RECOMMENDATION SYSTEM

Team Name: Analysis Wizards

Sanjay.B

Muthiah Muralitharan.U

Simson Anto Rai.I

***PROBLEM UNDERSTANDING & MODEL BUILDING:***

* To build an efficient predictive model using Machine Learning (ML), Deep Learning (DL), Natural Language Processing (NLP), or Large Language Models (LLM).
* The goal is to develop a well-optimized solution that is accurate, computationally efficient, interpretable, and user-friendly with an innovative frontend for visualization

***FEATURE ENGINEERING & DATA PREPROCESSING:***

* **Remove Unnecessary Columns**:Columns like url might be dropped if not needed for modeling.
* **Handle Missing Values**:Although all values are non-null, the authors and tags fields contain empty lists, which may need special handling.
* **Convert Timestamp**:Convert timestamp to a proper datetime object and extract useful features like,year, month, day, hour, day of week, etc.
* **Text Cleaning**:For title and text, perform,lowercasing removing punctuation, numbers, stopwords, tokenization lemmatization or stemming.

***MODEL PERFORMANCE & ACCURACY :***

***COMPUTATIONAL EFFICIENCY & OPTIMIZATION :***

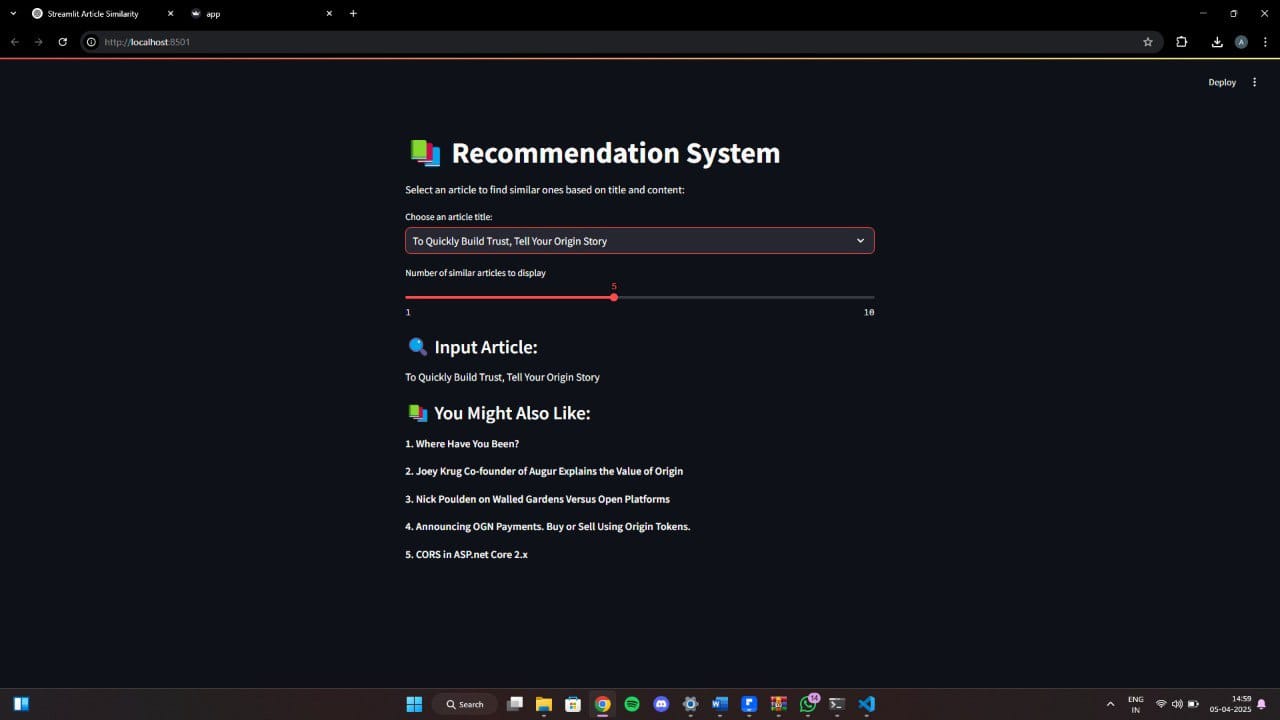
* **Caching with @st.cache\_data and @st.cache\_resource**: The use of Streamlit’s caching decorators ensures that expensive operations like data loading (load\_data()) and TF-IDF vectorization (compute\_tfidf()) are only computed once unless the inputs change. This significantly improves the runtime performance when the user interacts with the app repeatedly.
* **Efficient Text Vectorization with TF-IDF:** The TfidfVectorizer with a max\_features=5000 setting limits the vocabulary size, balancing between model accuracy and speed. This reduces the dimensionality of the vector space, leading to faster similarity computations.
* **Optimized Similarity Computation**: The cosine\_similarity() function is applied only between the selected article and the rest of the dataset, rather than computing pairwise similarities for the entire matrix, which saves a lot of computation time.
* **Flattening and Index Sorting Logic**: The combination of .flatten() and .argsort() in get\_similar\_articles() provides an efficient way to retrieve top similar articles without additional libraries or overhead, making the recommendation process quick and responsive.

***INNOVATION& CREATIVITY:***

* **Combining Title and Content for Richer Context**: Instead of relying solely on article titles or content, the model innovatively merges both (title + text) into a single field (cleaned\_content), enhancing the contextual understanding and quality of recommendations.
* **Interactive Frontend Integration with Streamlit**: The use of Streamlit to wrap a machine learning model into a user-friendly web app showcases innovation in accessibility—allowing non-technical users to explore ML-powered recommendations in real-time.
* **Customizable Output with Slider**: Giving users the control to choose how many similar articles to view (using a slider) adds personalization, making the system flexible and interactive—a smart design choice that improves user engagement.
* **Simplicity with Effectiveness**: Instead of complex deep learning models, it uses a simple yet effective TF-IDF + cosine similarity approach. This highlights thoughtful model creation where simplicity meets solid performance—especially for text-based recommendation tasks.
* **Scalable Framework for Future Enhancements**: The modular structure of the code (with separate functions for loading, processing, recommending) allows for easy future upgrades—like adding NLP preprocessing, clustering, or BERT embeddings—showing foresight in model creation.

***FRONTEND IMPLIMENTATION & VISUALIZATION :***

* **User-Friendly Interface with Streamlit**: The frontend is built using Streamlit, which provides a clean and interactive interface. Users can easily select an article title from a dropdown menu, making the input process intuitive and seamless.
* **Dynamic Similarity Control**: A slider widget is used to let users specify how many similar articles (up to 10) they want to view. This dynamic control allows users to customize the number of recommendations according to their preference.
* **Clear Visualization of Output**: The interface clearly separates the input article and the recommended articles using section headers and icons. This improves readability and helps users quickly understand the output.
* **Dark Theme Aesthetic**: The dark-themed UI enhances the visual appeal and provides a modern look, which can reduce eye strain and improve the user experience, especially in low-light environments.



***MODEL EXPLAINABILITY & INTERPRETABILITY:***

* **Transparent Similarity Measure (Cosine Similarity)**: The model uses cosine similarity, a well-understood and interpretable metric that calculates how close two text vectors are in meaning. This makes it easy to explain why certain articles are recommended—because they have similar textual content.
* **TF-IDF Feature Representation**: By using TF-IDF (Term Frequency-Inverse Document Frequency), the model gives more weight to important and unique words in the dataset. This provides an interpretable way to understand how the content contributes to similarity—frequent but meaningful words have more impact.
* **Input-Output Traceability**: The system clearly shows the input article and lists the recommended titles, making it easy for users to see what article was selected and why the similar ones were suggested—supporting intuitive human interpretation.
* **No Black-Box Behavior**: Unlike deep learning models, this approach avoids "black-box" complexity. Every step—from vectorization to similarity scoring—is explainable and can be broken down, making it highly suitable for educational or business use where trust and clarity matter.

***GITHUB LINK:***

<https://github.com/Simson2006/Round2>