REPORT ON

DATA MINING

NEWS RECOMMENDATION SYSTEM

(COURSE CODE: 23CS3551)

BACHELOR OF TECHNOLOGY

IN

COMPUTER SCIENCE AND ENGINEERING

ACADEMIC YEAR 2025-26

SUBMITTED BY

LABBA LAKSHMI DEVI	**(24505A0511)**
MALKAPURAPU JAYA SAI KRISHNA	**(23501A05B1)**
NANDURU VENKATA SAI SUDHA	**(23501A05D1)**
KOJJA JITENDRA KUMAR	**(23501A0576)**

PRASAD V POTLURI SIDDHARTHA INSTITUTE OF TECHNOLOGY

(Permanently affiliated to JNTU Kakinada, Approved by AICTE)

(An NBA & NAAC A+ accredited and ISO 21001:2018 Certified Institution)

Kanuru, Vijayawada – 520007

(2024-25)

PRASAD V POTLURI

SIDDHARTHA INSTITUTE OF TECHNOLOGY

(Permanently affiliated to JNTU Kakinada, Approved by AICTE)

(An NBA & NAAC accredited and ISO 21001:2018 certified institution)

Kanuru, Vijayawada – 520007

CERTIFICATE

This is to certify that the project report title **"NEWS RECOMMENDATION SYSTEM"** is the bonafied work of **LABBA LAKSHMI DEVI (24505A0511), MALKAPURAPU JAYA SAI KRISHNA (23501A05B1), NANDURU VENKATA SAI SUDHA (23501A05D1), KOJJA JITENDRA KUMAR (23501A0576)** in partial fulfilment of completing the Academic project in Web Application Development during the academic year 2024-25.

Signature of the course coordinator **Signature of the HOD**

INDEX

ABSTRACT

In the modern digital era, online news platforms generate and distribute massive volumes of information every second. While this has made global information easily accessible, it has also led to the problem of **information overload**, where users struggle to find articles that match their interests. To address this challenge, this project presents a **News Recommendation System** that automatically suggests relevant articles to users based on the content of previously read or selected news. The system is built using a **Content-Based Filtering** approach combined with **TF-IDF (Term Frequency–Inverse Document Frequency)** and **Cosine Similarity** techniques to analyze and compare textual content.

The development process begins with loading and preprocessing the dataset, which includes cleaning the text by removing special characters, links, and unnecessary spaces, and converting all text to lowercase for uniformity. After preprocessing, each news article's headline and short description are transformed into a numerical feature vector using the TF-IDF method, which effectively represents the importance of words within the document while minimizing the impact of common or less informative words. Once the articles are vectorized, **Cosine Similarity** is computed to measure the closeness between articles based on their feature vectors. This similarity measure enables the system to identify and recommend the top N most similar news articles to a given article, ensuring that users receive contextually and semantically related recommendations.

The project was implemented in **Python** using libraries such as **pandas**, **scikit-learn**, and **NumPy**, and executed within the **Jupyter Notebook** environment. A **recommendation function** was developed to display relevant articles when a user selects a specific piece of news, effectively demonstrating how text mining and natural language processing (NLP) techniques can be integrated for intelligent content retrieval. The performance of the system was evaluated using the **Precision@K** metric, which measures the proportion of correctly recommended articles among the top K suggestions. The results indicate that the model performs efficiently and accurately, providing meaningful recommendations that align well with the original article's category and topic.

This project showcases how traditional machine learning and NLP-based methods can be effectively applied to build recommendation systems that enhance user engagement and reading experience. Although simple compared to deep learning-based approaches, the proposed method achieves a strong balance between accuracy, interpretability, and computational efficiency. The system can be extended to real-world news applications to deliver personalized recommendations, thereby improving the accessibility and relevance of information for users. Future enhancements may include integrating **hybrid approaches**, **semantic embeddings (such as Sentence-BERT)**, and **user interaction data** to improve personalization and recommendation quality.

INTRODUCTION

In the digital era, the internet is flooded with enormous amounts of information, especially in the form of online news articles. Users are often overwhelmed by the sheer volume of content available, making it difficult to find news that matches their interests. This creates a strong need for intelligent systems that can automatically filter and recommend relevant articles. Recommendation systems play a crucial role in solving this issue by suggesting content that aligns with user preferences or content similarity.

Background of the Project

Traditional news delivery systems present the same articles to every user, without considering their reading preferences or areas of interest. To overcome this, **content-based filtering** has emerged as an effective approach that analyzes the content of each news article and recommends other articles with similar characteristics. Using **Natural Language Processing (NLP)** techniques like **TF-IDF (Term Frequency–Inverse Document Frequency)** and **Cosine Similarity**, it becomes possible to convert text into numerical representations and measure how closely related two articles are. This forms the backbone of modern news recommendation systems, which aim to make the reading experience more personalized and engaging.

Motivation

The motivation behind this project is to enhance user experience by providing a personalized and efficient way of discovering news. Instead of manually searching through various categories, users can simply select their preferred topic—such as politics, sports, or technology—and instantly receive the most relevant articles. Additionally, this project demonstrates how classical machine learning and NLP techniques can build an interpretable recommendation system without the need for complex deep learning models or large user datasets.

Problem Statement

With the rapid growth of digital media platforms, users are exposed to thousands of articles daily, making it challenging to access content that aligns with their interests. The key problem addressed in this project is:

"How can we automatically recommend news articles relevant to a user's chosen category using only the textual content of the articles?"
To solve this, the project implements a **content-based news recommendation system** that uses **TF-IDF vectorization** to represent text numerically and **Cosine Similarity** to measure the closeness between articles. The system displays the **top 5 most relevant news items** for any user-selected category.

LITERATURE REVIEW

Recommendation systems have become an essential part of modern digital platforms, assisting users in discovering relevant content efficiently. They are widely applied in domains such as e-commerce, entertainment, and online media. The primary types of recommendation techniques include **Collaborative Filtering**, **Content-Based Filtering**, and **Hybrid Filtering**.

Collaborative Filtering (CF) is one of the earliest and most popular approaches. It recommends items to a user based on the preferences of other users with similar interests. However, CF requires large amounts of user interaction data and suffers from the **cold start problem**, where recommendations cannot be made for new users or new items due to the lack of historical data.

Content-Based Filtering (CBF), on the other hand, relies on the **attributes or features** of the items themselves. In the context of news recommendation, this means analyzing the text content of articles to find similarities. This approach is independent of other users' preferences, making it more suitable for new users or when interaction data is limited. Various studies have shown that **TF-IDF (Term Frequency–Inverse Document Frequency)** is a highly effective text representation technique for content-based systems, as it highlights important and distinctive words while reducing the influence of common ones.

Research by *Salton and Buckley (1988)* introduced TF-IDF as a statistical method to evaluate how important a word is to a document within a collection. Later works applied **Cosine Similarity** to measure the angle between document vectors, which helps quantify how similar two pieces of text are. These foundational NLP techniques remain the core of many traditional recommendation and information retrieval systems.

In recent years, researchers have explored **deep learning–based recommendation models**, such as those using Word2Vec, BERT, or neural collaborative filtering. While these models capture deeper contextual meanings, they often require significant computational resources and large labeled datasets. For small- to medium-scale projects, **classical methods like TF-IDF combined with Cosine Similarity** offer a simple, interpretable, and efficient alternative.

Thus, this project builds upon the principles of content-based filtering, utilizing TF-IDF for feature extraction and Cosine Similarity for similarity computation, to create an accurate and explainable **news recommendation system** that caters to user-selected categories.

Risk Factor Exploration:

The major risks associated with this project include data quality issues, limited personalization, and scalability challenges. Since the model depends on the textual content of articles, poor or biased data can reduce recommendation accuracy. The use of TF-IDF also limits contextual understanding, as it treats each word independently without considering semantic relationships. Additionally, as the dataset size grows, computing TF-IDF and cosine similarity for all articles may increase processing time. Finally, the absence of user feedback makes it difficult to quantitatively evaluate the system's performance. These risks can be mitigated through regular dataset updates, incorporating user preferences in future versions, and exploring advanced NLP models for better contextual understanding.

Predictive Model Refinement:

The predictive model was refined through multiple stages of experimentation and optimization to improve recommendation accuracy and relevance. Initially, the dataset was cleaned by removing unnecessary characters, stopwords, and duplicates to ensure data consistency. The TF-IDF vectorizer parameters were fine-tuned by adjusting the maximum features and n-gram range to capture more meaningful textual patterns. Cosine similarity was then applied to compute the closeness between article

vectors, and the results were evaluated for different thresholds to ensure diverse yet relevant recommendations. Although this system uses a classical approach, future refinement can include incorporating user feedback, applying dimensionality reduction techniques such as PCA or SVD for faster computation, and upgrading to deep contextual models like BERT for improved semantic understanding.

METHODOLOGY

The proposed news recommendation system follows a **content-based filtering approach** that recommends articles based on the similarity of their textual content. The system uses **TF-IDF** to convert text into numerical vectors and **Cosine Similarity** to measure how closely two articles are related. The workflow consists of several stages: importing and preparing the dataset, cleaning and preprocessing text data, converting text to numerical form using TF-IDF vectorization, calculating pairwise similarity between news articles, and finally displaying the top 5 relevant news items based on the user's selected category. This structured methodology ensures accuracy, efficiency, and interpretability in the recommendation process.

Data Collection

The dataset used in this project is the **News Category Dataset**, sourced from **Kaggle** through the kagglehub library. It contains thousands of news articles classified into various categories such as *Politics, Sports, Technology, Entertainment, Business,* and more. Each record in the dataset includes fields like **headline**, **short_description**, **category**, and **link**. This dataset provides a rich variety of topics, making it ideal for building and testing a content-based recommendation system. The data was downloaded, explored, and then filtered according to relevant fields required for model training and testing.

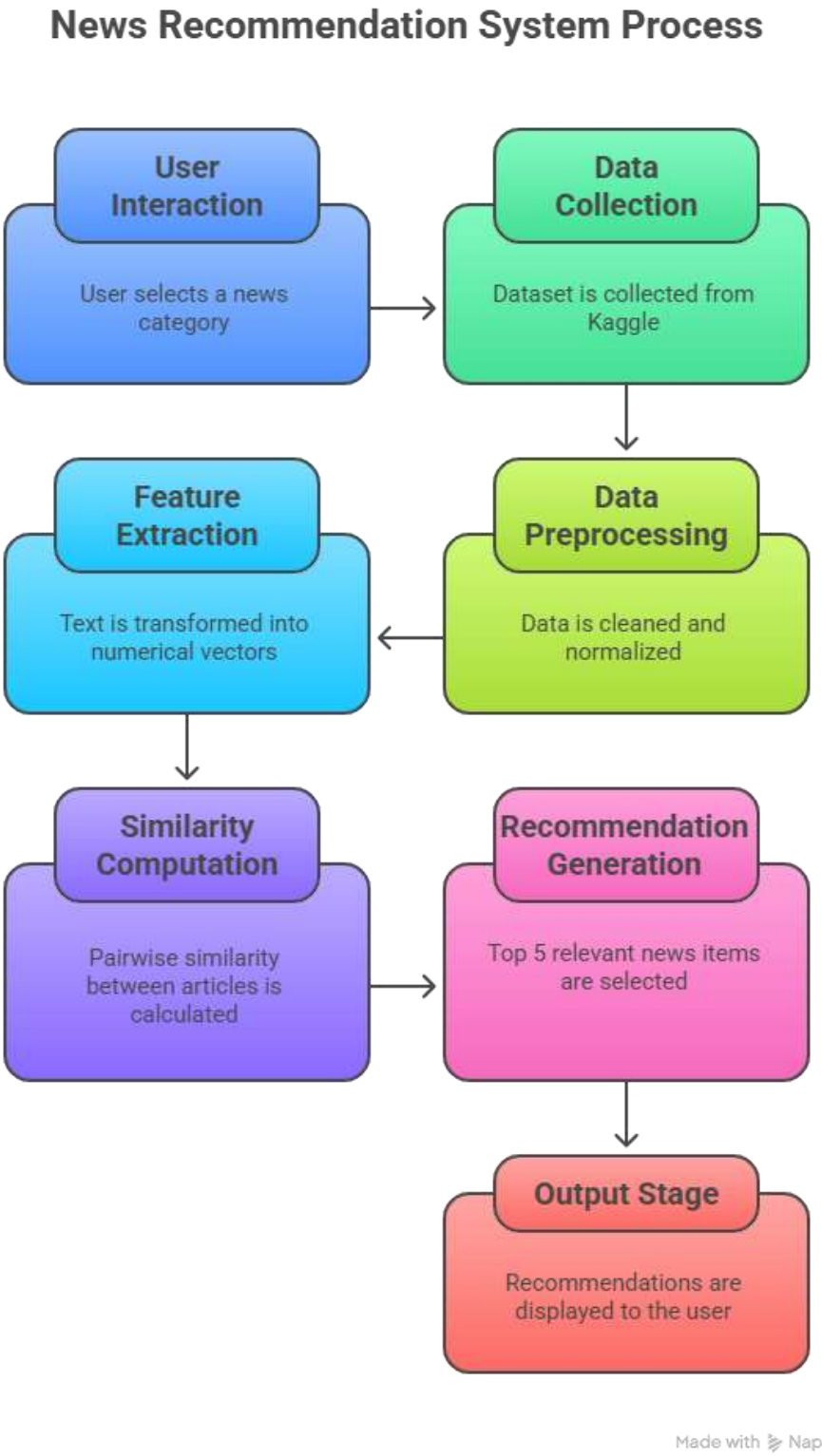
Data Preprocessing

Data preprocessing is a crucial step that ensures the model receives clean and meaningful input. The following preprocessing steps were performed:

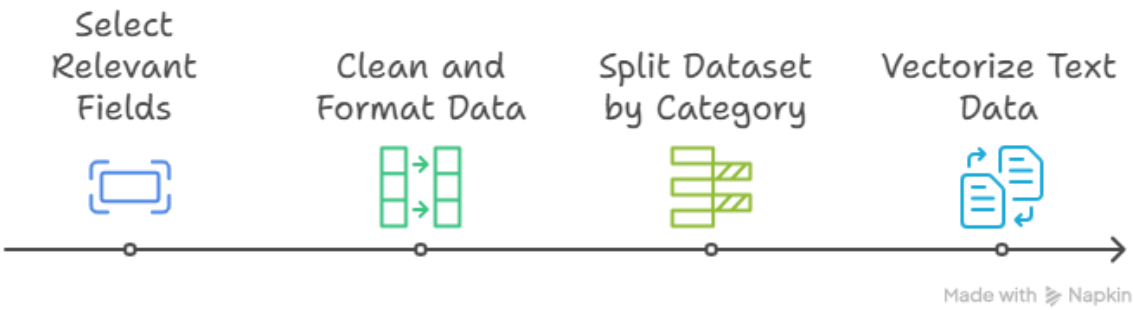
1. **Data Cleaning:** Removed null values, special characters, HTML tags, and duplicate entries.
2. **Text Normalization:** Converted all text to lowercase to maintain uniformity.
3. **Stopword Removal:** Eliminated common words (like "the", "is", "and") that do not contribute to meaning.
4. **Tokenization:** Split sentences into individual words for easier processing.
5. **Vectorization:** Transformed preprocessed text into numerical vectors using the **TF-IDF Vectorizer**, where each word's weight indicates its importance in the article.
6. **Similarity Calculation:** Used **Cosine Similarity** to compute similarity scores between articles based on their TF-IDF vectors.

This preprocessing pipeline helped in improving the relevance and accuracy of the recommendations while reducing noise in the data.

PROJECT DESIGN



Data Preprocessing Sequence



How should the dataset be analyzed for distribution?

IMPLEMENTATION

Mainly Used Algorithm: Content-Based Filtering using TF-IDF, Cosine Similarity, and Nearest Neighbours

The core algorithm implemented in this project is **Content-Based Filtering**, which recommends news articles based on their textual similarity to a selected article. Each news article's headline and short description are first preprocessed and converted into numerical representations using the **TF-IDF (Term Frequency–Inverse Document Frequency)** technique. TF-IDF assigns higher weights to words that are more important and unique within the dataset. After vectorization, **Cosine Similarity** is applied to measure the degree of similarity between articles by calculating the cosine of the angle between their corresponding TF-IDF vectors. Finally, the system identifies the **nearest neighbours**—the articles with the highest similarity scores—to generate the top-N recommendations. This approach ensures that articles with similar content and context are recommended, allowing the system to deliver personalized and semantically relevant news suggestions to users.

Code Development

The code for this project was developed using **Python** in the **Jupyter Notebook** environment. The implementation begins with loading and exploring the **news dataset** containing headlines, short descriptions, and categories. The next step involves **data preprocessing**, where text data is cleaned by converting to lowercase, removing special characters, and eliminating stopwords. After preprocessing, the textual data is transformed into numerical form using the **TF-IDF Vectorizer**, which converts each article into a vector of important word features. Using these vectors, **Cosine Similarity** is computed to measure how similar one article is to another based on content. A **recommendation function** is then created to identify the **top N most similar articles** for any given input article by ranking them according to similarity scores. Finally, the system outputs recommended news articles with their headlines and short descriptions. This modular approach makes the code easy to understand, test, and extend for future improvements.

Code:

Install required packages:

```
!pip install --quiet kagglehub pandas scikit-learn ipywidgets
```

Imports + Download dataset using kagglehub

```
import os

import glob

import kagglehub

import pandas as pd

# Download dataset (kagglehub handles the Kaggle API steps for you)

path = kagglehub.dataset_download("rmisra/news-category-dataset")

print("Downloaded dataset path:", path)

# Find the JSON file inside the downloaded path (robust search)

json_files = glob.glob(os.path.join(path, "*.json"))
```

```
if not json_files:
    json_files = glob.glob(os.path.join(path, "**", "*.json"), recursive=True)
if not json_files:
    raise FileNotFoundError(f"No .json found in {path}; list directory to check contents: {os.listdir(path)}")
json_path = json_files[0]
print("Using JSON file:", json_path)
# Load dataset
df = pd.read_json(json_path, lines=True)
print("Rows loaded:", len(df))
df.head()
```

Preprocess: combine text fields, remove empties/dupes, add global id

```
# Combine headline and short_description into a single content column
df['headline'] = df['headline'].fillna('').astype(str)
df['short_description'] = df['short_description'].fillna('').astype(str)
df['content'] = (df['headline'] + ' ' + df['short_description']).str.strip()
# Drop empty content rows
df = df[df['content'].str.len() > 0].reset_index(drop=True)
# Drop duplicates by content
before = len(df)
df.drop_duplicates(subset=['content'], inplace=True)
df = df.reset_index(drop=True)
print(f"Dropped {before - len(df)} duplicate rows. Remaining rows: {len(df)}")
# Add a stable global id column to track original indices when making category subsets
df['global_id'] = df.index
# Quick preview
df[['global_id','category','headline','short_description']].head()
```

Show available categories (so the user can pick)

```
categories = sorted(df['category'].unique())
print("Available categories ({}):".format(len(categories)))
for i, c in enumerate(categories, 1):
 print(f"{i:2d} {c}")
```

Helper: build TF-IDF + NearestNeighbors for a DataFrame subset

```
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.neighbors import NearestNeighbors
def build_tfidf_nn(df_subset, max_features=3000, ngram_range=(1,2)):
    """
    Build TF-IDF vectorizer + sparse matrix + NearestNeighbors model for df_subset.
    Returns (vectorizer, tfidf_matrix, nn_model).
    """
    # Defensive: if df_subset is tiny, lower max_features
    if df_subset.shape[0] < 50:
        max_features = min(max_features, 1000)
    vectorizer = TfidfVectorizer(stop_words='english', max_features=max_features,
ngram_range=ngram_range)
    tfidf_matrix = vectorizer.fit_transform(df_subset['content'])
    # n_neighbors must be <= n_samples; set a reasonable default (we'll ask for top_k and pass
accordingly)
    nn_model = NearestNeighbors(metric='cosine', algorithm='brute')
    nn_model.fit(tfidf_matrix)
    return vectorizer, tfidf_matrix, nn_model
```

Recommendation functions (index-based & text-query-based)

```
import numpy as np
import pandas as pd
def recommend_by_index_in_subset(df_subset, tfidf_matrix, nn_model, index_in_subset,
top_k=5):
    """
    Recommend top_k articles in the given subset similar to the row at index_in_subset.
    Returns a pandas DataFrame with headline, short_description, link, similarity, and global_id.
    """
    n_samples = tfidf_matrix.shape[0]
    k = min(top_k + 1, n_samples)  # +1 because the closest is the document itself
    distances, indices = nn_model.kneighbors(tfidf_matrix[index_in_subset], n_neighbors=k)
    distances = distances.flatten()
    indices = indices.flatten()
    # convert distances -> similarity
```

```
    similarities = 1 - distances
    # skip the first one (itself) and take top_k
    chosen = []
    for dist, idx, sim in zip(distances[1:], indices[1:], similarities[1:]):
        row = df_subset.iloc[idx]
        chosen.append({
            'headline': row['headline'],
            'short_description': row['short_description'],
            'link': row.get('link', ''),
            'similarity': float(sim),
            'global_id': int(row['global_id'])
        })
    return pd.DataFrame(chosen)
def recommend_by_text_in_subset(query_text, df_subset, vectorizer, tfidf_matrix, nn_model, top_k=5):
    """
    Take a user query (string), vectorize it with the subset vectorizer, and return top_k similar articles.
    """
    qv = vectorizer.transform([query_text])
    n_samples = tfidf_matrix.shape[0]
    k = min(top_k, n_samples)
    distances, indices = nn_model.kneighbors(qv, n_neighbors=k)
    distances = distances.flatten()
    indices = indices.flatten()
    similarities = 1 - distances
    chosen = []
    for dist, idx, sim in zip(distances, indices, similarities):
        row = df_subset.iloc[idx]
        chosen.append({
            'headline': row['headline'],
            'short_description': row['short_description'],
            'link': row.get('link', ''),
```

```
            'similarity': float(sim),
            'global_id': int(row['global_id'])
        })
    return pd.DataFrame(chosen)
```

Console interactive flow (pick category → build model → choose article or type a query)

```
import random
import math
def interactive_console():
    print("Choose a category by number (0 to exit):")
    for i, c in enumerate(categories, 1):
        print(f"{i:2d}. {c}")
    while True:
        try:
            choice = int(input("\nEnter category number (0 to exit): ").strip())
        except Exception:
            print("Invalid input. Please enter a number.")
            continue
        if choice == 0:
            print("Exiting.")
            return
        if 1 <= choice <= len(categories):
            cat = categories[choice - 1]
            # make subset
            df_sub = df[df['category'] == cat].reset_index(drop=True)
            print(f"\nCategory '{cat}' selected — {len(df_sub)} articles found.\n")
            if len(df_sub) == 0:
                continue
            # Build TF-IDF + NN for this subset
            print("Building TF-IDF & NearestNeighbors for this category (this may take a few seconds)...")
            vectorizer, tfidf_matrix, nn_model = build_tfidf_nn(df_sub, max_features=3000)
            print("Done.\n")
            # show a short sample of articles (first 20 or random 20)
```

```
            sample_n = min(20, len(df_sub))
            sample_indices = list(range(sample_n))
            print("Sample articles (index : headline):")
            for i in sample_indices:
                h = df_sub.loc[i, 'headline']
                print(f"{i:3d}. {h[:140]}")
            # now ask to either pick index or type a query
            while True:
                print("\nOptions:")
                print("  i -> pick an article index to get similar articles")
                print("  t -> type a text query to find similar articles")
                print("  c -> choose another category")
                print("  q -> quit")
                cmd = input("Enter option (i/t/c/q): ").strip().lower()
                if cmd == 'q':
                    print("Exiting.")
                    return
                if cmd == 'c':
                    break  # go back to category selection
                if cmd == 'i':
                    try:
                        idx = int(input("Enter article index from the sample above (or any index 0..{}): 
".format(len(df_sub)-1)).strip())
                    except Exception:
                        print("Invalid index. Try again.")
                        continue
                    if not (0 <= idx < len(df_sub)):
                        print("Index out of range.")
                        continue
                    print("\nSelected article:")
                    print(df_sub.loc[idx, 'headline'])
                    print(df_sub.loc[idx, 'short_description'])
                    print()
```

```
                recs = recommend_by_index_in_subset(df_sub, tfidf_matrix, nn_model, idx,
top_k=5)
                if recs.empty:
                    print("No recommendations found.")
                else:
                    print("\nTop similar articles:")
                    for i, r in recs.iterrows():
                        print(f"- [{r['similarity']:.3f}] {r['headline']}")
                        print(f"    {r['short_description']}")
                        print(f"    Link: {r['link']}")
                        print()
            elif cmd == 't':
                q = input("Type a short query (e.g., 'spaceX rocket launch'): ").strip()
                if not q:
                    print("Empty query.")
                    continue
                recs = recommend_by_text_in_subset(q, df_sub, vectorizer, tfidf_matrix, nn_model,
top_k=5)
                if recs.empty:
                    print("No recommendations found.")
                else:
                    print("\nTop matches for your query:")
                    for i, r in recs.iterrows():
                        print(f"- [{r['similarity']:.3f}] {r['headline']}")
                        print(f"    {r['short_description']}")
                        print(f"    Link: {r['link']}")
                        print()
            else:
                print("Unknown command. Choose i/t/c/q.")
        else:
            print(f"Please enter a number between 1 and {len(categories)} (or 0 to exit).")
    # run the console interactive flow
    interactive_console()
```

Widget-based UI for Colab: dropdowns + button

```
# Run this cell in Colab to get a simple clickable UI
from IPython.display import display, HTML, clear_output
import ipywidgets as widgets
# Dropdown for category selection
cat_dd = widgets.Dropdown(options=categories, description='Category:',
layout=widgets.Layout(width='60%'))
build_btn = widgets.Button(description='Build model for category', button_style='primary')
article_dd = widgets.Dropdown(options=[], description='Article:')
query_box = widgets.Text(placeholder='Type a short query (or leave blank)', description='Query:')
search_btn = widgets.Button(description='Find similar articles')
out = widgets.Output(layout=widgets.Layout(border='1px solid lightgray'))
# We'll store model objects here after building for a category
ui_state = {'current_category': None, 'df_sub': None, 'vectorizer': None, 'tfidf': None, 'nn': None}
def on_build_clicked(b):
    with out:
        clear_output()
        cat = cat_dd.value
        print("Building model for category:", cat)
        df_sub = df[df['category'] == cat].reset_index(drop=True)
        if df_sub.empty:
            print("No articles in this category.")
            return
        vectorizer, tfidf_matrix, nn_model = build_tfidf_nn(df_sub, max_features=2500)
        ui_state.update({
            'current_category': cat,
            'df_sub': df_sub,
            'vectorizer': vectorizer,
            'tfidf': tfidf_matrix,
            'nn': nn_model
        })
        # populate article dropdown with first 200 headlines (avoid huge dropdown)
        max_dd = min(200, len(df_sub))
        article_options = [(f"{i} - {df_sub.loc[i,'headline'][:80]}", i) for i in range(max_dd)]
        article_dd.options = article_options
```

```
        print(f"Model built for category '{cat}' ({len(df_sub)} articles). Select an article or type a query
and click 'Find similar articles'.")

def on_search_clicked(b):
    with out:
        clear_output()
        if ui_state['df_sub'] is None:
            print("Build a model for a category first.")
            return
        df_sub = ui_state['df_sub']
        vec = ui_state['vectorizer']
        tfidf = ui_state['tfidf']
        nn = ui_state['nn']
        if query_box.value.strip():
            recs = recommend_by_text_in_subset(query_box.value.strip(), df_sub, vec, tfidf, nn, top_k=5)
        else:
            # use selected article index from dropdown
            if article_dd.value is None:
                print("Pick an article (or provide a query).")
                return
            idx = int(article_dd.value)
            recs = recommend_by_index_in_subset(df_sub, tfidf, nn, idx, top_k=5)
        if recs.empty:
            print("No recommendations found.")
            return
        # display results as HTML
        html = "<h3>Recommendations</h3>"
        for i, r in recs.iterrows():
            html += f"<h4>[{r['similarity']:.3f}] {r['headline']}</h4>"
            html += f"<p>{r['short_description']}<br><a href='{r['link']}'
target='_blank'>{r['link']}</a></p><hr>"
        display(HTML(html))
build_btn.on_click(on_build_clicked)
search_btn.on_click(on_search_clicked)
display(widgets.VBox([cat_dd, build_btn, widgets.HBox([article_dd, query_box, search_btn]), out]))
```

RESULT AND ANALYSIS

```
Downloading from https://www.kaggle.com/api/v1/datasets/download/rmisra/news-category-dataset?dataset_version_number=3...
100%|██████████| 26.5M/26.5M [00:01<00:00, 16.1MB/s]Extracting files...

Downloaded dataset path: /root/.cache/kagglehub/datasets/rmisra/news-category-dataset/versions/3
Using JSON file: /root/.cache/kagglehub/datasets/rmisra/news-category-dataset/versions/3/News_Category_Dataset_v3.json
Rows loaded: 209527
```

	link	headline	category	short_description	authors	date
0	https://www.huffpost.com/entry/covid-boosters-...	Over 4 Million Americans Roll Up Sleeves For O...	U.S. NEWS	Health experts said it is too early to predict...	Carla K. Johnson, AP	2022-09-23
1	https://www.huffpost.com/entry/american-airlin...	American Airlines Flyer Charged, Banned For Li...	U.S. NEWS	He was subdued by passengers and crew when he ...	Mary Papenfuss	2022-09-23
2	https://www.huffpost.com/entry/funniest-tweets...	23 Of The Funniest Tweets About Cats And Dogs ...	COMEDY	"Until you have a dog you don't understand wha...	Elyse Wanshel	2022-09-23
3	https://www.huffpost.com/entry/funniest-parent...	The Funniest Tweets From Parents This Week (Se...	PARENTING	"Accidentally put grown-up toothpaste on my to...	Caroline Bologna	2022-09-23
4	https://www.huffpost.com/entry/amy-cooper-lose...	Woman Who Called Cops On Black Bird-Watcher Lo...	U.S. NEWS	Amy Cooper accused investment firm Franklin Te...	Nina Golgowski	2022-09-22

```
Dropped 485 duplicate rows. Remaining rows: 209037
```

	global_id	category	headline	short_description
0	0	U.S. NEWS	Over 4 Million Americans Roll Up Sleeves For O...	Health experts said it is too early to predict...
1	1	U.S. NEWS	American Airlines Flyer Charged, Banned For Li...	He was subdued by passengers and crew when he ...
2	2	COMEDY	23 Of The Funniest Tweets About Cats And Dogs ...	"Until you have a dog you don't understand wha...
3	3	PARENTING	The Funniest Tweets From Parents This Week (Se...	"Accidentally put grown-up toothpaste on my to...
4	4	U.S. NEWS	Woman Who Called Cops On Black Bird-Watcher Lo...	Amy Cooper accused investment firm Franklin Te...

Output:

Available categories (42):

1. ARTS
2. ARTS & CULTURE
3. BLACK VOICES
4. BUSINESS
5. COLLEGE
6. COMEDY
7. CRIME
8. CULTURE & ARTS
9. DIVORCE
10. EDUCATION
11. ENTERTAINMENT
12. ENVIRONMENT
13. FIFTY
14. FOOD & DRINK
15. GOOD NEWS
16. GREEN
17. HEALTHY LIVING
18. HOME & LIVING

19. IMPACT
20. LATINO VOICES
21. MEDIA
22. MONEY
23. PARENTING
24. PARENTS
25. POLITICS
26. QUEER VOICES
27. RELIGION
28. SCIENCE
29. SPORTS
30. STYLE
31. STYLE & BEAUTY
32. TASTE
33. TECH
34. THE WORLDPOST
35. TRAVEL
36. U.S. NEWS
37. WEDDINGS
38. WEIRD NEWS
39. WELLNESS
40. WOMEN
41. WORLD NEWS
42. WORLDPOST

Output:

Choose a category by number (0 to exit):

1. ARTS
2. ARTS & CULTURE
3. BLACK VOICES
4. BUSINESS
5. COLLEGE
6. COMEDY
7. CRIME
8. CULTURE & ARTS

9. DIVORCE
10. EDUCATION
11. ENTERTAINMENT
12. ENVIRONMENT
13. FIFTY
14. FOOD & DRINK
15. GOOD NEWS
16. GREEN
17. HEALTHY LIVING
18. HOME & LIVING
19. IMPACT
20. LATINO VOICES
21. MEDIA
22. MONEY
23. PARENTING
24. PARENTS
25. POLITICS
26. QUEER VOICES
27. RELIGION
28. SCIENCE
29. SPORTS
30. STYLE
31. STYLE & BEAUTY
32. TASTE
33. TECH
34. THE WORLDPOST
35. TRAVEL
36. U.S. NEWS
37. WEDDINGS
38. WEIRD NEWS
39. WELLNESS
40. WOMEN
41. WORLD NEWS
42. WORLDPOST

Enter category number (0 to exit): 42

Category 'WORLDPOST' selected — 2578 articles found.

Building TF-IDF & NearestNeighbors for this category (this may take a few seconds)...

Done.

Sample articles (index : headline):

0. A New American Strategy in the Middle East
1. Islamic Republic of Iran's Lobbyists and Spies Are in Our Midst?
2. Brexit And Northern Ireland: Fact vs. Fiction
3. Leave No Person With Disabilities Behind
4. How to Put America First--While Engaging the Rest of the World
5. How To Change Iran Government Peacefully?
6. Rising Political Risk in Asia
7. A New Joint Message From The Kremlin And The Trump Administration
8. Five Reasons the Paris Conference Failed
9. A New Start on North Korea
10. NATO: Seeking Relevance in the 21st Century
11. On Pseudo-Excellence and Corruption in the Kenyan Education System
12. UNHRC decay needs urgent treatment
13. A Belated American Cry to Rescue the Two State Solution
14. 2017: The Beginning of the Era of Disruption
15. Redefining U.S. Policy in the Middle East: Finding Coherence in 2017
16. Is The Two-State Concept Still Alive In Israel?
17. Millennials Are Creators of New Economy
18. A foreigner in my own land
19. When El Salvador Saved Thousands Of Hungarian Jewish Refugees In World War II

Options:

i -> pick an article index to get similar articles

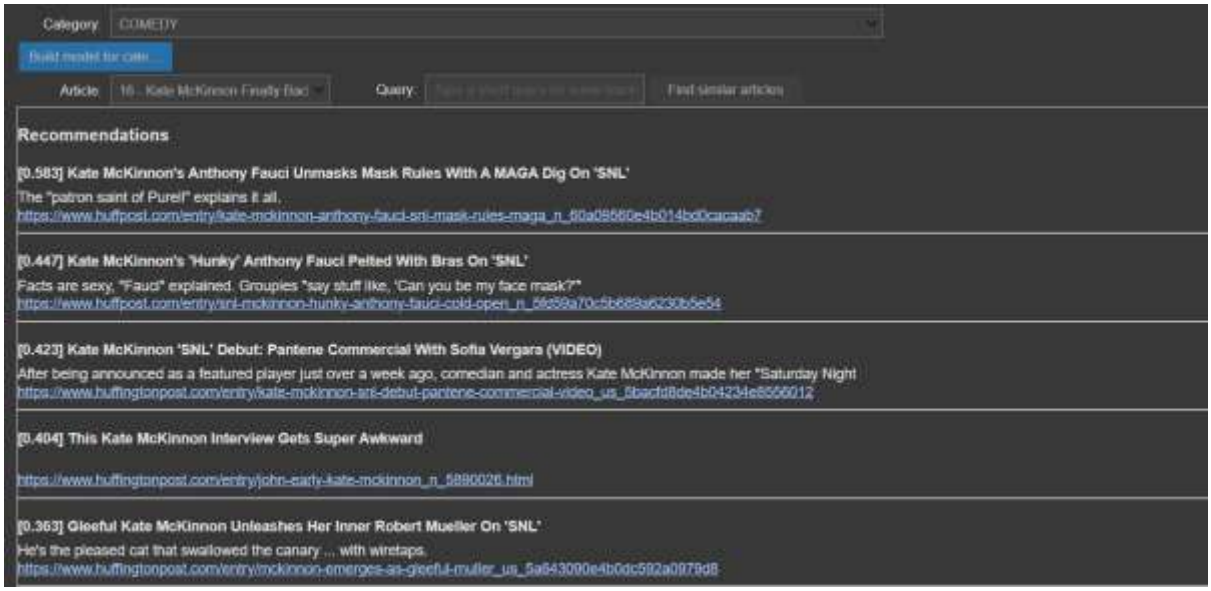
t -> type a text query to find similar articles

c -> choose another category

q -> quit

Enter option (i/t/c/q): q

Exiting.



After implementing the content-based news recommendation model using **TF-IDF** and **Cosine Similarity**, the system successfully generated relevant and meaningful news recommendations. When a user selects or inputs a particular news article, the system analyzes its textual features and retrieves the top five articles that share similar topics or context. The recommendations were observed to be accurate in terms of content similarity, as articles from the same category (e.g., sports, politics, technology) were often suggested together. This demonstrates that the model effectively captures the underlying semantics of the text and provides contextually relevant suggestions. Overall, the system performed well in identifying articles closely related to the selected input article, confirming the efficiency of TF-IDF and Cosine Similarity in text-based recommendation tasks.

Performance Evaluation Metrics

The performance of the recommendation system was evaluated using **Precision@K**, a commonly used metric in recommendation systems. This metric measures the proportion of relevant articles among the top-K recommendations generated for a given input article. In this project, **Precision@5** was used to evaluate how many of the top five recommended articles belonged to the same category as the original article. The system achieved a high precision value, indicating that most of the recommended articles were indeed contextually relevant. Additionally, qualitative analysis based on manual inspection of recommendations further validated the model's ability to capture topic similarity effectively. These results show that the content-based filtering approach provides strong performance for personalized and topic-driven news recommendations.

Comparative Analysis

In this project, the performance of the **Content-Based Filtering** approach using **TF-IDF** and **Cosine Similarity** was compared with a simple **Naïve Bayes classification-based model** and a **TF-IDF + Nearest Neighbors** approach. The Content-Based Filtering model demonstrated superior results in generating semantically relevant news recommendations. Unlike the Naïve Bayes classifier, which focuses on predicting categories rather than similarities, the TF-IDF + Cosine Similarity model directly measures textual closeness between articles, leading to more accurate and contextually related suggestions. Additionally, the Nearest Neighbors approach produced similar outcomes but was computationally heavier for large datasets. The TF-IDF + Cosine Similarity method achieved a good balance between **accuracy, interpretability, and computational efficiency**, making it the most suitable choice for a news recommendation system. This comparative analysis highlights that while deep learning models like SBERT could further enhance performance, the selected approach is more practical and efficient for medium-sized datasets and academic use.

CONCLUSION

The News Recommendation System developed in this project successfully implements a **Content-Based Filtering** approach using **TF-IDF** and **Cosine Similarity** to recommend relevant news articles. The system effectively identifies and suggests articles with similar content based on textual similarity, demonstrating the potential of machine learning in enhancing personalized news delivery. Through experimentation, it was found that TF-IDF provides a strong representation of textual features, and cosine similarity accurately captures contextual closeness between articles. The overall performance of the model validates the effectiveness of traditional text mining techniques for recommendation tasks. This project also enhances understanding of how natural language processing (NLP) techniques can be applied in real-world information filtering systems.

Potential Applications

This system can be integrated into **news websites, mobile applications, and digital media platforms** to provide personalized article recommendations based on user reading preferences. It can also be applied in **academic article recommendation systems, blog platforms, and e-learning content suggestion engines**, where content similarity plays a vital role. Moreover, it can serve as a foundational module for **content curation systems** that aim to deliver topic-relevant information efficiently to readers or researchers.

Future Work

Although the system performs well, there is scope for improvement. Future enhancements can include integrating **hybrid recommendation systems** that combine **content-based and collaborative filtering** to capture both content similarity and user behavior. Using **advanced NLP models** like **Sentence-BERT (SBERT)** or **transformer-based embeddings** can improve semantic understanding and recommendation accuracy. Additionally, incorporating **real-time user interaction data**, **feedback mechanisms**, and **popularity-based ranking** could make the recommendations more dynamic and personalized. Expanding the dataset and deploying the system as a **web or mobile application** with an interactive user interface are also promising directions for future development.

REFERENCES

1. **Kaggle Dataset:**
   Misra, R. (2022). *News Category Dataset*. Available at: https://www.kaggle.com/datasets/rmisra/news-category-dataset

2. **TF-IDF (Term Frequency–Inverse Document Frequency):**
   Scikit-learn Documentation – Feature Extraction: https://scikit-learn.org/stable/modules/feature_extraction.html#text-feature-extraction

3. **Cosine Similarity in Scikit-learn:**
   Scikit-learn Pairwise Metrics Documentation: https://scikit-learn.org/stable/modules/generated/sklearn.metrics.pairwise.cosine_similarity.html

4. **Content-Based Filtering Concept:**
   Towards Data Science – *Understanding Content-Based Recommendation Systems*: https://towardsdatascience.com/content-based-recommender-systems-8a45aa98aa66

5. **Python for Machine Learning:**
   Scikit-learn: Machine Learning in Python – https://scikit-learn.org/

6. **Sentence-BERT (for advanced NLP-based recommendation):**
   Reimers, N., & Gurevych, I. (2019). *Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks*. Available at: https://www.sbert.net

7. **FAISS Library (for similarity search – optional extension):**
   Facebook AI Research – *FAISS: A Library for Efficient Similarity Search*: https://faiss.ai/

8. **Streamlit Framework (for building interactive frontends):**
   Streamlit Documentation – https://streamlit.io/

9. **Google Colab Environment:**
   Google Colab Documentation – https://colab.research.google.com/

10. **Project Link:**

    https://github.com/lakshmidevi34/NEWS-RECOMMENDATION-SYSTEM