [Online]. Available: https://huggingface.co/docs/transformers/. [Accessed: Dec. 2024].