A black background with white text

AI-generated content may be incorrect.

“Today in News”

Daily Trend Detection Using Machine Learning

**Candidate:** Blerta Zenuni **Mentor:** Asst. Prof. Dr. Nuhi Besimi

Tetovo, 2025

# 

# STATEMENT OF ORIGINALITY

I hereby declare that this capstone project, titled **"Today in News"**, is the result of my own work and has not been copied or plagiarized from any source. All sources of information and ideas taken from the work of others have been acknowledged and referenced accordingly.

This project has been carried out solely for academic purposes as a requirement for completing my studies and has not been submitted elsewhere for any other degree or qualification.

I understand the consequences of committing academic dishonesty, and I affirm that this work meets the standards of originality and integrity expected by my institution.

Name\_\_\_\_\_\_\_\_\_\_\_\_\_ Signature\_\_\_\_\_\_\_\_\_\_\_\_\_ Date\_\_\_\_\_\_\_\_\_\_\_\_\_

# Abstract

This capstone project presents the development of an automated system titled **"Today in News”** designed to fetch daily news articles from multiple online sources, categorize them, identify trending topics per category, summarize key articles using AI, and store the results in a PostgreSQL database for user-friendly viewing through a Streamlit-based web interface.

The project is structured into five main phases: (1) collecting news articles using NewsAPI, Guardian API, and the New York Times API, (2) preprocessing and categorizing articles to detect trending keywords using TF-IDF, (3) generating human-readable summaries of the most relevant articles using OpenAI's GPT model, (4) storing summarized trends in a relational PostgreSQL database, and (5) building a clean and interactive UI for users to explore daily trends by category and date.

This system combines data engineering, natural language processing, and web development into a cohesive workflow, providing a real-time insight platform for trending news analysis. The resulting application is modular, scalable, and adaptable for future integration with more data sources or advanced summarization models. The project demonstrates practical application of AI in journalism and supports informed daily decision-making for general users.

# Acknowledgment

I would like to express my sincere gratitude to my academic advisor and supervising professor for their continuous support, guidance, and constructive feedback throughout the development of this capstone project. Their insights and encouragement played a significant role in shaping the project from its initial idea to its final implementation.

I am also thankful to the Faculty of Contemporary Sciences and Technologies at South East European University for providing a learning environment that fosters innovation and hands-on experience. Special thanks to the course coordinators and assistants whose support helped me navigate the technical and academic challenges of this work.

Finally, I would like to thank my family and friends for their patience, encouragement, and belief in my work throughout the duration of this project.

# Table of Contents

Contents

[STATEMENT OF ORIGINALITY 2](#_Toc202210889)

[Abstract 3](#_Toc202210890)

[Acknowledgment 4](#_Toc202210891)

[Table of Contents 5](#_Toc202210892)

[1. Introduction and Motivation 6](#_Toc202210893)

[1.1 Introduction 6](#_Toc202210894)

[1.2 Motivation 6](#_Toc202210895)

[2. Project Overview 9](#_Toc202210896)

[2.1 Objective 9](#_Toc202210897)

[2.2 Importance of the Project 10](#_Toc202210898)

[2.3 Technological Stack 11](#_Toc202210899)

[2.4 Design Philosophy 13](#_Toc202210900)

[2.5 Real-World Use Case 15](#_Toc202210901)

[3. Technical Implementation 17](#_Toc202210902)

[3.1 System Architecture Overview 17](#_Toc202210903)

[3.2 System Architecture Diagram 17](#_Toc202210904)

[3.3 Phase 1: News Collection Script 18](#_Toc202210905)

[3.4 Phase 2: Preprocessing & Trend Detection 19](#_Toc202210906)

[3.5 Phase 3: Summarization 20](#_Toc202210907)

[3.6 Phase 4: Database Integration 20](#_Toc202210908)

[3.7 Phase 5: User Interface (Streamlit) 21](#_Toc202210909)

[3.8 Folder Structure 23](#_Toc202210910)

[4. Related Work 24](#_Toc202210911)

[4.1 Industry Tools Comparison 24](#_Toc202210912)

[4.2 Academic Approaches 24](#_Toc202210913)

[5. Conclusion of Related Work 26](#_Toc202210914)

[6. Conclusion 27](#_Toc202210915)

[7. Future Developments 28](#_Toc202210916)

[8. References & Bibliography 30](#_Toc202210917)

# Introduction and Motivation

## 1.1 Introduction

In today’s fast-paced digital world, individuals are constantly inundated with an overwhelming abundance of online news content arriving from a multitude of sources. Every minute, countless articles are published on topics ranging from local events to international crises, creating a digital landscape where important updates are easily buried under a sea of less relevant or repetitive stories. As a result, readers face significant challenges in quickly identifying which news items are truly noteworthy, which trends are emerging, and how these developments might impact their personal or professional lives.

Traditional news platforms, while rich in content, often fail to provide an efficient way for users to filter, prioritize, and comprehend key stories across different areas of interest. Moreover, with the rapid evolution of information, what is relevant today might be outdated by tomorrow, further complicating the task of staying informed. The necessity for tools that can intelligently process vast amounts of news data in real time has never been greater.

This capstone project, titled “Today In News”, directly addresses this modern challenge by automating the end-to-end process of news analysis and trend detection. The system seamlessly integrates multiple technologies to fetch daily news articles from reliable sources through APIs, preprocess and clean the data, identify trending topics within pre-defined categories such as business, sports, technology, entertainment, and health, generate concise and human-readable summaries of these stories using advanced language models, and store the summarized information in a structured PostgreSQL database. The final output is presented through an intuitive, interactive web interface built with Streamlit, allowing users to effortlessly navigate through trending topics, read summaries, and access original articles.

By transforming complex news data into organized, easy-to-digest summaries, this system empowers individuals to stay updated on the most important stories without spending excessive time sifting through endless articles. It bridges the gap between the information explosion of the digital era and the practical needs of modern readers seeking clarity and relevance.

## 1.2 Motivation

The motivation behind this project stems from several interrelated challenges faced by news consumers in the digital age:

**1. Information Overload:**

Every day, readers are confronted with thousands of news articles published across numerous websites, blogs, and news platforms. This overwhelming flow of information makes it difficult to identify, at a glance, the most significant or impactful news stories. As a consequence, important updates might go unnoticed, while trivial stories can consume valuable time and attention. The inability to efficiently separate relevant news from irrelevant noise leads to frustration, fatigue, and, ultimately, disengagement from current events.

**2. Category-Based Relevance:**

Many individuals have specific areas of interest they wish to follow closely, such as business, sports, health, technology, or entertainment. However, mainstream news platforms typically present articles in a general feed or broad headlines, without personalized structuring that highlights trending topics within each category. This forces users to manually search through multiple websites or sections to find updates that matter to them, wasting time and diminishing the overall user experience.

**3. Lack of Contextual Summarization:**

Even when readers identify articles of interest, they are often faced with lengthy texts that require significant time to read in full. In our increasingly busy lives, many people lack the time or patience to read multiple long-form articles to get a sense of key developments. There is a growing need for clear, concise summaries that convey essential information in a matter of seconds.

**4. The Need for Historical Insights:**

While staying updated on current trends is important, understanding how topics evolve over time is equally valuable for researchers, journalists, analysts, and curious readers. Most existing news platforms do not offer tools that allow users to easily track trends by date, limiting opportunities for in-depth analysis and pattern recognition.

By building this system, the aim was to provide a smart, automated, and user-friendly tool that tackles these challenges head-on. The system not only fetches and organizes news articles daily but also applies advanced natural language processing techniques to detect trending topics, generates clear and concise summaries, and stores historical data for later analysis. This approach transforms the news-reading experience from a passive, overwhelming process into an active, personalized, and time-efficient activity.

The motivation extends beyond individual convenience: it reflects a broader vision of enhancing media literacy and informed decision-making. By providing transparent access to trends across categories, users can gain a balanced view of daily events, reduce susceptibility to sensationalism, and make better-informed decisions in their personal and professional lives.

Ultimately, *“Today In News”* aspires to redefine how people interact with news in the digital age—transforming the chaotic influx of online articles into an organized, relevant, and engaging experience that keeps users connected to what truly matters.

# Project Overview

## 2.1 Objective

The primary objective of the “Today In News” capstone project is to develop a robust, fully automated system that seamlessly collects, processes, analyzes, summarizes, and displays daily trending news articles categorized into key topics such as business, sports, technology, entertainment, and health. This objective is driven by the need to address one of the most pressing challenges faced by modern news consumers: the overwhelming volume of information available online, which often makes it difficult to discern what is truly relevant and significant.

In the contemporary digital landscape, hundreds—if not thousands—of news articles are published every hour across countless online platforms, from mainstream media websites to independent blogs. Readers, whether they are professionals, students, or general audiences, are often left sifting through an unmanageable sea of headlines, duplicate stories, and sensationalized content to find accurate and important news. This not only consumes valuable time but can also result in missing critical updates or failing to see emerging trends in areas of personal or professional interest.

Our goal with this project is to streamline and simplify the news consumption process by creating a unified platform that reliably delivers only the most relevant and concise summaries of trending articles within each predefined category. This ensures that users receive focused, meaningful insights without having to navigate multiple sources or read lengthy, repetitive articles.

The system is designed as a modular and scalable pipeline, where each phase contributes to transforming raw, unstructured data into actionable knowledge:

1. **News Collection**: The pipeline initiates by fetching the most recent news articles from trusted APIs, specifically NewsAPI, The Guardian API, and The New York Times API. These sources were chosen to ensure reliability, diversity of viewpoints, and broad coverage across categories.
2. **Preprocessing and Cleaning**: Raw articles undergo thorough text preprocessing, including normalization, tokenization, stopword removal, and lemmatization. This prepares the data for accurate analysis by standardizing language and reducing noise.
3. **Trend Detection**: The system employs Term Frequency-Inverse Document Frequency (TF-IDF) vectorization to identify the most significant and frequently discussed keywords within each category. This statistical technique highlights words that best represent the unique focus of articles for the day, effectively revealing what is trending.
4. **Summarization**: Using OpenAI’s advanced GPT language models, the system generates clear and concise 2–3 sentence summaries of the top articles associated with each detected trend. This step ensures that even complex or lengthy articles are distilled into easily digestible insights.
5. **Database Storage**: Summarized articles, along with key metadata (e.g., title, publication date, URL), are stored in a PostgreSQL relational database. This allows efficient querying by category and date, and supports long-term storage for historical trend analysis.
6. **User Interface Presentation**: The final output of the pipeline is presented through an interactive and intuitive web application built with Streamlit. The UI is designed with user experience in mind, providing dynamic tabs for each category, a date selector for exploring past trends, and a clean layout that makes it easy to read summaries and access original articles.

Through this systematic, end-to-end approach, the “Today In News” system aims to empower users to stay informed about key developments across various domains with minimal effort, reducing information overload while enhancing the speed, relevance, and clarity of news consumption.

## 2.2 Importance of the Project

The “Today In News” capstone project holds significant importance as it directly addresses several pressing real-world needs in today’s fast-paced, information-driven society. First and foremost, it provides a powerful solution for **media summarization**, catering to individuals who lack the time or capacity to read multiple full-length articles every day. By generating concise, human-readable summaries of the most important news stories in each category, the system empowers busy professionals, students, and casual readers alike to stay informed with minimal effort.

Another key aspect of its importance lies in its ability to offer **categorical filtering**, which enables users to focus only on topics relevant to their interests, such as technology, sports, health, business, or entertainment. Unlike traditional news platforms that often present a generic feed of headlines, “Today In News” allows readers to dive directly into the categories they care about, making their news consumption experience more personalized, efficient, and engaging.

Equally crucial is the system’s support for **historical tracking of trends by date**, which offers immense value to journalists, media analysts, researchers, and academics. By storing summarized trend data in a structured PostgreSQL database, the system enables retrospective exploration of how topics and discussions evolve over days, weeks, or even months. This functionality can aid in research on news cycles, public interest patterns, or the progression of specific stories over time, providing a unique analytical perspective not readily available on mainstream news platforms.

Furthermore, the project demonstrates the **academic integration of cutting-edge technologies**, uniting natural language processing (NLP), API consumption, database management, and modern web frameworks into a cohesive and practical system. This integration provides a hands-on example of how different computer science concepts can be applied synergistically to solve complex, real-world problems. For students, educators, and developers, “Today In News” serves as a model project that showcases best practices in modular design, scalable architecture, and the application of AI for societal benefit.

Beyond academic and individual use cases, the potential for **real-world deployment** further underscores the importance of the project. The system could be implemented in universities to help students and faculty stay updated on relevant news, or in newsrooms to support journalists with rapid trend analysis and article summarization. Additionally, with further development, the project could evolve into a full-scale commercial news aggregation platform, offering organizations and media companies a powerful tool to deliver curated, summarized, and trend-focused news experiences to their audiences.

By addressing these needs, “Today In News” demonstrates not only technical innovation but also a meaningful commitment to enhancing how society engages with information in an increasingly complex media landscape.

## 2.3 Technological Stack

**Technological Stack**

To bring the *“Today In News”* system to life, a thoughtfully selected, modern technological stack was assembled, covering every layer of the application—from the backend logic that powers the core functionality to the frontend interface that presents results to users in an intuitive way. Each tool and technology was carefully chosen with a focus on performance, ease of use, integration capabilities, and suitability for the project’s unique requirements. Together, these technologies form a cohesive and scalable ecosystem capable of efficiently processing and delivering daily news insights.

At the heart of the system is the **Python programming language**, selected for its extensive ecosystem and unparalleled support for natural language processing (NLP), machine learning, data manipulation, and web development tasks. Python’s readability and community support also made it an ideal choice for rapid development and future maintenance.

For **news fetching**, the project relies on three major APIs: NewsAPI, The Guardian API, and The New York Times API. These trusted sources were chosen to ensure diversity and reliability of news content, providing real-time access to fresh articles across a wide range of topics and categories. These APIs serve as the primary data input layer of the pipeline.

The **text preprocessing** stage is implemented using the Natural Language Toolkit (**NLTK**), a leading Python library for text analysis and processing. NLTK provides robust tools for tokenization, stopword removal, and lemmatization, enabling the system to clean and normalize raw article text effectively, preparing it for accurate trend detection.

For **trend detection**, the project employs **TF-IDF vectorization via Scikit-learn**, a proven technique for identifying the most important keywords within a collection of documents. TF-IDF scores are calculated across articles in each category to highlight significant terms, which are then used to detect emerging or prominent topics for the day.

In the **summarization** phase, the system leverages the power of **OpenAI’s GPT-3.5 or GPT-4 language models**, which generate concise and coherent 2–3 sentence summaries of selected trending articles. These AI-generated summaries provide users with clear, human-readable insights without the need to read full articles.

For **data storage**, a **PostgreSQL relational database** is utilized. PostgreSQL was chosen for its robustness, scalability, and powerful querying capabilities, allowing the system to store, retrieve, and manage summarized trend data efficiently. Its support for advanced indexing and data integrity makes it well-suited for maintaining historical records of daily news trends.

To deliver an engaging and user-friendly experience, the **Streamlit web framework** is used to build the **web interface**. Streamlit enables rapid development of interactive dashboards and applications, allowing the project to present categorized news summaries with a modern, responsive design. Features like date selectors and category tabs enhance the user experience by providing easy navigation and exploration of daily and historical trends.

For **environment management**, the **dotenv library** is used to securely store and manage sensitive credentials, such as API keys and database connection strings. By keeping this information in environment variables, the project maintains good security practices and simplifies configuration across different environments.

Finally, for **visualization and flow mapping**, tools like **Adobe Illustrator** and other diagramming software were employed. These were essential for creating clear architectural diagrams, workflow charts, and other visual aids used in both project documentation and presentations, ensuring complex system designs are easily understood.

Together, these technologies form a solid foundation that makes *“Today In News”* a reliable, maintainable, and scalable solution for daily news trend detection.

**Technology Stack Table**

|  |  |  |
| --- | --- | --- |
| Component | Technology | Description |
| Programming Language | Python | Chosen for its extensive ecosystem, particularly in NLP and web development |
| News Fetching | NewsAPI, Guardian API, NYT API | These APIs allow real-time access to news articles across multiple categories |
| Text Preprocessing | NLTK (Natural Language Toolkit) | Used to clean, tokenize, and normalize raw article text |
| Trend Detection | TF-IDF via Scikit-learn | Identifies the most important keywords in articles per category |
| Summarization | OpenAI GPT-3.5 / GPT-4 | Generates concise and readable summaries of trending articles |
| Database | PostgreSQL | A robust relational database system used to store categorized trend data |
| Web Interface | Streamlit | Framework used to build the interactive user interface of the application |
| Environment Management | dotenv | Stores sensitive API keys securely in environment variables |
| Visualization & Flow Mapping | Adobe Illustrator / Diagrams | Used for project visualization and system architecture presentation |

## 2.4 Design Philosophy

The design philosophy behind the *“Today In News”* system is rooted in principles of modularity, maintainability, scalability, and user-centricity. These guiding concepts were applied throughout every phase of the project to ensure that the system is not only technically robust but also easy to understand, modify, and expand upon in the future.

At its core, the system embraces a **modular architecture**, where each stage of the news processing pipeline is implemented as an independent module with clearly defined inputs and outputs. This modularity brings several critical advantages: it improves maintainability, as modifications or bug fixes in one module (e.g., preprocessing) do not inadvertently affect the functionality of others; it promotes reusability, as components like the preprocessing scripts or database utilities can be repurposed in future projects or adapted to handle different types of text data; and it supports expandability, allowing new features—such as additional news APIs, extra language support, or advanced trend detection techniques—to be integrated seamlessly without overhauling the entire codebase.

Another central tenet of the design philosophy is **data-driven processing**, which avoids hardcoding summaries or keywords. Instead, the system dynamically updates its outputs based on fresh data collected from news APIs each day. This ensures that every user interaction reflects the latest, most relevant information, creating a highly responsive and adaptive user experience. By relying on real-time data and automated processing, the system guarantees that the news insights presented are always timely, accurate, and reflective of current events.

The design also places strong emphasis on **transparency and interpretability**. While many modern AI systems function as “black boxes,” *“Today In News”* was intentionally crafted to provide clear, understandable outputs at every stage. For instance, detected keywords from the TF-IDF analysis can be printed or logged for debugging, offering insights into why certain articles were selected as trending. Summaries generated by OpenAI’s GPT models are concise and human-readable, enhancing trust in the system’s outputs. The clean, tabbed user interface allows users to navigate categories effortlessly and examine daily or historical trends with minimal effort.

Moreover, the **separation of concerns** is a key design strategy reflected in the project’s folder and code structure. For example, the database logic resides entirely in the write\_read\_database.py script, keeping data storage tasks isolated from UI logic in app.py. This clear division of responsibilities simplifies debugging and future enhancements while following software engineering best practices.

Scalability was another major consideration in the design philosophy. By structuring the system as a sequence of independent, linear phases—fetching, preprocessing, trend detection, summarization, storage, and presentation—the project can handle increasing data volumes or more complex analysis with minimal changes to the architecture. Each phase can be optimized, parallelized, or migrated to cloud services in the future to accommodate growing user bases or expanded feature sets.

Finally, **user experience** was prioritized throughout the design process. The Streamlit-based web interface was intentionally developed to be intuitive, responsive, and visually clear, ensuring that even non-technical users can comfortably explore trending news summaries. Elements like real-time status updates, date selectors, and category tabs were included to enhance usability and engagement.

Collectively, these design principles ensure that *“Today In News”* is not only a technically sound project but also a sustainable and user-friendly solution capable of evolving alongside the changing needs of its users and technological advancements in news aggregation and natural language processing.

## 2.5 Real-World Use Case

To illustrate the practical value and user-centric benefits of the *“Today In News”* system, consider a typical real-world scenario involving a professional who needs to stay updated on daily events across multiple domains, such as a journalist, business analyst, researcher, or even a casual news enthusiast.

Imagine a user launching the *Today In News* web application on their laptop or mobile device early in the morning. As soon as the application loads, the system’s automated pipeline begins working behind the scenes: it connects to the NewsAPI, The Guardian API, and The New York Times API, retrieving the latest articles published overnight across categories like business, technology, sports, health, and entertainment. This collection process ensures the user receives up-to-date coverage from reliable and diverse sources.

Once fetched, the articles are quickly cleaned and preprocessed. Text normalization and tokenization remove irrelevant characters and standardize language, while stopwords are eliminated, and words are lemmatized to their root forms. This preprocessing ensures that the subsequent analysis focuses on the most meaningful content within each article.

The pipeline then identifies emerging **trending topics** in each category using TF-IDF vectorization, highlighting keywords that are most significant for that day. For example, if there’s a major development in artificial intelligence, the technology category might detect keywords like “AI regulation,” “OpenAI,” or “chatbot,” bringing relevant articles to the forefront.

The system selects the top articles representing these keywords and generates **concise, human-readable summaries** using OpenAI’s GPT model. These summaries distill complex stories into 2–3 sentence overviews that provide essential context without overwhelming the user with excessive detail.

After the summarization phase, the summarized articles and their metadata are securely stored in the PostgreSQL database with an association to the execution date. This makes it possible for users to not only explore today’s trends but also revisit and analyze historical data to see how topics have evolved over time.

The user then interacts with the **Streamlit-based web interface**, which displays a loading status indicating the progress of the pipeline. Once complete, the application provides a clean, interactive dashboard with tabs for each news category. Within each tab, the top trending summaries for the selected date are displayed in a user-friendly format, along with titles, publication dates, and clickable links to the original articles for those interested in reading full stories.

Moreover, the **date picker** integrated into the interface allows the user to select any previous date stored in the database, instantly retrieving and displaying historical trends. This feature is invaluable for professionals conducting research on media coverage patterns or tracking how stories develop over time.

For example, a journalist covering financial markets could use the application to monitor daily business news trends, quickly identifying emerging topics like corporate mergers, stock market fluctuations, or economic policies. By having concise summaries readily available, they can respond faster and prepare reports with greater context. Alternatively, a researcher might use the tool to study how public interest in certain topics—such as climate change or healthcare—has evolved over weeks or months.

This real-world use case highlights how *“Today In News”* transforms the daily news experience into a streamlined, efficient, and insightful process, saving users time, enhancing awareness, and supporting data-driven decision-making across various professions and interests.

# Technical Implementation

This section describes the architecture and workflow of the *“Today In News”* system, built in Python as a sequential pipeline. Each phase processes news data step by step: fetching articles, preprocessing text, detecting trends with TF-IDF, summarizing key stories using OpenAI’s GPT, and storing results in PostgreSQL. A Streamlit web interface presents summaries interactively, combining all components into a cohesive, automated system.

## 3.1 System Architecture Overview

The system follows a modular architecture, broken into five major phases:

1. **News Collection** (Phase 1)
2. **Preprocessing & Trend Detection** (Phase 2)
3. **Summarization** (Phase 3)
4. **Database Storage** (Phase 4)
5. **User Interface** (Phase 5)

The main.py or app.py file acts as the entry point, triggering all processes in sequence.

Each module is responsible for a defined task and communicates using structured data formats like CSV files or direct function calls. A PostgreSQL database persists results, while Streamlit serves as the UI.

## 3.2 System Architecture Diagram

A computer screen shot of a computer

AI-generated content may be incorrect.

Figure 1: Overview of the Today In News pipeline architecture.

## 3.3 Phase 1: News Collection Script

This phase is responsible for **automating the collection of daily news articles** across key categories including business, sports, health, technology, and entertainment. It forms the foundation of the pipeline by ensuring a fresh and comprehensive dataset of news stories is available each day for subsequent processing.

The main script, fetchers/news\_fetcher.py, acts as the **central coordinator**, orchestrating the entire fetching process by sequentially calling three dedicated fetcher scripts:

* guardian\_fetcher.py — connects to The Guardian API to retrieve recent articles for categories like business, sport, technology, culture, and lifestyle.
* newsapi\_fetcher.py — interacts with NewsAPI.org, searching for category-specific news articles in English from multiple global publishers.
* nyt\_fetcher.py — fetches articles from the New York Times Developer API, covering topics like business, technology, sports, arts, and health.

Each fetcher script performs several important tasks:

* **Constructs API requests** using API keys securely stored in the .env file, protecting sensitive credentials while enabling flexible deployment across different environments.
* **Handles pagination**, automatically iterating through multiple result pages to maximize the number of retrieved articles per category and ensure comprehensive data collection.
* **Standardizes output** by structuring the fetched articles into a consistent pandas DataFrame format, including essential fields like title, description, publication date, URL, and category.
* **Saves cleaned data** into CSV files organized by category, stored in the data/raw/ directory using filenames like business\_articles.csv, ensuring a structured, organized dataset ready for the preprocessing phase.

This phase guarantees that each execution of the pipeline fetches the **most current articles available from reliable sources**, laying the groundwork for accurate trend detection and summarization in later phases. By consolidating diverse news sources into a single, consistent format, this component significantly reduces redundancy, improves data quality, and ensures that the system captures a representative snapshot of global news each day.

## 3.4 Phase 2: Preprocessing & Trend Detection

This phase transforms raw news articles into clean, structured text suitable for analysis, and identifies the most significant keywords representing emerging trends within each category. It consists of two tightly connected steps: preprocessing and trend detection.

First, the **preprocessing step**, implemented in preprocess\_articles.py, systematically prepares the text by combining each article’s title and description, then applying advanced natural language processing techniques using the NLTK library. The preprocessing process includes:

* Converting text to lowercase for consistency.
* Removing punctuation and special characters to reduce noise.
* Tokenizing text into individual words.
* Removing common English stopwords, such as “the” or “and,” which do not contribute meaningfully to trend detection.
* Lemmatizing words using WordNetLemmatizer to reduce them to their base form (e.g., “running” → “run”), ensuring better matching during keyword analysis.

The cleaned text for each article is stored in a new column, cleaned\_text, and saved in updated CSV files in the data/processed/ directory. This prepares the data for accurate analysis by standardizing language and removing irrelevant content.

Next, the **trend detection step**, implemented in vectorizer.py, processes the cleaned articles using the TF-IDF (Term Frequency-Inverse Document Frequency) vectorization technique from Scikit-learn. This statistical method quantifies the importance of each word within a category by comparing its frequency in individual articles against its frequency across all articles. The script:

* Calculates TF-IDF scores for all words in the cleaned\_text column.
* Extracts the top 10 highest-scoring keywords per category, which best represent trending topics for that day.
* Prints these keywords for transparency and debugging.
* Selects up to three representative articles per category containing the top keywords, considering them as examples of the day’s trends.
* Saves these selected articles, along with relevant metadata and associated keywords, into CSV files in the data/trends/ directory.

This phase ensures that each day’s most relevant and frequently discussed topics within each news category are accurately identified, providing a clear foundation for generating meaningful summaries in the next phase.

## 3.5 Phase 3: Summarization

This phase focuses on transforming selected trending articles into **concise, human-readable summaries**, making it easier for users to quickly understand key stories without reading entire articles. The summarization step is implemented in the script summarize\_trends.py.

During this phase, the system reads the CSV files saved in the data/trends/ directory, each containing the top 3 articles selected per category based on trending keywords. For every article, the script:

* Combines the article’s title and description into a single text prompt, providing sufficient context for the summarization model.
* Sends the combined text to OpenAI’s GPT model (gpt-3.5-turbo) using the OpenAI API, requesting a clear summary in 5–7 sentences to ensure a balance of brevity and informativeness.
* Handles errors gracefully to avoid pipeline interruptions, logging any issues during API calls.
* Collects the generated summaries along with the article’s metadata (title, URL, publication date, category, and keyword) for later use.

The final output of this phase is a new set of CSV files saved in the data/final/ directory, each file containing summarized articles for a specific category. These summaries are designed to be easy to read and quickly convey the essence of each trending news story, preparing them for storage in the database and display in the user interface.

This phase plays a crucial role in **reducing information overload**, distilling complex stories into digestible insights, and ensuring that users receive clear, actionable summaries of the most important daily news.

## 3.6 Phase 4: Database Integration

This phase ensures that the summarized trending articles are **securely stored** in a structured, queryable format to enable historical analysis and fast data retrieval. The database integration is implemented in the script write\_read\_database.py.

After summarization, the system processes the CSV files in the data/final/ directory, which contain summaries for each news category. During this phase, the script:

* Connects to a PostgreSQL database using credentials stored securely in the .env file, ensuring sensitive information like database URLs remain protected.
* Creates a dedicated table for each news category if it does not already exist, with fields including an auto-incrementing ID, date, title, summary, URL, and the article’s original publication date.
* Deletes any existing records for the current execution date to avoid duplicate entries when the pipeline runs multiple times in a day, maintaining clean and accurate data.
* Inserts each summarized article into its corresponding category table, preserving both the content and relevant metadata.
* Provides feedback during processing by printing status updates, making it easy to monitor successful storage or troubleshoot issues.

In addition, the script includes **read functions** to retrieve stored trends by category and date. These functions allow the Streamlit web interface to display historical trending articles whenever a user selects a past date.

This phase guarantees **data persistence**, supports historical trend tracking, and creates a solid foundation for analyzing how topics evolve over time, all while keeping the database organized and optimized for daily use.

## 3.7 Phase 5: User Interface (Streamlit)

This final phase integrates the outputs of all previous stages into an **interactive, dynamic web application**, making it easy for users to explore trending news in an organized and engaging way. The interface is developed in app.py using the Streamlit framework, which allows rapid creation of modern data-driven apps with minimal code.

When the application starts, it automatically runs the entire pipeline—fetching fresh news, preprocessing text, detecting trends, summarizing articles, and storing results—if it hasn’t been executed yet in the current session. The UI displays a **status tracker** with real-time updates for each step, providing transparency and reassurance that the system is working as expected.

At the top of the interface, users are presented with an intuitive **date selector**, implemented with Streamlit’s date input component. This feature allows users to choose any day for which data is available, seamlessly switching between today’s trends or exploring historical summaries stored in the database.

Below the date picker, the interface features **tabbed navigation**, with one tab dedicated to each news category—Business, Sports, Technology, Entertainment, and Health. These tabs organize the content clearly, ensuring users can instantly access summaries in their areas of interest without sifting through unrelated topics.

Inside each tab, the app dynamically retrieves summarized articles from the PostgreSQL database for the selected date and category. It displays:

* The article’s title as a heading,
* The publication date for context,
* A concise, AI-generated summary for quick understanding,
* And a clickable link labeled “Read Full Article” directing users to the original source for deeper reading.

The UI gracefully handles cases where no trending articles exist for a selected date, notifying users with a friendly warning message instead of showing empty spaces.

This phase is essential for transforming complex, behind-the-scenes processes into a **simple, accessible experience**. By providing a visually appealing and easy-to-navigate platform, it empowers non-technical users, journalists, researchers, or casual readers to gain instant insights into the day’s most important news, while also enabling them to look back at how topics evolved over time.

Through Streamlit’s responsive design, the interface works smoothly on both desktop and mobile devices, ensuring accessibility across various platforms. This final phase completes the system by connecting powerful back-end processes to an approachable front-end, making Today In News a comprehensive solution for modern news trend analysis.

## 3.8 Folder Structure

Today-In-News/

├── fetchers/

│ ├── newsapi\_fetcher.py

│ ├── guardian\_fetcher.py

│ └── nyt\_fetcher.py

├── preprocessing/

│ ├── preprocess\_articles.py

│ └── vectorizer.py

├── summarizing/

│ └── summarize\_trends.py

├── db\_tools/

│ └── write\_read\_database.py

├── data/

│ ├── raw/

│ ├── processed/

│ ├── trends/

│ └── final/

├── app.py

├── .env

└── main.py

This structure allows each phase to be developed, tested, and maintained independently.

# Related Work

The Today In News project draws inspiration from and improves upon both existing industry tools and academic methods for news aggregation, trend detection, and summarization. By analyzing current solutions and research, this project aims to fill important gaps and provide a more complete, user-focused system.

## **4.1 Industry Tools Comparison**

Several well-known platforms offer features related to news aggregation and topic discovery, but they fall short in areas like automated summarization, transparent trend detection, or ease of use:

* **Google News** aggregates headlines from thousands of sources worldwide and uses ranking algorithms to personalize feeds based on user interests. However, it does not provide concise summaries of articles nor does it highlight why certain topics are trending, which can leave users overwhelmed by sheer volume.
* **Feedly** offers AI-based article clustering and smart feeds for professionals, but lacks fine-grained keyword extraction and direct summarization of trending topics. Its focus is on organizing articles into feeds rather than detecting daily trends.
* **Flipboard** organizes articles visually based on user-selected interests but doesn’t perform algorithmic analysis to extract or explain key trends. Articles appear in magazine-like layouts, which can be appealing but don’t reveal the core issues or keywords driving each topic.

Unlike these tools, Today In News combines **TF-IDF-based keyword extraction** and **AI-powered summarization** to automatically identify the most significant topics of the day and generate clear, human-readable summaries. This bridges the gap between discovery and comprehension, allowing users to not only see what’s trending but also quickly understand why it matters. Additionally, the use of **Streamlit** ensures the interface is lightweight, responsive, and portable, enabling faster deployment and easier access compared to heavier, ad-driven platforms.

## **4.2 Academic Approaches**

Academic research on news trend detection and summarization has produced powerful but often complex methodologies:

* **Named Entity Recognition (NER)** techniques identify important people, organizations, or locations in news text, but require complex entity linking and disambiguation for effective trend detection.
* **Topic modeling algorithms**, such as Latent Dirichlet Allocation (LDA), can automatically discover topics across large text corpora, but they can produce abstract or overlapping topics that are hard for non-experts to interpret.
* **Neural embedding models**, like BERT, offer advanced contextual understanding of text, but they come with high computational demands and require sophisticated post-processing to extract meaningful trends.

While these methods can achieve state-of-the-art performance, they are often impractical for building user-facing applications due to their complexity, interpretability challenges, and resource requirements.

In contrast, Today In News adopts a **TF-IDF-based approach** that balances interpretability, accuracy, and ease of implementation. By using straightforward keyword extraction, the system produces trends that are both transparent and understandable to general users. Combined with natural language summarization and a user-friendly UI, the project goes beyond traditional academic research by delivering an **end-to-end, usable solution**—one that not only extracts trends but also summarizes them, stores them for historical analysis, and presents them in an accessible web interface.

This integration of practical technologies with academic principles ensures that Today In News provides a comprehensive, effective tool for daily news consumption and analysis.

# Conclusion of Related Work

In summary, while existing industry tools like Google News, Feedly, and Flipboard offer valuable ways to aggregate and personalize news feeds, they often lack transparency in how trends are determined, and they rarely provide concise, AI-generated summaries to help users quickly grasp the essence of key stories. These platforms typically focus on breadth of coverage or aesthetic presentation rather than delivering actionable insights into why certain topics are trending.

On the other hand, academic research into news analysis has produced powerful techniques such as NER, topic modeling, and neural embeddings, which excel at uncovering patterns in large datasets. However, these methods often prioritize technical performance over user accessibility, making them difficult to translate into practical, day-to-day tools for general audiences. They can also introduce challenges related to complexity, high computational requirements, and difficulty in interpreting outputs.

Today In News positions itself at the **intersection of these two worlds**, drawing on proven academic methodologies like TF-IDF keyword extraction while emphasizing simplicity, transparency, and usability. By combining straightforward trend detection with human-readable AI summarization and a lightweight, intuitive interface, this project offers a unique solution that makes advanced news analysis accessible to everyone—from busy professionals to curious everyday readers. The system not only identifies what topics are trending but also explains them clearly, preserves historical data for future reference, and empowers users to explore news in a faster, more meaningful way.

Through this synthesis of research-based techniques and practical design, Today In News fills critical gaps left by both existing industry tools and academic approaches, providing a comprehensive, real-world application that addresses the modern challenge of information overload in an increasingly complex media landscape.

# Conclusion

The Today In News capstone project successfully demonstrates the development of a complete, end-to-end system for automating the collection, analysis, and presentation of trending news articles. By seamlessly integrating multiple technologies—including API-based data fetching, natural language preprocessing, TF-IDF keyword extraction, AI-powered summarization with OpenAI GPT, PostgreSQL database storage, and a Streamlit-based user interface—the project transforms a complex process into an efficient and user-friendly solution for modern news consumption.

From the initial phase of gathering fresh articles from trusted sources like NewsAPI, The Guardian, and The New York Times, through rigorous text cleaning and trend detection, to concise summarization and storage of key information, the system ensures that users receive only the most relevant, up-to-date insights. The interactive web interface empowers users to explore daily and historical trends effortlessly, offering clarity on what is happening and why it matters, all while saving time compared to traditional news reading.

This project not only provides technical value by showcasing a robust and modular pipeline architecture but also demonstrates how machine learning and AI can be applied to solve a real-world problem—information overload in today’s fast-paced digital world. The practical skills honed during the development process, including API integration, NLP techniques, database design, and user interface development, are directly transferable to professional roles in software engineering, data science, and AI application development.

By offering transparency in trend detection, readability through clear summarization, and ease of use with an intuitive interface, Today In News goes beyond typical news aggregators or research prototypes. It sets a new standard for user-centric news analysis tools, providing meaningful insights that can aid informed decision-making for professionals, students, and anyone who wants to stay up to date with key developments across various topics.

Overall, this capstone project demonstrates not just technical competence, but also a commitment to building solutions that address real user needs—paving the way for future improvements and potential expansion into larger-scale, commercially viable news analysis products.

# Future Developments

While the current implementation of the Today In News system successfully fulfills its primary objectives, there is significant potential to enhance its functionality, scalability, and user engagement through targeted future improvements. The following areas have been identified as key opportunities for development:

**1. Improved Trend Detection Techniques**

At present, the system relies on TF-IDF scores to detect trending topics. Future iterations could incorporate more advanced natural language processing techniques, such as topic modeling with Latent Dirichlet Allocation (LDA), keyword clustering methods, or Named Entity Recognition (NER), to capture more nuanced and accurate trends. Additionally, integrating real-time social media signals—such as trending hashtags from Twitter/X—could further improve the system’s ability to identify emerging topics with higher precision.

**2. Custom User Personalization**

Introducing user authentication and preference management would enable each individual to tailor their experience, selecting specific categories of interest, preferred news sources, or preferred languages. Furthermore, implementing a recommendation engine based on user behavior or explicit feedback could make the platform more engaging, delivering personalized trend summaries and encouraging regular use.

**3. Multilingual Support**

Expanding the system to handle non-English articles would greatly increase its utility for a global audience. By detecting the language of each article automatically and applying appropriate preprocessing and summarization pipelines, the system could deliver trend detection and summaries across multiple languages, powered by multilingual NLP models.

**4. Performance Optimization**

As the platform scales, optimizing performance will be critical. Migrating time-intensive tasks such as summarization and article fetching to asynchronous background jobs—using tools like Celery or FastAPI workers—would ensure a smoother user experience. Additionally, scheduling the entire pipeline to run periodically using task schedulers like cron, Apache Airflow, or Prefect would enable fully automated daily updates without manual intervention.

**5. Data Visualization and Dashboards+**

Adding visual analytics capabilities could make trends easier to interpret and more engaging. Integrating Streamlit’s charting features to visualize keyword frequency over time, and including elements like heatmaps, pie charts, or word clouds, would provide users with an interactive and insightful way to explore evolving news topics.

**6. Mobile-Responsive UI**

Optimizing the Streamlit interface for smartphones and tablets, or developing a dedicated mobile application using frameworks such as Flutter, would greatly enhance accessibility for users on the go. A mobile app could connect to the backend via REST APIs, providing a seamless and intuitive experience across devices.

**7. API Access**

Exposing RESTful or GraphQL APIs would allow third-party developers, researchers, or organizations to programmatically access trend summaries, extending the system’s utility beyond its web interface. This could enable integration with other applications, dashboards, or academic projects.

**8. Database Optimization and Historical Analytics**

Introducing advanced indexing and archiving strategies within the PostgreSQL database would support long-term trend analysis while maintaining performance as data volume grows. Adding analytics tools to track how specific keywords or topics evolve over weeks or months could provide valuable insights for journalists, researchers, and businesses monitoring public discourse.

# References & Bibliography

Below is a list of sources, libraries, tools, and APIs referenced or used during the implementation of the *Today In News* capstone project:

**Academic and Technical References:**

• Jurafsky, D., & Martin, J. H. (2021). *Speech and Language Processing* (3rd Edition Draft). [Link](https://web.stanford.edu/~jurafsky/slp3/)  
• Manning, C. D., Raghavan, P., & Schütze, H. (2008). *Introduction to Information Retrieval.* Cambridge University Press.

**Python Libraries and Tools:**

• pandas: Data manipulation and analysis – [pandas.pydata.org](https://pandas.pydata.org/)  
• scikit-learn: TF-IDF vectorization and preprocessing – [scikit-learn.org](https://scikit-learn.org/)  
• NLTK: Natural Language Toolkit for stopword removal, tokenization, and lemmatization – [nltk.org](https://www.nltk.org/)  
• psycopg2: PostgreSQL adapter for Python – [psycopg.org](https://www.psycopg.org/)  
• Streamlit: UI framework for building data apps – [streamlit.io](https://streamlit.io/)  
• dotenv: Load environment variables securely – [pypi.org/project/python-dotenv](https://pypi.org/project/python-dotenv/)

**APIs Used:**

• NewsAPI – [newsapi.org](https://newsapi.org/)  
• The Guardian API – [open-platform.theguardian.com](https://open-platform.theguardian.com/)  
• New York Times Developer API – [developer.nytimes.com](https://developer.nytimes.com/)  
• OpenAI GPT API (gpt-3.5-turbo) – [platform.openai.com/docs](https://platform.openai.com/docs)

**Development Environment:**

• Python 3.9+  
• PostgreSQL 15+  
• MacOS Terminal / zsh  
• Visual Studio Code  
• pgAdmin 4  
• Adobe Illustrator (for architectural and system diagrams)