Wikinews Insights: Real-time Correlation and Summarization of Wikipedia Pageviews and News Headlines

Kanchan Naik, Mrunali Katta, Prasad Shimpatwar, Yashasvi Kanchugantla
Dept. of Applied Data Science
San Jose State University

***Abstract*—The "WikiNews Insights" tool is designed to establish a real-time analytical connection between Wikipedia pageviews and trending news headlines. This offers valuable insights into public interests and behavior patterns. The primary goal is to build a scalable big data pipeline that processes Wikipedia data on an hourly basis to detect trending topics, which are then linked to the latest news articles sourced from major news APIs. This system highlights the relationship between public engagement and current events, showcasing the results through intuitive, real-time visualizations and interactive dashboards. Leveraging advanced data processing techniques and summarization algorithms, the project enhances the understanding of digital content consumption and serves various purposes, from journalistic research to academic studies. Furthermore, it provides organizations and policymakers with a robust tool to monitor public sentiment, facilitating real-time, data-driven decision-making. This innovative approach redefines media analytics by improving the responsiveness, accuracy, and relevance of media distribution, fostering a more informed and connected society.**

Key Terms - ***Decaying Window Algorithm, Locality Sensitive Hashing (LSH), Text Summarization, React.js, Django, Python, Kafka, Spark***

Github Link to the project

I. Introduction

In today's fast-paced digital age, understanding the dynamics of public interest is crucial for media, businesses, and policymakers. With billions of users accessing digital content daily, platforms like Wikipedia serve as valuable indicators of what topics are currently capturing the public's attention. Similarly, news headlines reflect the global pulse, providing insights into ongoing events and societal trends. However, the connection between these two data sources—Wikipedia pageviews and trending news articles—remains somewhat unexplored till date.

The **"WikiNews Insights"** project aims to bridge this gap by creating a real-time analytical framework that links trending topics on Wikipedia to the latest news headlines. By leveraging big data technologies, the project builds a scalable pipeline to process vast amounts of Wikipedia pageviews data and news articles, offering actionable insights into how public interest aligns with current events. This integration not only enriches the understanding of digital content consumption but also provides real-time insights for various stakeholders, including journalists, researchers, and businesses, enabling them to respond more effectively to emerging trends.

Through advanced data processing techniques, summarization algorithms, the "WikiNews Insights" tool delivers a comprehensive view of the relationship between public engagement and media coverage. This approach marks a significant step forward in media analytics, enhancing the relevance, timeliness, and impact of digital content in an increasingly connected world.

II. Motivation

Wikipedia is the most widely used web encyclopedia currently available. It has projects in 337 languages and are being actively maintained. There are 61M pages and about 238M page views are observed just on en.wikipedia.org. It is the source of all primary information one needs to start with. In this project, we wanted to understand and explore the relationship between public interest(as reflected in Wikipedia page views) and current events in the news. This project uniquely combines Wikipedia pageviews analysis with news aggregation, creating a novel approach to understanding public interest. Solving this use case is a good opportunity to apply several Big Data components learned in this course. Integrating different data sources helps make it useful for social researchers, journalists, and media persons.

III. Problem Statement

The aim of this project is to implement a "WikiNews Insights" system that combines real-time Wikipedia pageviews data with current news headlines to provide insights into public interest and information seeking behavior. It monitors statistics across multiple language editions, aggregates and analyzes news headlines, and provides concise summaries of correlated articles and news stories. The system ensures user's privacy and updates in real-time.

IV. Objective

- Identify and analyze trending Wikipedia topics in real-time.
- Integrate Wikipedia pageview data with current news headlines.
- Utilize big data techniques (e.g., Decaying Window) and summarization algorithms for efficient data processing.
- Support decision-making for media professionals and policymakers.

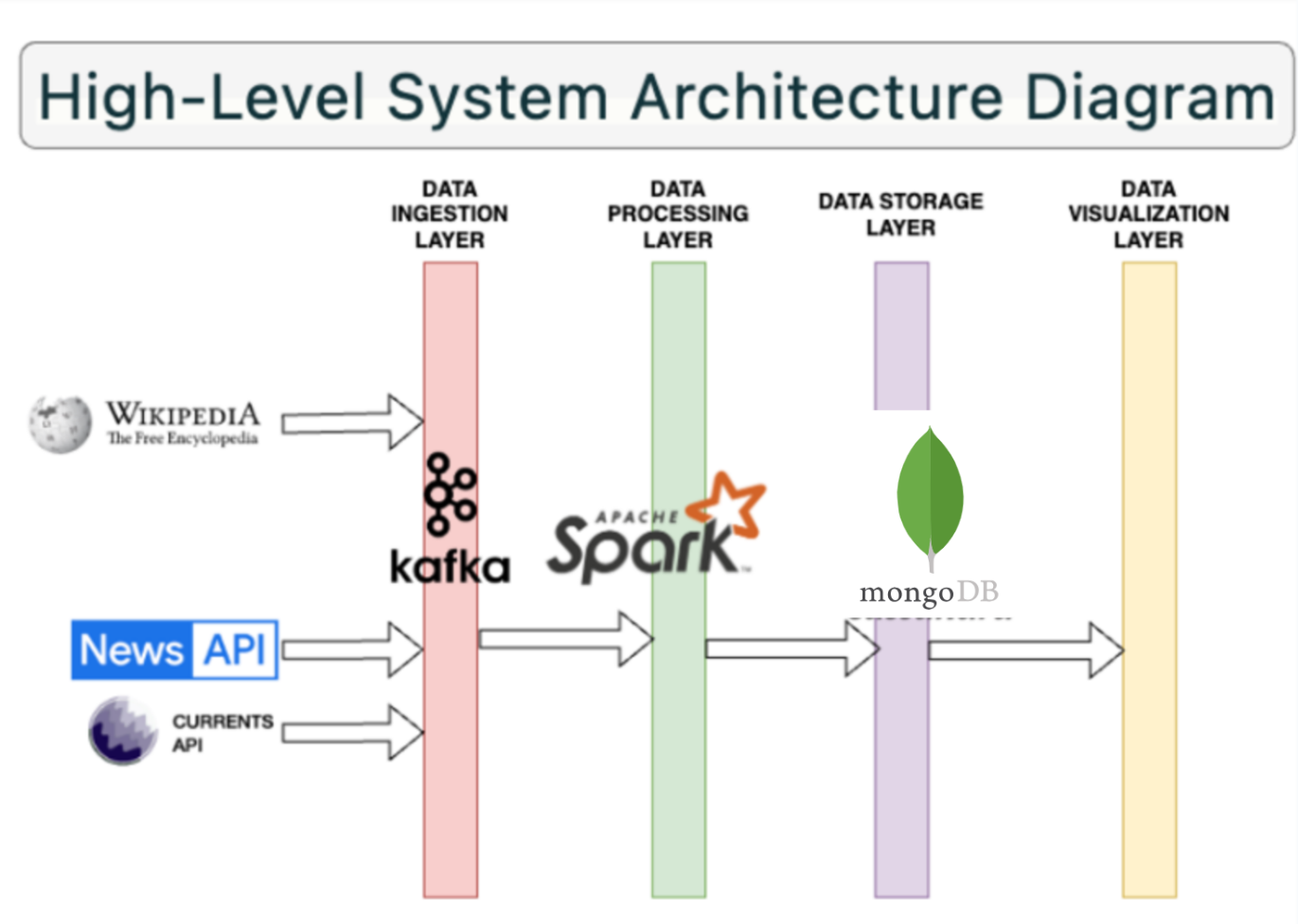


Fig. 1. Architecture Diagram

- Ensure privacy, transparency, and responsible data use in analytics.

V. Literature Review

The proposed *WikiNews Insights* system leverages methodologies from big data analytics, real-time processing, and public interest analysis. A study on decaying window algorithms introduces techniques for summarizing data streams with time decay, ensuring that recent trends are prioritized over historical data [1]. This is particularly relevant for analyzing hourly Wikipedia pageviews in real-time. Similarly, research on Locality-Sensitive Hashing (LSH) demonstrates its efficiency in clustering similar records, such as grouping related Wikipedia articles and news headlines, using minhashing for scalable similarity detection. [2].

In the context of financial applications, a study explores how Wikipedia pageviews correlate with stock market performance, finding that spikes in traffic mirror trends like Nasdaq movements but lack predictive power [3]. Another paper examines Wikipedia usage patterns before stock market shifts, highlighting how changes in views or edits reflect early public engagement with financial topics [4]. Additionally, a project using big data tools like Hive and MapReduce showcases scalable methods to process large-scale Wikipedia datasets, aligning with the project's need for efficient data pipelines [5]. These works collectively inform the design of *WikiNews Insights*, providing robust methodologies for real-time processing, clustering, and summarization of public engagement trends.

VI. Methodology

Scope

1) To create a data pipeline for collecting and processing Wikipedia pageviews statistics and news headlines.
2) To develop algorithms to identify and quantify correlations between spikes in Wikipedia pageviews and trending news topics.
3) To create concise, informative summaries of correlated Wikipedia articles and news stories to provide key information related to trending topics.
4) To validate the accuracy and reliability of the system's correlations and summaries through rigorous testing and comparison with known events.

5) To contribute to the broader understanding of digital information consumption patterns and their relationship to real-world events.

Data Collection

1) We are using programmatic access using Python to Wiki statistics using various APIs provided by Wikipedia and statistics calculated from data for the past one hour which is then stored in the system.
2) To fetch articles data we are calling various APIs to fetch articles data
3) Response contains article details such as title, description, source, images etc.
4) For each news article we are scrapping data using BeautifulSoup and collecting article content.

1) Wikipedia -
   a) **Source:**[https://stream.wikimedia.org/v2/stream/pageviews].
   b) **Format:** Zip files
   c) **Description:** Wikipedia pageview data using the Wikimedia EventStreams API
2) News Headlines and Top News data from News APIs
   a) **Source:**[https://newsapi.org/v2/everything?q=Erik-Menendez&X-Api-Key=⟨APIKEY⟩].
   b) **Format:** API Response
   c) **Description:** News APIs provides news snippets and data for recent news for given topic. We are passing keyword along with date and language specification to fetch news article details
3) News Headlines and Top News from Currents
   a) **Source:**[https://api.currentsapi.services/v1/latest-news?language=en&apiKey=⟨APIKEY⟩].
   b) **Format:** API Response
   c) **Description:** Currents APIs provides news snippets and data for recent news for given topic. We are passing keyword along with date and language specification to fetch news article details
4) Extract and process raw JSON.
5) Web Crawling to fetch content of original news articles for news summarization.

Data Pre-Processing

1) **Data Extraction**
   a) Extracting relevant fields from News articles and Wiki logs.
   b) Extracting Title from web page. c) To extract main content from crawled web pages without advertisements and page elements.
2) **Data Cleaning**
   a) Removing invalid entries like internal logs or where the title or data source is not mentioned.
   b) To erase duplicate data from logs and news articles.
   c) Aggregating Wikipedia data by page views, categories and weights calculated by decaying window algorithm.
3) **Text Cleaning**
   a) To remove HTML tags or other special characters from News articles.
   b) Not considering other language text from the articles. c) Filtering only english language pages and news articles
4) Standardize the Date and Time to UTC or common time zone.

Data Processing

1) Tokenize the text into words or sub-words.
2) Remove stop words.
3) Apply stemming or lemmatization to reduce words to their base forms.
4) Perform named entity recognition to identify people, places, organizations, etc.
5) Extract Topic, Categories, and Sentiment Score.
6) Correlation coefficient between wiki page view spike and current news trends.
7) Structure the data for efficient storage and retrieval.

News Wiki Correlation

1) Implement algorithms to identify spikes in Wikipedia page views.
2) Calculate correlation coefficients between page view spikes and relevant news topics.
3) Fetch the Top news articles from News websites and display the most relevant news articles.

Content Summarization

1) Generate a summary of news articles and wiki articles to highlight current events.
2) We are using LED [Long Form Encoding Decoding] summarizer.
3) This summarizer combines the approach of extractive and abstractive summarizing technique.

Data storage and results

1) We are storing summarized data and wiki spike-related information in distributed databases such as MongoDB.
2) The summaries are displayed using an application endpoint built using Python or Django.

Evaluation Methods

1) **Human Evaluation** - For summarization of news topics we used the help of human evaluation. i.e. we checked by ourselves if a generated summary is of good quality or not.
2) **Temporal Alignment Accuracy** - We checked page views spike and Time of News article publication aligns or not.
3) **Performance and Latency** - While the first time generating summary latency of a system is very high so we fetched all the articles and generated corresponding summaries. To optimize this we stored the data in MongoDB and if a summary for a particular topic is generated within one hour we showed the same summary to the user.

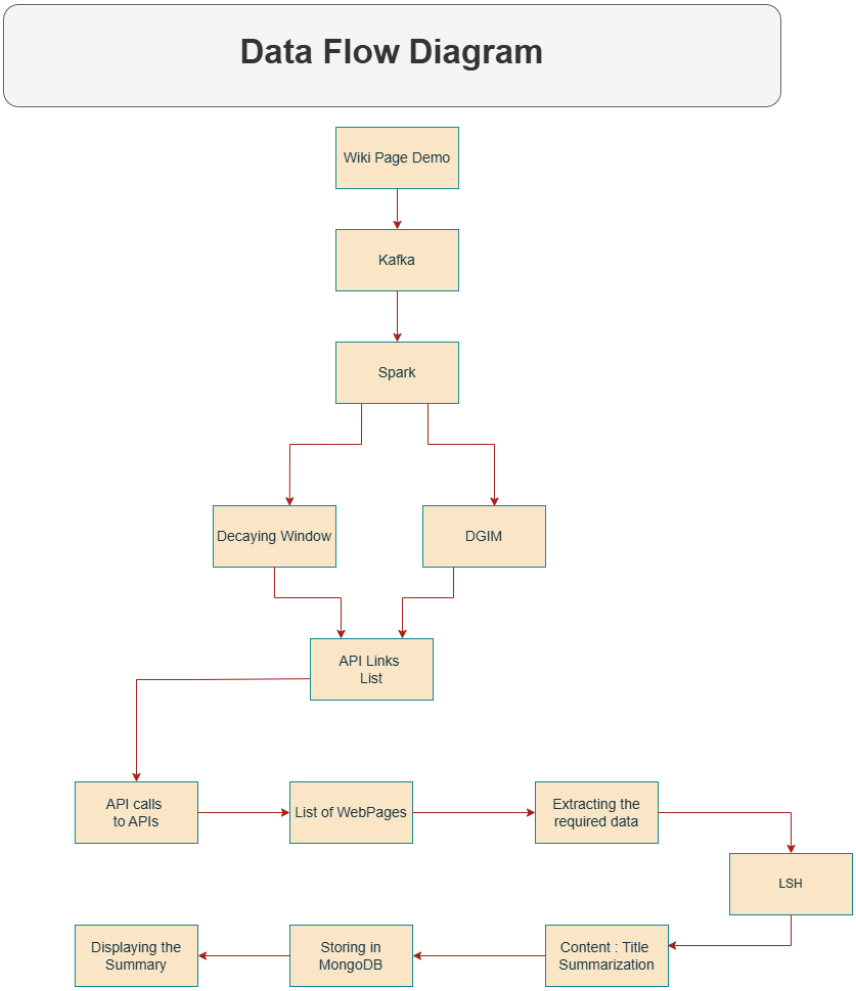


Fig. 2. Data Flow Diagram

4) **Comparison by different α values in decaying window:** To evaluate the performance of the decaying window we compared weights calculated on the same data using different α values. For this, we compared α - 0.95, 0.9, 0.85, and 0.8 values. we observed different trends with different α values which are explained in findings.

A. *DataModel*

We have four entities in our data model.

1) Wiki PageViews: After fetching the data from the wiki page dump and running the decaying window algorithm on it we are storing top k (k=10) page view data along with its decayed average in MongoDB. This data is further used to show a trending topic list.
2) News Articles: After getting trending topics for each trending topic, if a user visits we fetch trending news articles from news websites and store the title, source, description, URL, image URL, and content snippet in the data database.
3) Correlations: To establish co-relation between topic and articles we are keeping separate tables to store correlation.
4) Summary: Once the summary is fetched for articles we store it in db with summary ID, topic ID, and articles processed linked to it.

B. *WorkFlow*

1) First we are collecting data from the wiki dump by the hour in the form of a zip folder.
2) After receiving the zip folder, we programmatically extract it and read each line from the file.
3) We are then filtering this line based on language domain. If the language domain is 'en' then only we are selecting the line for further processing.
4) Once the line passes this filtering check, we publish it on the Kafka topic to simulate streaming.
5) Using spark we are accessing this line published on the kafka topic and calculating the decayed weight for the same. The initial decay rate 0.9 is considered.
6) We used $S_t = \alpha \cdot x_t + (1 - \alpha) \cdot S_{t-1}$ for calculating the decaying average.

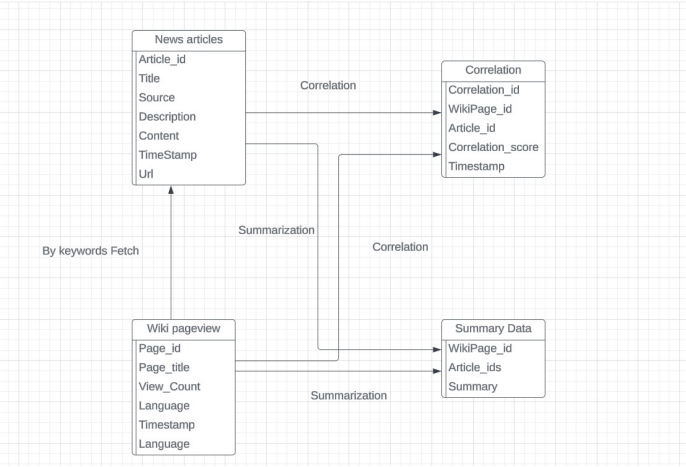
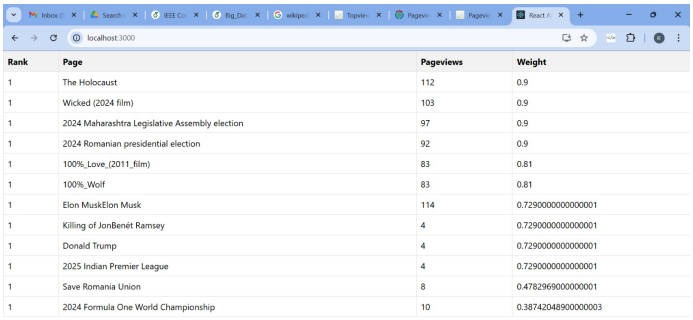


Fig. 3. DataModel Diagram



Rank	Page	Pageviews	Weight
1	The Holocaust	112	0.9
1	Wicked (2024 film)	103	0.9
1	2024 Maharashtra Legislative Assembly election	97	0.9
1	2024 Romanian presidential election	92	0.9
1	100%_Love_(2011_film)	83	0.81
1	100%_Wolf	83	0.81
1	Elon MuskElon Musk	114	0.7290000000000001
1	Killing of JonBenét Ramsey	4	0.7290000000000001
1	Donald Trump	4	0.7290000000000001
1	2025 Indian Premier League	4	0.7290000000000001
1	Save Romania Union	8	0.4782969000000001
1	2024 Formula One World Championship	10	0.38742048900000003

7) This decaying average and trending topics are stored in MongoDB.
8) Once these trending topics are filtered we show it on the dashboard.
9) When a user clicks on any topic we are fetching news articles related to that topic and showing the most recent news articles.
10) News APIs provide abstract details of articles hence we are calling actual news URLs and fetching the content of an article using web crawling.
11) The scraped content is being cleaned and we are extracting the title and main content of the new articles which is further processed for summarization.
12) We use Locality Sensitive hashing to evaluate the news articles for summary. Only news articles with non similar contents a re filtered for summarization.
13) Once a summary is generated from the summarizer we store it in MongoDB.

C. *Deliverables*

1) A comprehensive report that summarizes the project's development, key technologies, methodologies etc.
2) The web application that summarizes the news articles.
3) Slides to give an overview on the project.

VII. Implementation

1) **Decaying Window:**
In our WikiNews Insights project, we employed a decaying window algorithm using Apache Spark and Kafka to analyze real-time data streams from Wikipedia and news sources. This approach utilizes a decaying window formula to adjust the relevance of data over time, enhancing our ability to spotlight trending topics effectively. By dynamically updating our dashboard with these insights, we provide users with timely and relevant information, facilitating better engagement and informed decision-making based on the most impact-full news trends.

2) **LSH Algorithm:**
We implemented Locality Sensistive Hashing (LSH), for the news articles of each page, in order to reduce the number of articles used to generate summary. This is done because, latest news, fetched from recent articles is highly likely to have similar data. LSH module is created by shingling as 5 grams- 5 words are considered as one shingle. The threshold for similarity is customizable. Below are figures of article nodes for different threshold values. The datasketch library used for LSH implementation customizes the band width based on the threshold, such that the threshold for Locality-Sensitive Hashing (LSH) is given by:

$$\text{threshold} \approx \left(\frac{1}{b}\right)^{\frac{1}{r}}$$

where:

$$b \times r = \text{num\_perm}$$

with:

r : Rows per band.

b : Number of bands.

3) **API Calls:**
The use of APIs is fundamental to access and integrat real-time data. This system harnesses the capabilities of RESTful APIs provided by various news organizations and the Wikimedia EventStreams API to continuously collect fresh data concerning global events and Wikipedia page activities. By leveraging these APIs, WikiNews Insights can dynamically fetch the latest news articles, ensuring that the analysis reflects the most current public interests and trends. This seamless integration of real-time data streams allows for more accurate and timely insights, which is critical for media outlets, researchers, and policymakers who rely on up-to-date information to make informed decisions.
4) **Extraction using web scraping:**
In the WikiNews Insights project, web scraping is implemented for enhancing our data collection beyond what APIs offer. Using tools like Beautiful Soup, we extracted web pages to extract detailed texts, multimedia, and rich

metadata. This comprehensive approach allowed us to capture nuanced content and contextual details, enriching our dataset for a deeper analysis of current trends. By leveraging real-time data from primary sources, our system remains up-to-date and relevant, providing us with a competitive edge in digital media analysis.

5) **Summarizer:** We utilized the LEDTokenizer and LEDForConditionalGeneration [6] from transformer models to efficiently summarize extensive articles. These tools, part of the advanced Longformer-Encoder-Decoder model, allow us to handle documents with thousands of words, transforming them into concise, insightful summaries. By integrating these capabilities into our system, we ensure that information is not only accessible and engaging for our users but also delivered in real-time. This approach significantly enhances user interaction and comprehension of complex information, improving the overall user experience.
6) **Dashboard using Reactjs:** The React.js dashboard displays news article topics and summaries in real-time, providing an interactive and engaging user experience. This setup allows users to explore data dynamically and stay updated with the latest insights as they unfold.

VIII. Findings and Results

1) **Trending Topics:** Using the Decaying window algorithm we are calculating weights for trending topics. The top 10 topics are displayed on the dashboard along with the topic title, weight, and page views. We are filtering topics that have very few page views as they are influencing the algorithm more and which is giving incorrect results. For decaying window algorithm we used$\mathbf{S}_t = \alpha \cdot \mathbf{x}_t + (1 - \alpha) \cdot \mathbf{S}_{t-1}$ formula where α is a decaying constant that specifies the influence of the topic. We tried changing α values to check the performance of the algorithm concerning the alpha value.

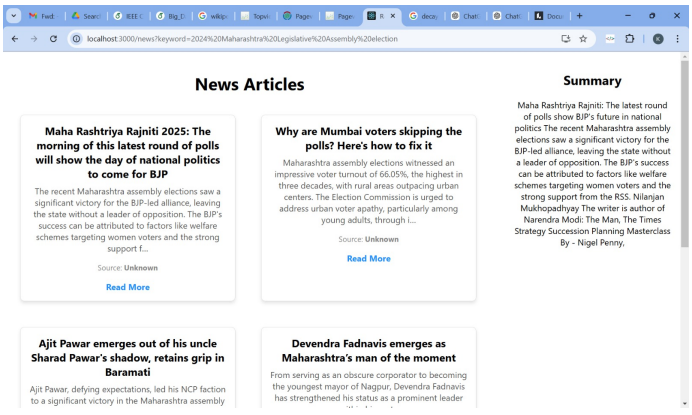


Fig. 5. News Articles Summarization

2) **Finding of Decaying Window:** α values set to 0.9 and 0.95 we kept high values of α to reduce the influence of recent data on weights. This we did to capture slower trends and avoid the short-term influence of searches. We observed that for these topics we found relevant news articles. However, it was not able to capture very recent hot topics. α values set to 0.8 and 0.85 Setting these values was able to capture new searches quickly but the problem with this was even if these topics were trending we were not able to find relevant news articles. This could be due to since recent topics get more importance they were shown on topi but there were not many articles published in a short time. Pages for which count was less were influencing weights and trending topics to avoid this problem we considered weights of topics having more than a certain number of page views.
3) **Locality-Sensitive Hashing (LSH) for retaining the variety of articles shown:** Using Locality Sensitive Hashing, as discussed above, we have created buckets for similar articles. One article from each bucket is shown the news articles of a Wikipedia page. For experimentation, we have shown 3 threshold values of similarity to bucket the bands in LSH and we can see that the number of clusters have decreased with decrease in threshold value. threshold value. threshold value.

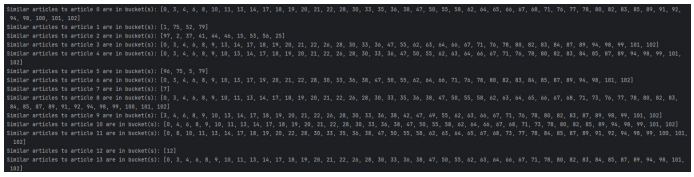


Fig. 6. Similar News article clusters upon Locality Sensitive Hashing

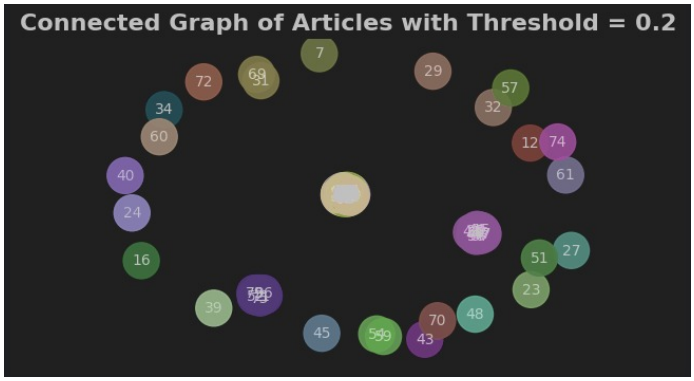


Fig. 7. LSH with Threshold: 0.2

4) **Summarizer:** Using the LEDTokenizer and LEDForConditionalGeneration significantly enhances our ability to digest and summarize lengthy documents. These tools, which are adept at processing extensive texts up to thousands of words long, efficiently produce concise, insightful summaries. This technology not only speeds up data processing but also makes complex information

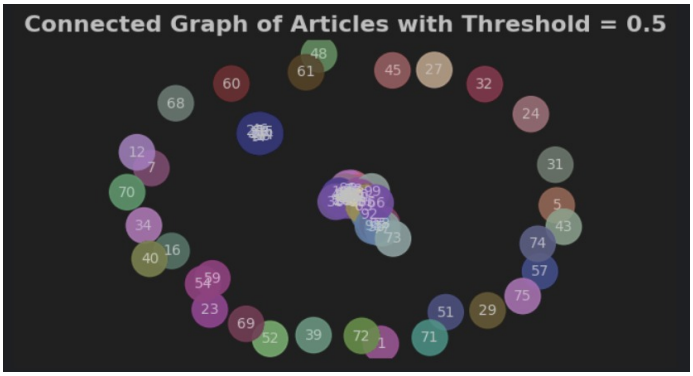


Fig. 8. LSH with Threshold: 0.5

more accessible and engaging for users. The integration of these advanced summarization capabilities into our real-time system has markedly improved user interaction with and understanding of intricate content, leading to a better overall user experience.

IX. Tools and Technologies

1) **Python:** Python is used for developing the Summarizer tool and all the other algorithms, implementing complex algorithms like LSH. It integrates with other tools and technologies seamlessly, providing flexibility and versatility in data processing and other tasks.
2) **Pyspark:** PySpark is used to process the large Wikipedia pageviews data in parallel, performing complex data analysis and transformations. It's ideal for big data processing due to its scalability, enabling efficient data processing and analysis. Also, PySpark is used to identify top pages by aggregating and categorizing news articles and pageviews.
3) **Apache Kafka:** We used Apache Kafka to stream Wikipedia pageviews data in real-time, enabling the collection and processing of statistics and news headlines. Then we have created a data pipeline that leverages Kafka's producer-consumer architecture, where producers write to topics and consumers read from them. This allows for efficient and scalable data processing and analysis.
4) **Apache Spark:** Using Apache Spark, we have created a session to process Wikipedia pageviews data. We have implemented a Parquet file-based solution to store and retrieve the data efficiently. Additionally, we have implemented a decaying window algorithm to analyze the data and identify trends.
5) **Summarizer:** The LEDTokenizer and LEDForConditionalGeneration from the Longformer-Encoder-Decoder (LED) model are designed for efficiently summarizing long articles. This model, part of the Hugging Face's Transformers library, can handle texts up to 16,384 tokens due to its scalable attention mechanism. It typically produces summaries of a few hundred words, focusing on capturing the essence of extensive texts and discarding extra detail. It works as both an Extractive and Abstractive summarizer. This makes it particularly effective for summarizing lengthy articles where crucial information is spread across the text.
6) **Locality-Sensitive Hashing (LSH):** Locality-Sensitive Hashing (LSH) is a technique used for dimensionality reduction within data, particularly effective in high-dimensional spaces. It hashes similar input items into the same "buckets" with high probability, which helps in finding approximate nearest neighbors. LSH is commonly utilized in systems where rapid processing of large datasets is required, such as image retrieval, duplicate detection, or document similarity in large text corpora. This makes it an useful tool for enhancing performance and efficiency in data-intensive applications.
7) **MongoDB:** We utilized MongoDB as our distributed database to efficiently store and manage the summarized data and spikes in Wikipedia pageviews. This setup allows us to cache certain results for up to an hour, reducing the need to recompute the same data repeatedly. By leveraging MongoDB's capabilities, we ensure that our system remains efficient and responsive, providing timely and accurate information to users without unnecessary processing delays.
8) **ReactJS:** We have used ReactJS to build the web page that presents topics and summaries of news articles. ReactJS, known for its efficiency in updating and rendering the right components when data changes, is ideal for handling the real-time nature of our data flows. This framework supports our goal to provide a dynamic user experience, where interactive visualizations and the latest summarized content are seamlessly displayed, enabling users to engage deeply with the news insights we generate.
9) **Beautiful Soup:** Beautiful Soup plays a critical role in web scraping, enabling us to extract and parse data from various news websites and Wikipedia pages efficiently. We use this library to navigate the HTML content of these pages and selectively pull out relevant text, links, and other data needed for our analysis. By integrating BeautifulSoup into our data collection pipeline, we ensured that the information presented on our dashboard is both current and richly detailed, enhancing the overall quality and depth of the insights we offer to users.
10) **Django:**We utilized Django to create a robust backend that effectively manages and processes large datasets from Wikipedia and various news APIs. We also developed RESTful APIs using Django to facilitate real-time data exchange between the front end and backend, ensuring dynamic and interactive user experiences.
11) **Git:** We used git for version control where we uploaded all our source code.
12) **Jira:** We used JIRA by Atlassian for maintaining task list and dividing tasks among team members.

X. Recommendations

1) The system we have implemented is for all the english Wikipedia pages. While this gives an overall view on how they correlate with news trends, applying it for domain specific pages and news creates a viable product for media who cover niche topics of interest.
2) Wikipedia page views is a representation of interest in the society. Content creators can leverage this to understand the pulse of different focused domains and cater accordingly.

XI. Version Control

In our WikiNews Insights project, we employed Git as our primary version control system to manage the development of software components efficiently. This system was crucial for coordinating team efforts, tracking revisions, and maintaining a historical record of the entire project development process. By hosting our repository on GitHub, we facilitated collaborative reviews and contributions, ensuring that all team members could work simultaneously without conflicts. Utilizing Git allowed us to revert to previous versions when necessary and understand the evolution of our code through branches and commits. This practice not only increased our productivity but also ensured the integrity and continuity of our codebase throughout the project lifecycle.

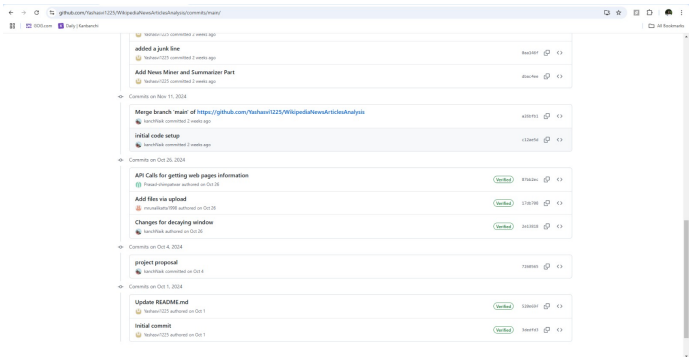


Fig. 9. Version Control

XII. Lessons Learned

We've learned crucial lessons about managing large data sets and the importance of integrating various technologies. Our experience highlighted the value of real-time data processing and the effectiveness of using advanced analytical methods to understand public interests. We discovered that decaying window algorithms are vital for capturing current trends without being overwhelmed by fleeting data spikes. Additionally, our work underscored the need for a robust and flexible infrastructure to adapt to rapidly changing data landscapes, proving that continuous innovation and learning are key to success in data analytics.

XIII. Prospects of winning Competition or Publication

This project has practical applications of getting topics or domain-specific trend analysis and information correlation hence we can generalize this solution to establish the correlation of trending topics with recent news articles published, therefore, we think that we can publish a paper on generalized solutions.

XIV. Innovation

We have implemented the "WikiNews Insights" tool, marking the first comprehensive system to seamlessly summarize and correlate real-time Wikipedia pageviews with news headlines. This innovation enables a direct analysis for showcasing the public interest measured through digital content interactions and current global events, providing stakeholders with unprecedented insights for strategic decision-making and trend analysis.

XV. Teamwork

In our project, everyone contributed their unique talents, guided by agile practices and pair programming to enhance our teamwork. We strategically divided responsibilities to draw on our varied strengths, boosting both the efficiency and effectiveness of our work. This collaborative approach fostered a nurturing and supportive environment, sparking open communication and creativity—crucial drivers behind our project's success.

1. Implementation to ingest stream data from wiki	Yashasvi Kanchugantla
2. Decaying window algorithm	Kanchan Ashok Naik
3. Data cleaning	Prasad Pramod Shimpatwar
4. Correlation evaluator	Yashasvi Kanchugantla
5. Summarizer	Prasad Pramod Shimpatwar
6. Data Aggregation and processing (Tagging, categorizing other required things)	Mrunali Katta
7. News and currents api call and filtering relevant results	Mrunali Katta
8. Web crawling to fetch and extract data from original news articles	Yashasvi Kanchugantla
9. UI desktop in Django for displaying output	Kanchan Ashok Naik
10. Data base storage	Kanchan Ashok Naik
11. LSH algorithm	Yashasvi Kanchugantla

Fig. 10. Team Members and Roles

XVI. Technical Difficulties

1) Managing high-throughput, real-time data pipelines. Each hour data has taken more than 2hrs to process.
2) Building advanced NLP-based text summarization - Large context models were used, instead of chunking articles.

3) Exposing them for displaying results using React, Django framework. Integrating multi-source data feeds. Different news API have different structure.
4) Summarization modules - ran out of memory / RAM while running the model to generate attention values for explainability.

XVII. Practiced Pair programming

We used pair programming to boost both collaboration and the quality of our work. This method allowed team members to pair up, share insights, and tackle coding challenges together, enhancing our collective understanding and efficiency. The experience not only improved our code but also strengthened our team dynamics, making the development process more engaging and productive.

We also used Copilot app - Codeium with our IDE. It helped us understand each other's existing codes. The chat, when given with explain command, gives a short summary of the code, improving our understandability.

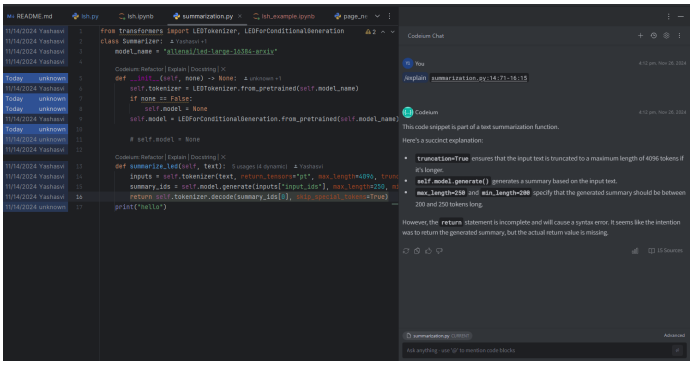


Fig. 11. Codeium chat on Pycharm IDE

XVIII. Practiced Agile

Agile methodologies to enhance our workflow and adaptability. This approach enabled us to iterate quickly, respond to changes efficiently, and continuously improve our project through regular feedback loops. By practicing Agile, we fostered a flexible and dynamic team environment that was crucial for navigating the complexities of real-time data analysis and application development.

XIX. Used Grammarly

We used the free version of Grammarly to correct tense and grammar.

XX. Used LaTeX

Yes, we have used LaTeX (Overleaf) to create the report. Proof is attached with the image showing Grammarly in use.

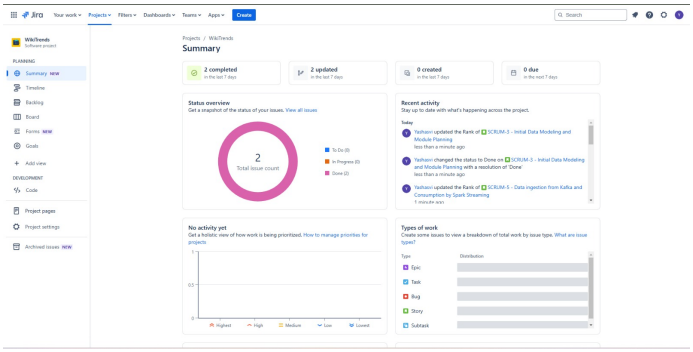


Fig. 12. JIRA Dashboard

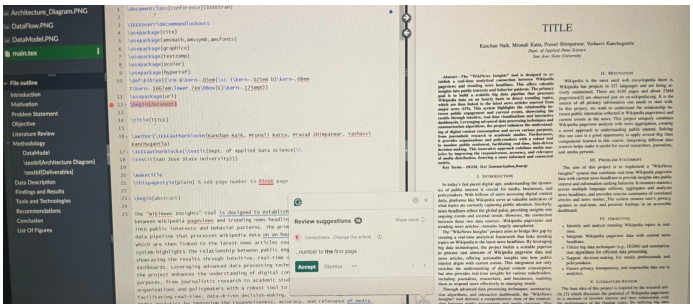


Fig. 13. Use of free version of Grammarly

XXI. Used creative Presentation Techniques

We used Generative Ai for generating slide deck and some images to add on slide deck. We also used Generative Ai to create elevator pitch video from narrative provided. Narrative was generated from our project report using GenAI. We had to modify narrative to some extend but majority of narrative was provided by GenAI. Link for tool we used for creating elevator pitch video [https://simpleshow.com/]

XXII. Use of Streaming Algorithms

1) **Decaying Window Algorithm** is used to assign page views in a decaying importance fashion. We have given a decaying constant of 0.9 for those pages that are not occurring in the next dump.

XXIII. Use of Stream Processing Frameworks such as Spark, Flink, Kafka

1) **Kafka** is used to process all the page view dumps exposed by Wikipedia every hour. They are then consumed by **Spark Streaming** object to filter using desired criteria like Domain of Wikipedia page, language, etc.
2) **Spark and Spark Streaming** are used to consume page views data from Kafka and apply various Big data algorithms - the Decaying Window algorithm.

XXIV. Use of Locality Sensitive Hashing

Several articles fetched for each page title from the news Apis can get very large i.e. $\tilde{2}00$ to $\tilde{5}000$ articles. So we have used LSH to cluster similar articles and picked only one article

per cluster, to maintain variation in the articles. You can refer to Implementation for more information.

XXV. Use of Privacy techniques

This is not applicable to our project because our dataset contains no sensitive data and we are not generating any sensitive data. However, to implement this algorithm, we still treated the page title, device, and time as quasi-identifiers and generated k-1 redundant records.

XXVI. Other tools and techniques covered in the course and not included in other criteria

1) **Tokenization and Vectorization of text:** Converting to Tokens and vectors for generating summaries through Encoder models was necessary. We picked up a Longformer Encoder-Decoder (LED), as it had a large context window. To encode large articles in one go, we used max_length 4096.
2) **Explainable AI:** Used Explainable AI for generating scores and how algorithms determine the relevance of Wikipedia page views and news trends we calculated attention scores using library named captum.attr to understand what tokens are given more importance.

Fig. 14. Explainable AI output

XXVII. Use of new tools that were not used for any of the HW

1) **Beautiful Soup:** BeautifulSoup of the bs4 library is used for fetching total content from the articles. News API endpoints have content-length limits.
2) **Dashboard using Reactjs** are used to display the page views and news summaries.
3) **Networkx** is used to visualize the clusters of similar news articles classified by Locality Sensitive Hashing for every Wikipedia page.

XXVIII. Conclusion

The "WikiNews Insights" project effectively connects Wikipedia pageviews with trending news headlines, offering meaningful insights into what captures public attention. Using real-time data pipelines and advanced summarization techniques, the system efficiently processes large volumes of data, making it easy to understand the correlation between public interest and current events. The interactive display of results provide an intuitive way for users such as journalists, researchers, and policymakers to explore these trends and make informed decisions. The system ensures scalability and reliability by employing innovative technologies like Apache Kafka, PySpark, and Decaying Window Algorithm, even with continuous data streams. This project highlights the power of big data to uncover patterns in information consumption and helps build a deeper understanding of societal interests. With further refinements, it has the potential to broaden its impact across multiple languages and diverse audiences.

XXIX. List Of Figures

List of Figures

References

[1] A. Bifet and R. Kirkby, "Data stream mining: A practical approach," *arXiv*, 2015.
[2] P. Indyk and R. Motwani, "Approximate nearest neighbors: Towards removing the curse of dimensionality," *International Symposium on Algorithms and Computation*, 1998.
[3] T. Preis, H. Moat, and H. E. Stanley, "Quantifying trading behavior in financial markets using google trends," *Scientific Reports*, 2013.
[4] T. Yasseri, R. Sumi, and A. Rung, "Dynamics of conflicts in wikipedia," *PLOS ONE*, 2012.
[5] G. Community, "Big data analysis of wikipedia." Project repository.
[6] K. Choromanski, V. Likhosherstov, D. Dohan, X. Song, A. Gane, T. Sarlos, P. Hawkins, J. Davis, A. Mohiuddin, Kaiser, D. Belanger, L. Colwell, and A. Weller, "Rethinking attention with performers," *arXiv preprint arXiv:2004.05150*, 2020.