Article Recommendation System

Project submitted to the

SRM University – AP, Andhra Pradesh

for the partial fulfillment of the requirements to award the degree of

Bachelor of Technology

In

Computer Science and Engineering

School of Engineering and Sciences

Submitted by **Group 15**

LOKESH DASARI	**AP21110011352**
ANANTHA TEJA DASARI	**AP21110011369**
RAKESH TIRUMALAPARAPU	**AP21110011374**
SANJEEV SURAM	**AP21110011392**

Under the Guidance of

Dr. Rajiv Senapati

SRM University–AP

Neerukonda, Mangalagiri, Guntur

Andhra Pradesh – 522 240

Certificate

Date: 30-Apr-24

This is to certify that the work present in this Project entitled "**Article Recommendation System**" has been carried out **by Lokesh Dasari, Anantha Teja Dasari, Rakesh Tirumalaparapu, Sanjeev Suram** under my supervision. The work is genuine, original, and suitable for submission to the SRM University – AP for the award of Bachelor of Technology in the **School of Engineering and Sciences**.

Supervisor

(Signature)

Dr. Rajiv Senapati

Designation,

Affiliation.

Acknowledgments

I extend my sincere appreciation and heartfelt thanks to **Dr. Rajiv Senapati**, my mentor, for her outstanding guidance, continuous monitoring, and unwavering encouragement throughout the duration of our project, titled "Article Recommendation System".

Dr. Rajiv Senapati expertise and mentorship have been instrumental in our successful completion of this project. Her support has been a beacon, guiding me through various stages and significantly contributing to my academic and professional journey. I am grateful for her cordial assistance, valuable insights, and direction, which played a pivotal role in achieving the milestones of this undertaking.

I am dedicated to maintaining the same level of commitment in providing insightful responses that cater to your specific needs and contribute to a deeper understanding of emerging technologies. Please do not hesitate to contact me for any further inquiries or requests.

Abstract

This project presents a model for classifying and recommending articles based on their titles. The model utilizes cosine similarity to measure the similarity between article titles, allowing it to recommend the top 5 most related articles to a given title. The dataset used for training the model includes crime articles and general articles, providing a diverse set of articles for recommendation.

The implementation includes a user-friendly interface developed using Flask, a Python web framework. This interface allows users to input article titles and receive personalized recommendations, complete with direct links to the recommended articles. This seamless integration of recommendation functionality into the user interface enhances the overall user experience and facilitates easy access to relevant content.

Furthermore, the project showcases the flexibility and scalability of cosine similarity in content recommendation systems. It demonstrates the potential for this approach to be applied in various domains beyond article recommendation, highlighting its adaptability and effectiveness in personalized content delivery.

1. Introduction

In the digital age, accessing relevant and engaging content is paramount. However, with the vast amount of information available online, finding articles that align with one's interests can be challenging. This project addresses this challenge by presenting a single platform for article classification and recommendation, leveraging advanced techniques to provide users with personalized recommendations.

The primary objective of this project is to create a platform where users can input an article title and receive recommendations for similar articles. This platform aims to streamline the process of content discovery, offering users a curated selection of articles based on their interests. By utilizing cosine similarity, the model can accurately assess the similarity between article titles and recommend the topmost related articles to the user. Imagine typing in the title of an article you liked, and instantly getting a list of similar articles to explore. It's like having a personal assistant for discovering new content! This approach ensures that users receive relevant and engaging content tailored to their interests.

Problem Statement:

The problem statement revolves around the difficulty users face in discovering articles that are relevant to their interests. With the abundance of online content, users often struggle to find articles that align with their preferences. This project seeks to solve this problem by creating a platform that utilizes advanced techniques to recommend articles similar to a given title. By doing so, users can easily discover and explore articles that match their interests, enhancing their overall browsing experience.

Proposed Solution:

To address the problem statement, this project proposes the development of a model that utilizes cosine similarity to recommend articles based on their titles. The model will be trained on a dataset comprising crime articles and general articles, ensuring a diverse range of content for recommendation.

To achieve this, the project utilizes a dataset comprising crime articles and general articles, ensuring a diverse range of content for recommendation. This dataset is processed and analyzed using Python, a versatile programming language, with the Flask web framework used to develop the user interface. The interface allows users to easily input an article title and receive personalized recommendations, complete with direct links to the recommended articles, enhancing the overall user experience.

By implementing this platform, users can discover and explore articles that align with their interests, thereby enhancing their browsing experience. Moreover, the project demonstrates the scalability and adaptability of cosine similarity in content recommendation systems, showcasing its potential for application in various domains beyond article recommendation.

Overall, this project aims to provide users with a comprehensive and personalized platform for article recommendation, offering a seamless and efficient way to discover relevant and engaging content in the vast landscape of online articles.

2. Literature Work

The exponential growth of digital content has made it increasingly challenging for users to discover relevant information. This challenge has led to the development of various techniques and approaches in the fields of information retrieval and recommendation systems.

One common approach is collaborative filtering, which recommends items based on the preferences of similar users. While effective, collaborative filtering relies heavily on user interaction data, which may not always be available or reliable.

Another approach is content-based filtering, which recommends items based on their attributes and the user's preferences. This approach is often used in conjunction with collaborative filtering to improve recommendation accuracy.

2.1 Reviewing Existing Approaches and Technologies in Article Recommendation Systems

Reviewing existing approaches and technologies used in article recommendation systems reveals a diverse landscape of methods employed to address various content recommendation challenges. These approaches leverage advancements in artificial intelligence (AI), machine learning (ML), natural language processing (NLP), and data analytics to provide personalized recommendations and decision support. Here's an overview of some key approaches and technologies:

For our website, we integrated an article recommendation system using Flask, leveraging cosine similarity to recommend articles from the dataset. This system utilizes AI, ML, NLP, and data analytics to analyze article content and user preferences, providing personalized recommendations.

Machine Learning Models:

Supervised Learning: Utilizing labeled data to train models for tasks such as article classification, content prediction, and outcome modeling.
Unsupervised Learning: Extracting patterns and clusters from unlabeled data to identify relationships and insights in article datasets.
Text Analysis: Parsing and extracting information from articles to support content recommendation.
Entity Recognition: Identifying key entities (e.g., topics, keywords) and their relationships in articles for improved recommendation accuracy.

Collaborative Filtering and Recommender Systems:

User Similarity Analysis: Using collaborative filtering techniques to identify similar user profiles and recommend articles based on user preferences and interactions.

Content-Based Filtering: Recommending articles based on their content features and relevance to user preferences.

2.2 A Critical Analysis of Related Work in Article Recommendation Systems

The field of article recommendation systems showcases various strengths and limitations in existing approaches. Researchers and developers have made significant progress in leveraging technologies such as artificial intelligence (AI), machine learning (ML), and natural language processing (NLP) to enhance content discovery and user engagement. However, there are key considerations and areas for improvement:

Strengths of Existing Work:
Personalized Recommendations: Many systems excel in providing personalized recommendations based on user behavior and preferences, leading to increased user engagement and satisfaction.

Content-Based Filtering: Systems that recommend articles based on content features and relevance to user interests often provide accurate and relevant recommendations.

Efficiency and Automation: Automation of article recommendation tasks can streamline content delivery processes, improving user experience and reducing manual effort.

Limitations and Areas for Improvement:
Data Quality and Interoperability: Challenges related to data quality and interoperability hinder the performance of recommendation systems, especially when integrating data from diverse sources.

Privacy and Security Concerns: Ensuring the privacy and security of user data in recommendation systems is essential to maintain user trust and comply with data protection regulations.

User Engagement and Trust: Enhancing user engagement and building trust in AI-driven recommendations requires transparent communication and user involvement in the recommendation process.

Scalability and Generalizability: Recommendation systems must be scalable to handle large amounts of data and generalizable to different content domains and user populations.

In summary, the literature indicates that machine learning techniques, especially those based on NLP and cosine similarity, are highly effective in content recommendation systems. These techniques play a crucial role in helping users discover relevant and engaging content in the vast landscape of online articles, aligning perfectly with the primary goal of this project.

3. System Architecture:

The system architecture of the article recommendation system is designed to efficiently process user interactions, analyze article similarities using cosine similarity, and generate personalized article recommendations. Below is a detailed breakdown of the key components and their interactions within the system architecture:

User Interface (UI):
Input Interface: Allows users to input their pre-read articles.
Output Interface: Displays the recommended articles based on user input.
User Interaction: Provides a user-friendly interface for seamless interaction.

Backend Services:
Input Processing Module: Receives and processes user input, such as user preferences or interests.
Cosine Similarity Calculation: Calculates the cosine similarity between user preferences and article features to identify similar articles.
Recommendation Generation Module: Generates recommendations based on cosine similarity scores.
Data Integration Layer: Accesses datasets containing article information for recommendation generation.

Data Management:
Article Databases: Stores and manages datasets containing article information.
Data Preprocessing: Cleans, transforms, and prepares raw data for use by the cosine similarity calculation module.

Machine Learning Components:

Feature Extraction:

Feature extraction is the process of selecting or transforming relevant information from articles that can be used to calculate the similarity between articles.

It helps in identifying important characteristics or patterns in articles that are used to compute cosine similarity.

Cosine Similarity Model:

The cosine similarity model computes the similarity between two articles based on the cosine of the angle between their feature vectors.

It quantifies the similarity between articles, with a value of 1 indicating identical articles and a value of 0 indicating completely dissimilar articles.

Recommendation Engine:

The recommendation engine is responsible for processing the cosine similarity scores between articles and generating a list of recommended articles.

It selects articles with the highest similarity scores to the user's input or preferences, thereby recommending relevant articles.

Cosine Similarity Processing:

This component processes the output from the cosine similarity calculation to identify the most similar articles.

It filters and ranks the articles based on their similarity scores, ensuring that only the most relevant articles are recommended to the user.

Display and Delivery:

This component is responsible for presenting the recommended articles to the user through the user interface (UI).

It ensures that the recommended articles are displayed in a user-friendly manner, making it easy for the user to access and engage with the content.

The system architecture of the article recommendation platform integrates various components to deliver a seamless and intelligent content recommendation solution. By leveraging cosine similarity and user-friendly interfaces, the architecture enables accurate article recommendations based on user preferences, enhancing user engagement and content discovery. This robust framework ensures scalability, security, and reliability, making it a valuable tool for users seeking personalized article recommendations.

4. Methodology

To achieve the project goal of creating a platform for article classification and recommendation, we tried to adopt the following methodology. First, we collected a dataset comprising crime articles and general articles. This dataset served as the foundation for training the recommendation model. The dataset was then preprocessed to extract relevant features, such as article titles and content summaries, which were used as inputs for the model.

Next, the model was developed using Python, leveraging the scikit-learn library for natural language processing (NLP) tasks. Specifically, the CountVectorizer method was used to convert text data into numerical features, which were then used to compute cosine similarity between articles. Cosine similarity is a measure of similarity between two non-zero vectors and is commonly used in content-based recommendation systems to determine the similarity between items.

Natural Language Processing (NLP) is a branch of artificial intelligence that focuses on the interaction between computers and humans through natural language. It enables computers to understand, interpret, and generate human language, allowing for more natural interactions between humans and machines. In this project, NLP is used to process and analyze text data from articles, enabling the model to recommend similar articles based on their titles.

Stemming and lemmatization are two common techniques used in NLP to reduce words to their base or root form.

- Stemming involves cutting off prefixes or suffixes of words to extract their base form. For example, the words "running," "ran," and "runs" would all be reduced to "run."
- Lemmatization, on the other hand, reduces words to their dictionary form or lemma. For instance, "better" would be lemmatized to "good."

These techniques help standardize words, ensuring that different variations of the same word are treated as the same, which is crucial for accurately analyzing and recommending articles based on their titles.

Cosine similarity

Cosine similarity is a metric used to measure the similarity between two vectors, often used in natural language processing and information retrieval. It calculates the cosine of the angle between two non-zero vectors and produces a value between -1 and 1. A value of 1 indicates that the vectors are identical; 0 means they are orthogonal (i.e., unrelated); and -1 indicates they are diametrically opposed.

$$\cos(\theta) = \frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\|\|\mathbf{B}\|} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2} \sqrt{\sum_{i=1}^{n} B_i^2}}$$

Fig:1 Formula for cosine similarity

In the context of this project, cosine similarity is used to compare the similarity between article titles. Each article title is represented as a vector in a high-dimensional space, where each dimension corresponds to a unique word in the vocabulary. By calculating the cosine similarity between the vectors representing two article titles, the model can determine how similar they are.

4.1 Dataset Description

We utilize two datasets, one comprising crime articles and the other containing general articles, to recommend similar articles based on their titles. By combining these datasets, the model can offer a diverse range of content for recommendation, ensuring that users receive relevant and engaging articles tailored to their interests.

The model analyzes the titles of the articles using cosine similarity, a measure of similarity between two non-zero vectors, to recommend the top 5 most related articles to the user. This approach enhances the browsing experience by providing users with a curated selection of articles that align with their preferences, making it easier for them to discover new and interesting content.

7k unique crime articles

Dataset Link: https://www.kaggle.com/code/soumendraprasad/crime-articles-recommendation-system/input?select=7k++Unique+crime+articles.csv

This dataset contains articles related to crime incidents, providing information about various criminal activities. It includes details about crimes such as theft, assault, and other unlawful acts, along with reports and summaries of the incidents. The dataset aims to capture a wide range of criminal activities to provide a comprehensive view of crime trends and patterns.

It contains 8 columns.

- **heading**- This column Contain Heading of Article
- **content_summary**- This column contains summary of the article
- **article_link-** This columncontains link of the respective articles
- **img_link-** This column contains image links with respect to individual articles.
- **Month Date:** Date when the article was published, represented in month-date format.
- **time-** This column contains time of publication. (IST time zone)
- **Year -** This column contains the year of publication.

The heading and content summary are crucial features for recommending similar articles. These features provide a concise representation of the article's content, which can be used to match with other articles based on similarity.

General Articles Dataset:

Dataset Link:

https://github.com/YamiAtem/ArticleRecomendation/blob/main/data/articles_data.csv

This dataset consists of general articles covering various topics beyond crime, offering a wide range of content for recommendation. It includes articles on diverse subjects such as technology, business, health, and lifestyle, providing a comprehensive collection of articles for users to explore. The dataset aims to cater to a broad audience with varied interests, ensuring that users can find articles relevant to their preferences. Each article in the dataset is labelled with a title, text, and URL, making it easy for users to access and read the articles.

Features:

- **id:** The unique identifier for each article.
- **index:** The index of the article in the dataset.
- **timestamp:** The timestamp of when the article was shared or published.
- **eventType:** The type of event related to the article (e.g., "CONTENT SHARED").
- **contentId:** The content identifier of the article.
- **authorPersonId:** The unique identifier of the author of the article.
- **authorSessionId:** The session identifier of the author when the article was published.
- **authorUserAgent:** The user agent of the author who published the article.
- **authorRegion:** The region of the author who published the article.
- **authorCountry:** The country of the author who published the article.
- **contentType:** The type of content of the article (e.g., "HTML").
- **url:** The URL link to access the full article.
- **title:** The title or headline of the article.

- **text:** The text content of the article.
- **lang:** The language of the article (e.g., "en" for English).

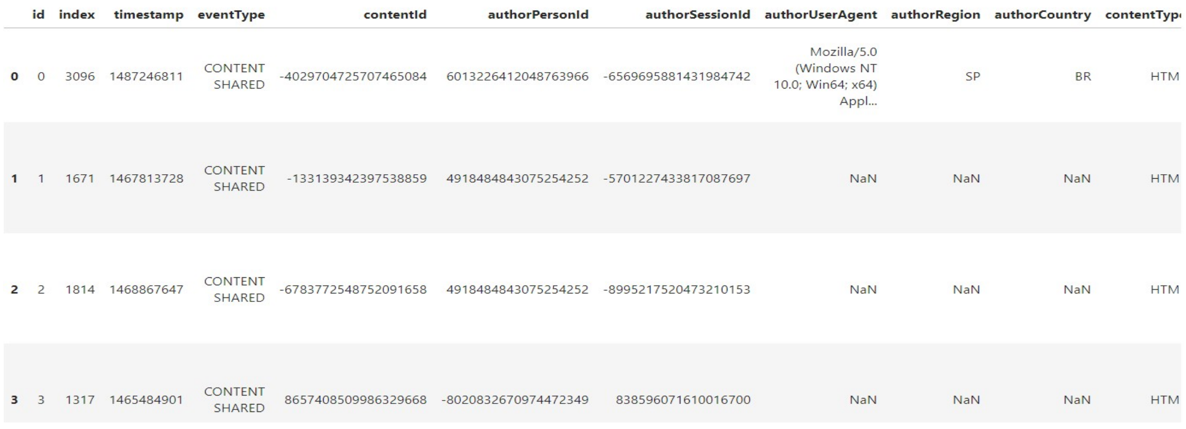
The title and text features are crucial for recommending similar articles. These features provide detailed information about the article's content, allowing for a more in-depth analysis of similarities between articles.

4.2 Data Preprocessing:

Data preprocessing is a crucial step in preparing the dataset for analysis and modeling. In this project, we performed several preprocessing steps on the crime and general articles datasets to ensure the data is clean and ready for training the recommendation model.

General Articles Dataset Statistics:

Loading a dataset from 'articles_data.csv' into a pandas Data Frame and sorting this Data Frame in descending order based on the 'total_events' column.
The total_events column in the dataset represents a crucial metric, indicating the number of interactions or activities associated with each article listed.

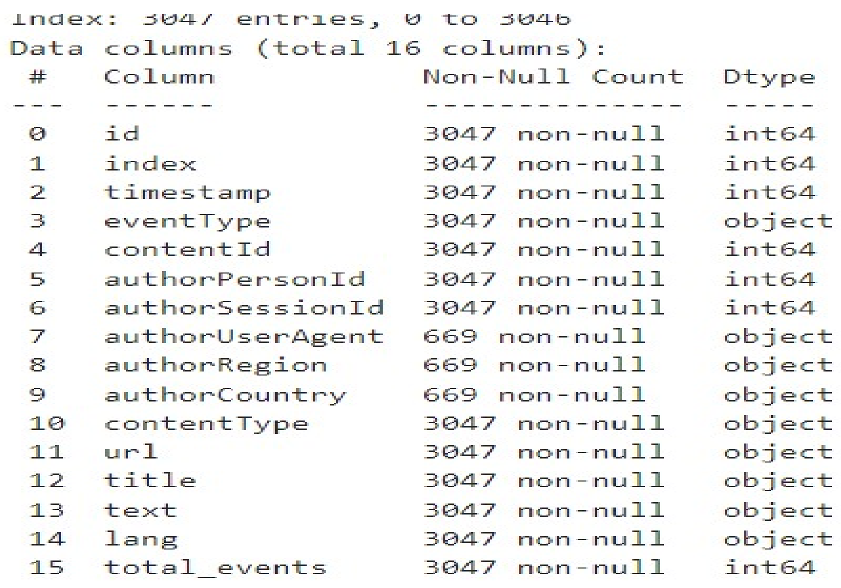


	id	index	timestamp	eventType	contentId	authorPersonId	authorSessionId	authorUserAgent	authorRegion	authorCountry	contentTyp
0	0	3096	1487246811	CONTENT SHARED	-4029704725707465084	6013226412048763966	-6569695881431984742	Mozilla/5.0 (Windows NT 10.0; Win64; x64) Appl...	SP	BR	HTM
1	1	1671	1467813728	CONTENT SHARED	-133139342397538859	4918484843075254252	-5701227433817087697	NaN	NaN	NaN	HTM
2	2	1814	1468867647	CONTENT SHARED	-6783772548752091658	4918484843075254252	-8995217520473210153	NaN	NaN	NaN	HTM
3	3	1317	1465484901	CONTENT SHARED	8657408509986329668	-8020832670974472349	838596071610016700	NaN	NaN	NaN	HTM

Fig:2 view over the dataset.

articles.info()

It provides a concise summary of the articles dataset, including the data types of each column and the number of non-null elements in each column. It helps to quickly understand the structure of the dataset and identify any missing values.



```
Index: 3047 entries, 0 to 3046
Data columns (total 16 columns):
 #   Column           Non-Null Count  Dtype
---  ------           --------------  -----
 0   id               3047 non-null   int64
 1   index            3047 non-null   int64
 2   timestamp        3047 non-null   int64
 3   eventType        3047 non-null   object
 4   contentId        3047 non-null   int64
 5   authorPersonId   3047 non-null   int64
 6   authorSessionId  3047 non-null   int64
 7   authorUserAgent  669 non-null    object
 8   authorRegion     669 non-null    object
 9   authorCountry    669 non-null    object
 10  contentType      3047 non-null   object
 11  url              3047 non-null   object
 12  title            3047 non-null   object
 13  text             3047 non-null   object
 14  lang             3047 non-null   object
 15  total_events     3047 non-null   int64
```

Fig:3 Total non-null count

articles.shape

This provides the dimensions of the whole data set. The number of rows and the number of columns.

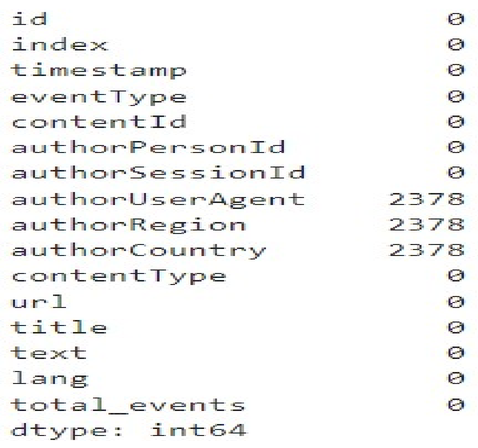
(3047, 19)

Our data set has 3047 rows and 19 columns.

To calculate the total number of null elements in each.

articles.isnull().sum()

It calculates the total number of null (missing) elements in each column of the article dataset. This information is crucial for data cleaning and preprocessing steps, as null values can impact the analysis and recommendations derived from the dataset.



```
id                     0
index                  0
timestamp              0
eventType              0
contentId              0
authorPersonId         0
authorSessionId        0
authorUserAgent     2378
authorRegion        2378
authorCountry       2378
contentType            0
url                    0
title                  0
text                   0
lang                   0
total_events           0
dtype: int64
```

Fig :4 count of null

In this case, since the null values do not contribute significantly to the recommendation process, they are being replaced with a null value. This step ensures that the dataset is clean and ready for further analysis and recommendation generation.

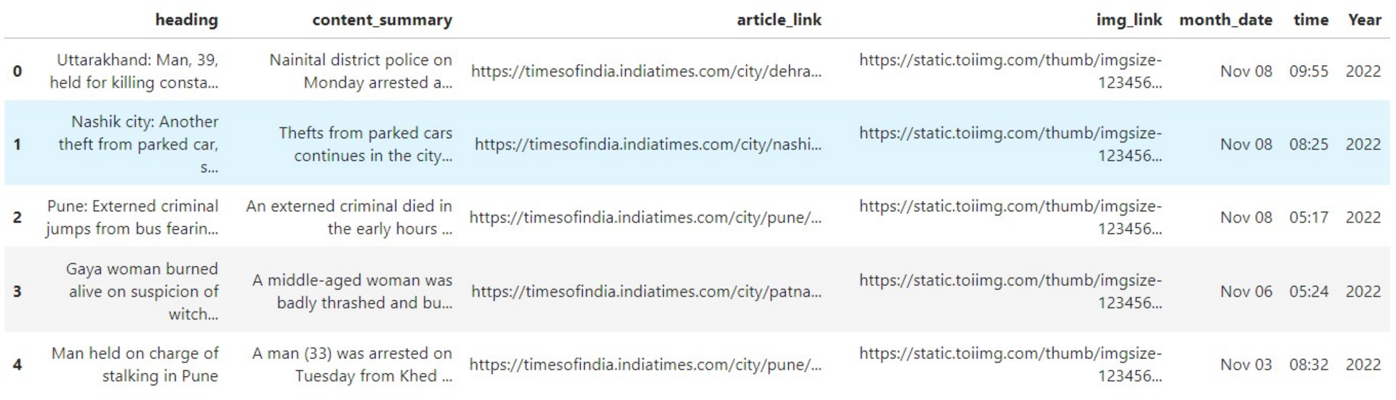
```
#To check duplicate values
articles.duplicated().sum()
0
```

It is used to check for duplicate rows in the article dataset. It calculates the total number of duplicated rows based on all columns and returns the sum of these duplicates. In this case, the result is 0, indicating that there are no duplicate rows present in the dataset.

This information is important because duplicate rows can skew analysis and recommendations, leading to inaccurate results. By verifying the absence of duplicate rows, we ensure the integrity of the dataset for our recommendation system.

Crime Articles Dataset Statistics:

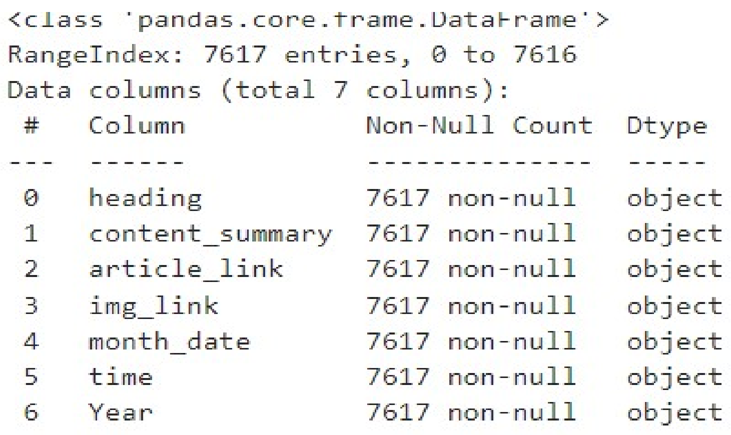
This code reads a file called '7k Unique crime articles.csv' and stores its contents in a table-like format. Then, it shows the first few rows of this table, giving you a preview of what's in the file. This helps you understand the kind of information the file contains.



	heading	content_summary	article_link	img_link	month_date	time	Year
0	Uttarakhand: Man, 39, held for killing consta...	Nainital district police on Monday arrested a...	https://timesofindia.indiatimes.com/city/dehra...	https://static.toiimg.com/thumb/imgsize-123456...	Nov 08	09:55	2022
1	Nashik city: Another theft from parked car, s...	Thefts from parked cars continues in the city...	https://timesofindia.indiatimes.com/city/nashi...	https://static.toiimg.com/thumb/imgsize-123456...	Nov 08	08:25	2022
2	Pune: Externed criminal jumps from bus fearin...	An externed criminal died in the early hours ...	https://timesofindia.indiatimes.com/city/pune/...	https://static.toiimg.com/thumb/imgsize-123456...	Nov 08	05:17	2022
3	Gaya woman burned alive on suspicion of witch...	A middle-aged woman was badly thrashed and bu...	https://timesofindia.indiatimes.com/city/patna...	https://static.toiimg.com/thumb/imgsize-123456...	Nov 06	05:24	2022
4	Man held on charge of stalking in Pune	A man (33) was arrested on Tuesday from Khed ...	https://timesofindia.indiatimes.com/city/pune/...	https://static.toiimg.com/thumb/imgsize-123456...	Nov 03	08:32	2022

Fig:5 Overview of the crime article data

df.info()



```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 7617 entries, 0 to 7616
Data columns (total 7 columns):
 #   Column           Non-Null Count  Dtype 
---  ------           --------------  ----- 
 0   heading          7617 non-null   object
 1   content_summary  7617 non-null   object
 2   article_link     7617 non-null   object
 3   img_link         7617 non-null   object
 4   month_date       7617 non-null   object
 5   time             7617 non-null   object
 6   Year             7617 non-null   object
```

Fig:6 non-null rows count

It provides an overview of the dataset, including the total number of entries (rows) and the number of columns. It also displays the data type of each column, which helps in understanding how the data is stored. Additionally, it shows the number of non-null values in each column, indicating if there are any missing values that need to be addressed. This information is essential for data cleaning and preparation before analysis.

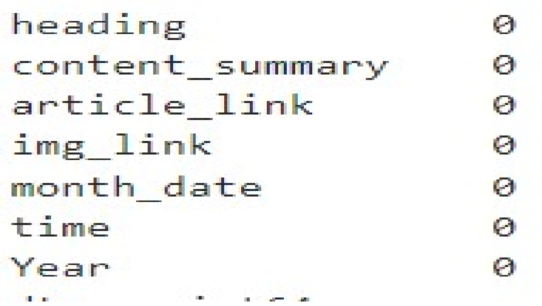
df.shape

This provides the dimensions of the whole data set. The number of rows and the number of columns.

(7617, 7)

Our data set has 7617 rows and 7 columns.

df.isnull().sum()



```
heading            0
content_summary    0
article_link       0
img_link           0
month_date         0
time               0
Year               0
```

Fig:7 count of null in dataset

It checks for null values in each column of the dataset and returns the total number of null values found in each column. In this case, the result indicates that there are no null values present in any column of the dataset. This is beneficial because it means that the dataset is complete and does not require any imputation or handling of missing values before analysis.

df.duplicated().sum()

The function checks for duplicated rows in the dataset and returns the total number of duplicated rows found. In this case, the result indicates that there are no duplicate rows present in the dataset.

This is important because duplicated rows can skew the analysis results and lead to incorrect conclusions. The absence of duplicated rows ensures the integrity of the dataset and indicates that each row is unique, which is crucial for accurate analysis and modelling.

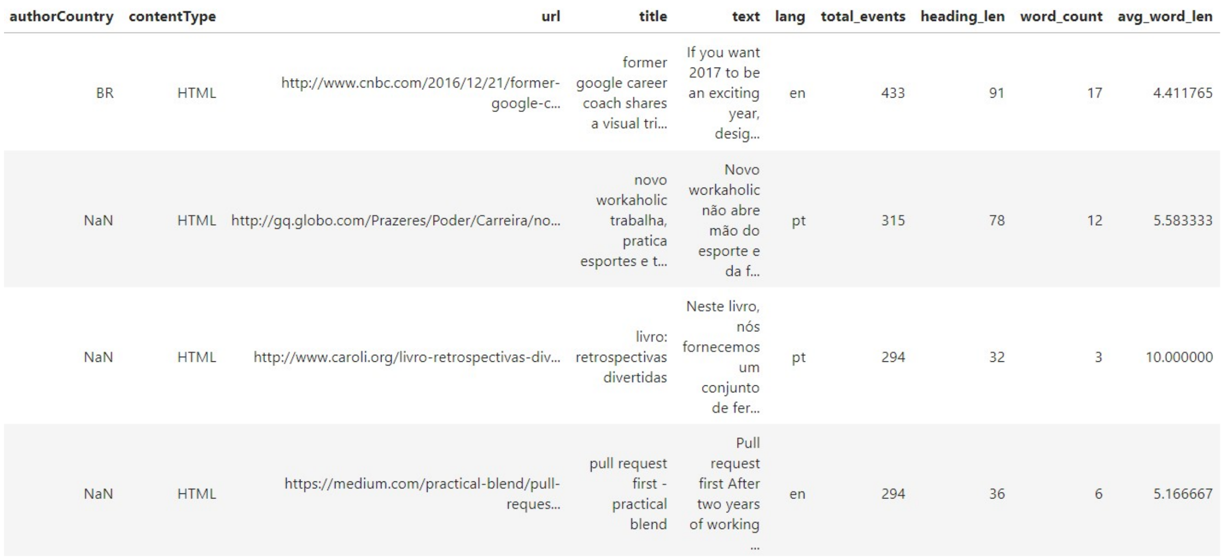
Creating some column to know the words distribution

articles["heading_len"]=articles["title"]. apply(lambda x:len(x)): This line calculates the length of each article's title (number of characters) and assigns it to the new column heading_len. It uses a lambda function to apply the len() function to each title in the title column.

articles["word_count"]=articles["title"]. apply(lambda x:(len(x.split()))): This line calculates the number of words in each article's title and assigns it to the new column word_count. It uses a lambda function to split each title into words using the split() method and then calculates the length of the resulting list, which gives the number of words in the title.

The get_avg_word_len function calculates the average word length in each text. This function is useful in text processing tasks as it provides insights into the complexity and structure of the text. By applying this function to both datasets, we can analyze the average word length in the article headings and summaries. This analysis can help us understand the typical length of words used in the articles, which can be useful in various NLP tasks.

By calculating and adding a new column called avg_word_len to the DataFrame, we now have a way to understand the average length of the words in each article title. This new detail can help us explore how the length of words in a title might relate to how much attention the article gets from readers. In simpler terms, it can help us see whether shorter or longer words in titles might make articles more or less appealing to readers.

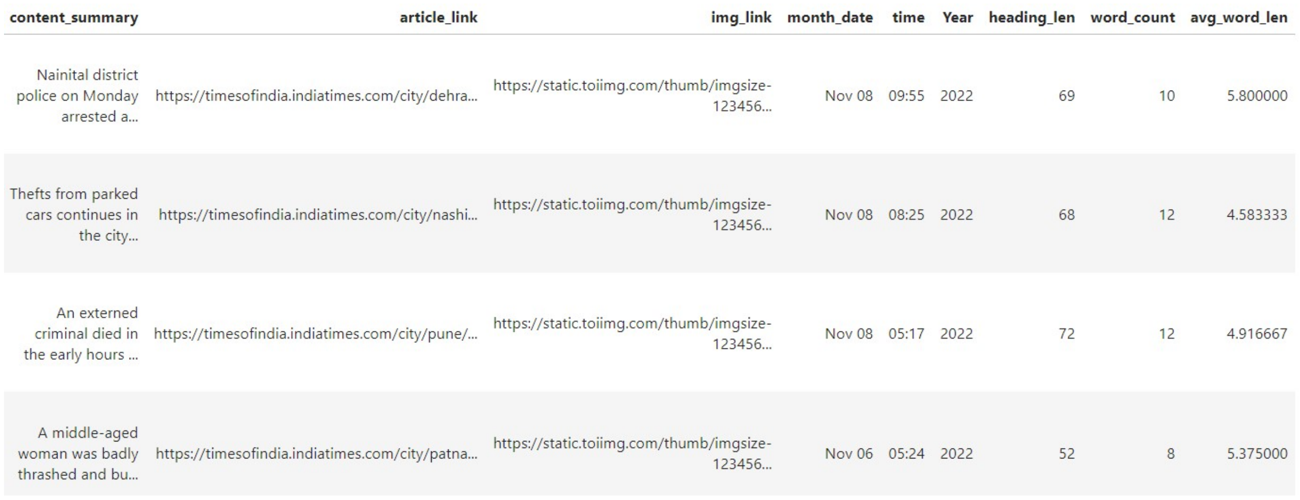


authorCountry	contentType	url	title	text	lang	total_events	heading_len	word_count	avg_word_len
BR	HTML	http://www.cnbc.com/2016/12/21/former-google-c...	former google career coach shares a visual tri...	If you want 2017 to be an exciting year, desig...	en	433	91	17	4.411765
NaN	HTML	http://gq.globo.com/Prazeres/Poder/Carreira/no...	novo workaholic trabalha, pratica esportes e t...	Novo workaholic não abre mão do esporte e da f...	pt	315	78	12	5.583333
NaN	HTML	http://www.caroli.org/livro-retrospectivas-div...	livro: retrospectivas divertidas	Neste livro, nós fornecemos um conjunto de fer...	pt	294	32	3	10.000000
NaN	HTML	https://medium.com/practical-blend/pull-reques...	pull request first - practical blend	Pull request first After two years of working ...	en	294	36	6	5.166667

Fig:8 overview after adding the columns

Creating some column to know the words distribution (for crime dataset)

We've added new columns to the DataFrame for crime articles to include additional information. This information includes the length of the heading (title) of each article, the total number of words in the heading, and the average length of words in the heading. These columns provide insights into the structure and content of the articles, which can be useful for further analysis and understanding of the dataset.

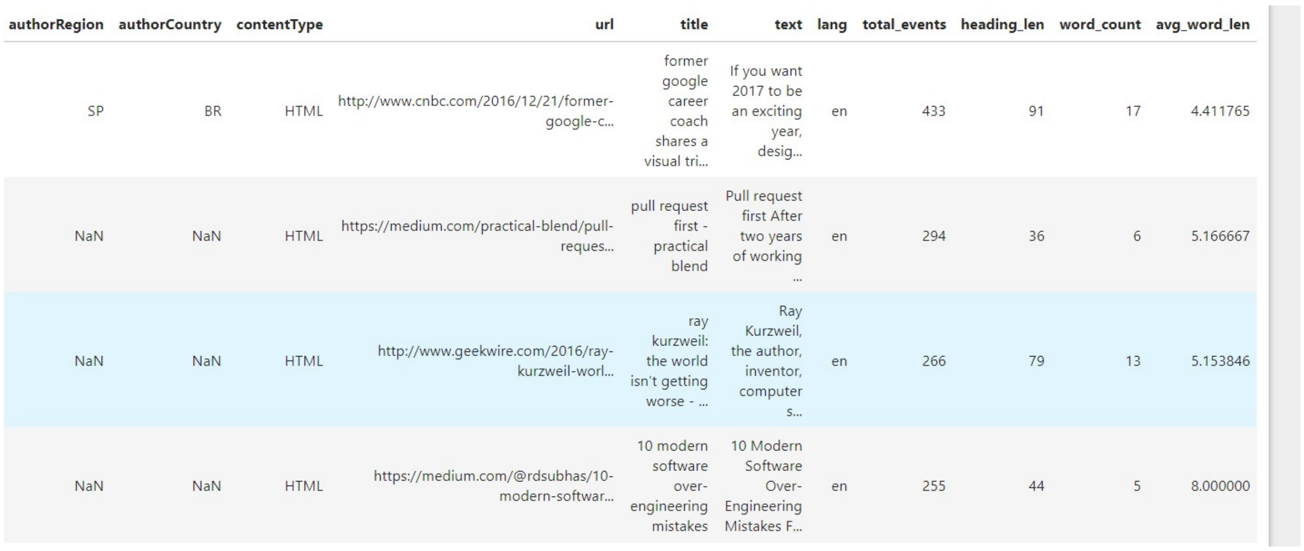


content_summary	article_link	img_link	month_date	time	Year	heading_len	word_count	avg_word_len
Nainital district police on Monday arrested a...	https://timesofindia.indiatimes.com/city/dehra...	https://static.toiimg.com/thumb/imgsize-123456...	Nov 08	09:55	2022	69	10	5.800000
Thefts from parked cars continues in the city...	https://timesofindia.indiatimes.com/city/nashi...	https://static.toiimg.com/thumb/imgsize-123456...	Nov 08	08:25	2022	68	12	4.583333
An externed criminal died in the early hours ...	https://timesofindia.indiatimes.com/city/pune/...	https://static.toiimg.com/thumb/imgsize-123456...	Nov 08	05:17	2022	72	12	4.916667
A middle-aged woman was badly thrashed and bu...	https://timesofindia.indiatimes.com/city/patna...	https://static.toiimg.com/thumb/imgsize-123456...	Nov 06	05:24	2022	52	8	5.375000

Fig:9 Filtering the dataset to retain only English articles for model training.

As we can see in the Articles dataset, most of the articles are in English, so we are going to take a data frame for only English articles and drop the remaining ones in articles dataset. As the size of the other articles is very small, we won't consider them for training the model.

```
df_en = articles[articles['lang'] == 'en']  df_en.head()
```



authorRegion	authorCountry	contentType	url	title	text	lang	total_events	heading_len	word_count	avg_word_len
SP	BR	HTML	http://www.cnbc.com/2016/12/21/former-google-c...	former google career coach shares a visual tri...	If you want 2017 to be an exciting year, desig...	en	433	91	17	4.411765
NaN	NaN	HTML	https://medium.com/practical-blend/pull-reques...	pull request first - practical blend	Pull request first After two years of working ...	en	294	36	6	5.166667
NaN	NaN	HTML	http://www.geekwire.com/2016/ray-kurzweil-worl...	ray kurzweil: the world isn't getting worse - ...	Ray Kurzweil, the author, inventor, computer s...	en	266	79	13	5.153846
NaN	NaN	HTML	https://medium.com/@rdsubhas/10-modern-softwar...	10 modern software over-engineering mistakes	10 Modern Software Over-Engineering Mistakes F...	en	255	44	5	8.000000

Fig:10 English articles over view

This provides the dimensions of the whole data set. The number of rows and the number of columns.

```
print(df_en.shape)
(2211, 19)
```

There are 2211 rows and 19 columns in the dataset.

Converting the heading and summary to lowercase in both datasets is a crucial preprocessing step in natural language processing (NLP) tasks. This process ensures that the text data is normalized and consistent, avoiding issues related to case sensitivity.

By converting the text to lowercase, we can avoid treating words with different cases as different entities, which could lead to inaccurate results in our analysis. Additionally, converting the text to lowercase helps in reducing the complexity of the text data, making it easier to process and analyze. This step is particularly useful when performing tasks such as text matching, where case sensitivity is not required.

By applying this step to the heading and summary columns in both datasets, we ensure that the text data is uniform and ready for further processing, such as tokenization and vectorization, which are common tasks in NLP.

The remove_tags function is used to remove HTML tags from text data. In the context of the dataset, this function is crucial for cleaning and preprocessing the text, especially if the dataset contains articles or content copied from web pages where HTML tags are present.

HTML tags are used to format and structure web pages but are unnecessary and can be distracting when analyzing the text. By removing these tags, we ensure that the text is clean and ready for further processing, such as natural language processing (NLP) tasks.

Removing HTML tags is significant because it allows us to focus on the actual content of the text, rather than the formatting. This is important for tasks like text analysis, sentiment analysis, and machine learning, where the content of the text is more relevant than its presentation.

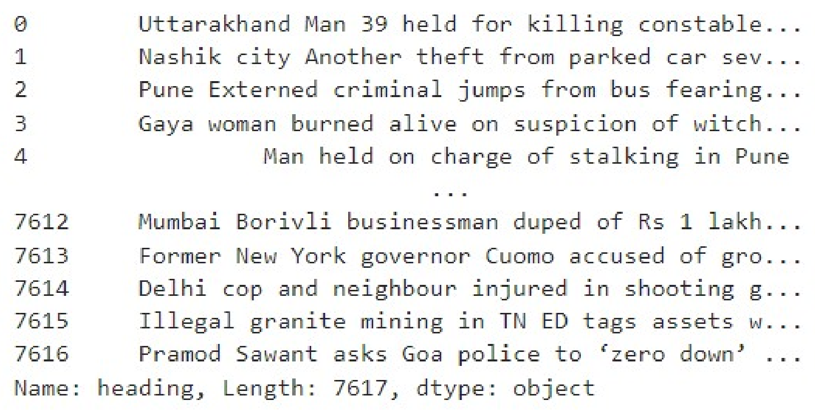
The remove_punc function is used to remove punctuation from text data. Punctuation marks such as periods, commas, and exclamation marks are common in text but are often unnecessary for certain text analysis tasks. Removing punctuation helps to normalize the text and can improve the accuracy of text processing algorithms.

By removing punctuation, we ensure that the text is cleaned and ready for further analysis. This is important for tasks such as text classification, where the presence of punctuation may not be relevant to the classification process. Additionally, removing punctuation can help reduce the size of the vocabulary, which can be beneficial for some machine learning models.

The stem function applies stemming to text data. Stemming is the process of reducing words to their base or root form, called the stem. This process helps to normalize words so that different forms of the same word are treated as the same word, which can improve the performance of text analysis algorithms.

In the stem function, the ps.stem method from the nltk library is used to stem each word in the text. The stemmed words are then joined back together to form the processed text. This preprocessing step is common in text analysis tasks where word variations are not critical to the analysis, such as in information retrieval or text mining.

Resultant data frame looks like this:



```
0        Uttarakhand Man 39 held for killing constable...
1        Nashik city Another theft from parked car sev...
2        Pune Externed criminal jumps from bus fearing...
3        Gaya woman burned alive on suspicion of witch...
4                  Man held on charge of stalking in Pune
                                ...
7612     Mumbai Borivli businessman duped of Rs 1 lakh...
7613     Former New York governor Cuomo accused of gro...
7614     Delhi cop and neighbour injured in shooting g...
7615     Illegal granite mining in TN ED tags assets w...
7616     Pramod Sawant asks Goa police to ‘zero down’ ...
Name: heading, Length: 7617, dtype: object
```

Fig:11 Data frame of crime articles

4.3 Exploratory Data Analysis:

Exploratory data analysis (EDA) is a critical step in understanding the dataset's structure, identifying patterns, and extracting insights that can inform the modeling process. It involves summarizing the main characteristics of the dataset, often using statistical and visualization techniques.

EDA helps in detecting anomalies, checking assumptions, and formulating hypotheses for further analysis. It lays the foundation for building models by providing insights into the relationships between variables and the underlying distribution of the data.

For 7k unique crime articles

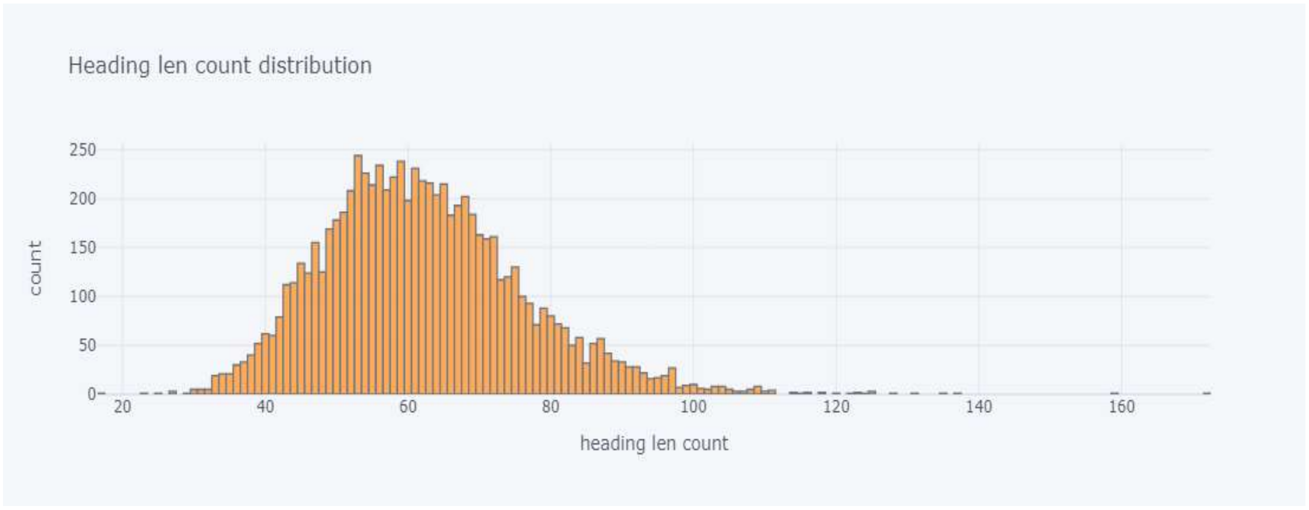


Fig:12 heading length trends, influencing article structure and engagement.

This plot illustrates the distribution of heading lengths in the articles. It provides insights into the length of headings used in the articles and can be indicative of the level of detail or informativeness of the headings. Analyzing heading lengths can help in understanding how articles are structured and titled, which is crucial for content organization and user engagement.

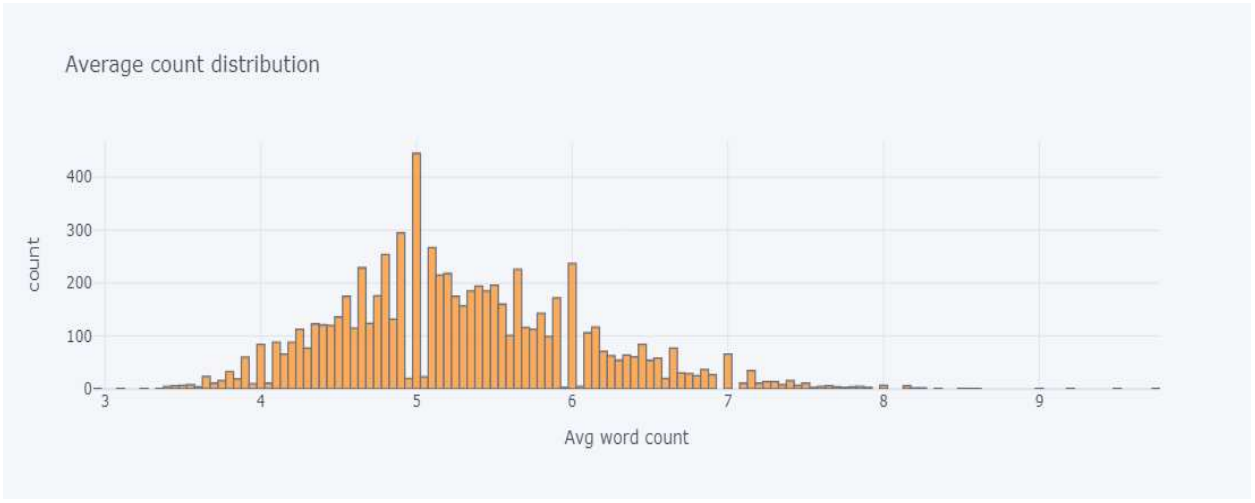


Fig:13 variations in average word lengths

This plot displays the distribution of the average word lengths in the articles. It helps in understanding the complexity or readability of the articles. Variations in average word length can indicate differences in writing styles or content types, which can be useful for content analysis.

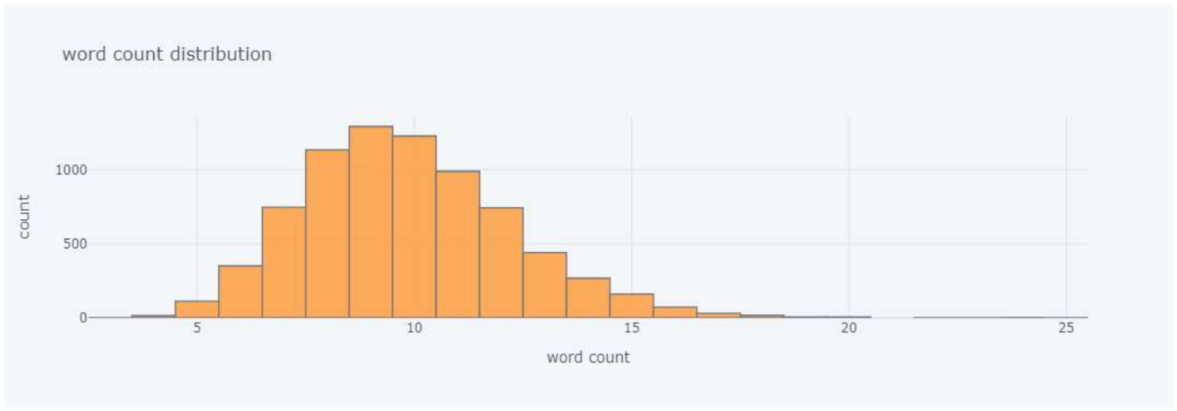


Fig:14 Word count distribution and outliers in articles dataset.

This plot shows the distribution of the word counts in the articles. It can provide insights into the length of the articles and the variation in word counts across the dataset. This information can clarify the typical length of articles in the dataset and identify any outliers or patterns in article length.

General Articles Dataset:

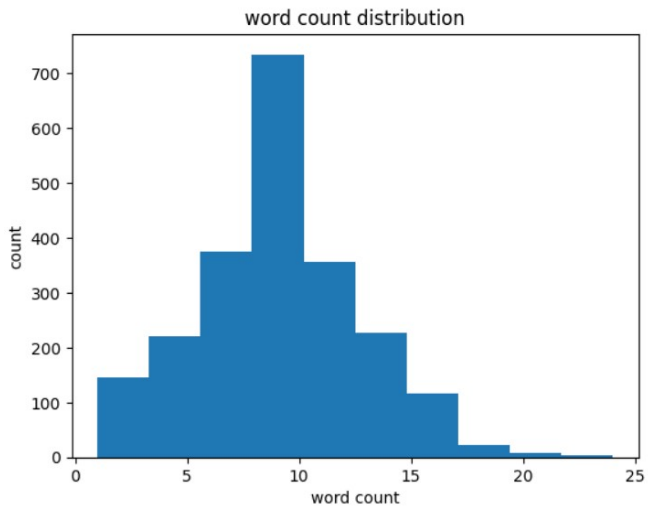


Fig:15 Word count distribution

This plot shows the distribution of word counts in the English article's dataset (df_en). It provides insights into the length of the articles and the variation in word counts across the dataset.

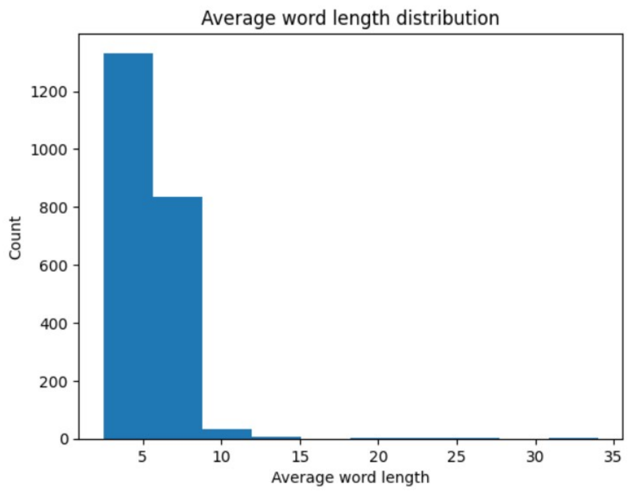


Fig:16 Average of the length of words

This plot displays the distribution of average word lengths in the English article's dataset (df_en). It helps in understanding the complexity or readability of the articles.

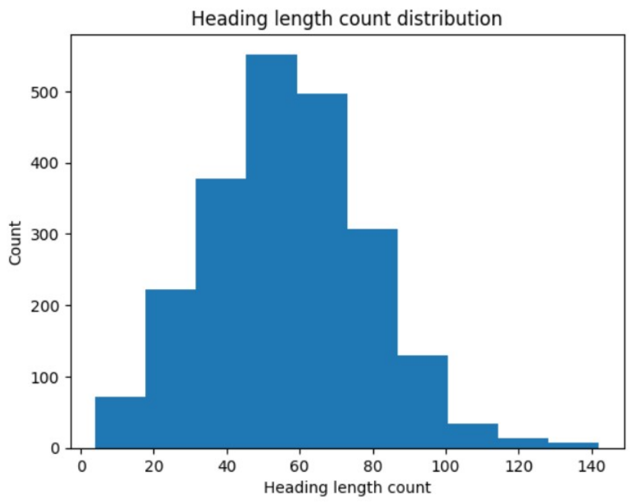


Fig:17 The length of words used in heading

This plot illustrates the distribution of heading lengths in the English article's dataset (df_en). It provides insights into the length of headings used in the articles.

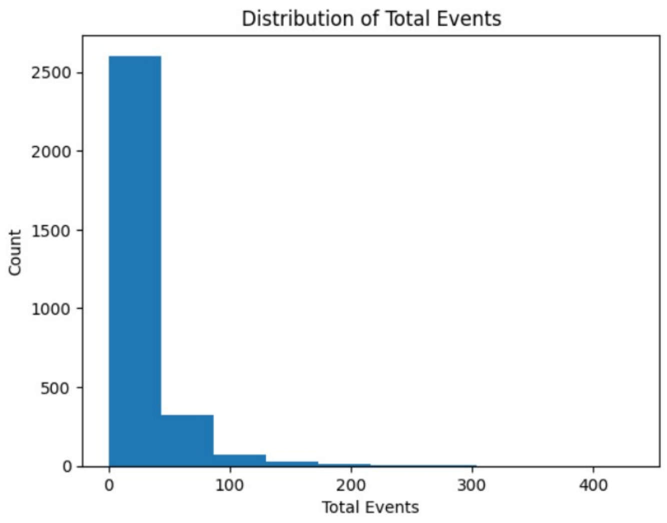


Fig:18 The distribution of total events count

This plot shows the distribution of total events in the article's dataset. It helps in understanding the frequency of events covered in the articles.

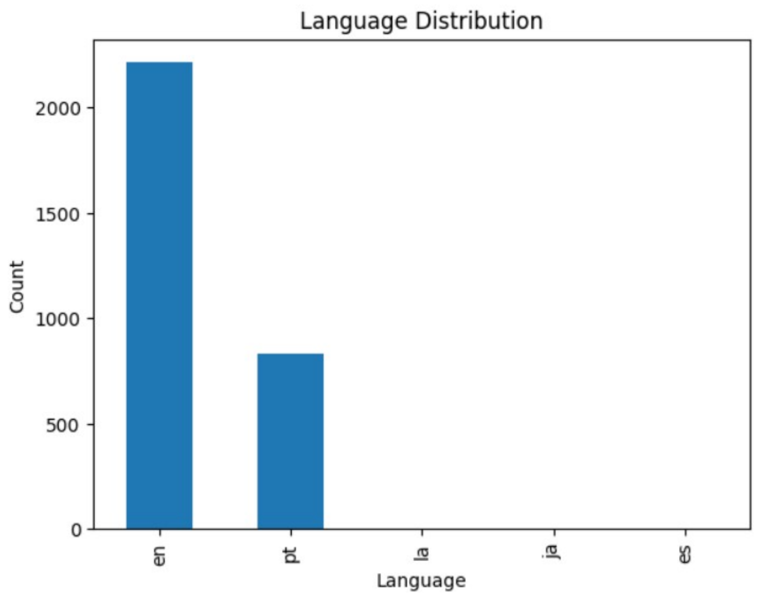
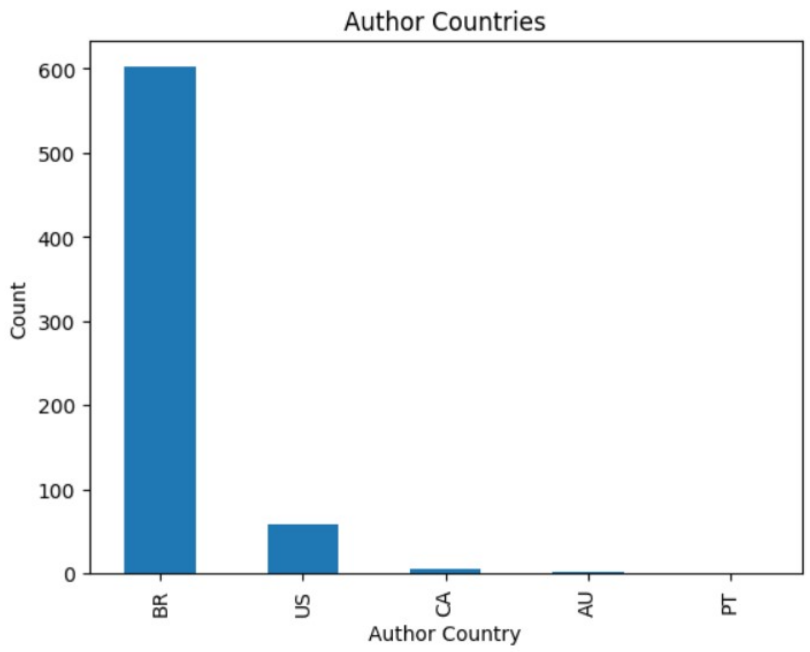


Fig:19 language distribution plot

This bar plot displays the distribution of languages in the article's dataset. It provides insights into the languages in which the articles are written. The articles written in English stands at the first place with over 2000+ articles and at second place is Portuguese



.

Fig:20 Author Countries plot

This bar plot displays the distribution of the top 10 author countries in the article's dataset. It provides insights into the countries to which the authors of the articles belong.

In conclusion, our exploratory data analysis (EDA) of the crime articles dataset has provided valuable insights into the characteristics and distribution of the data. We have gained an understanding of the dataset's structure, including its size, columns, and data types. Through various analyses, we have identified trends such as the distribution of words in article headings and the average word length. Additionally, we have confirmed that the dataset contains no null values or duplicated rows. These findings lay a solid foundation for further analysis and modeling, enabling us to extract meaningful information and insights from the dataset.

5. Implementation:

The project utilizes two distinct datasets to facilitate article recommendation. The first dataset comprises crime-related articles, offering insights into various criminal activities. It includes details such as crime types, locations, and dates, providing a comprehensive view of crime-related content. The second dataset consists of general articles covering a wide range of topics, such as technology, sports, and entertainment. This dataset adds diversity to the recommendation system, ensuring a broad selection of articles for users to explore.

5.1 Key Features in Each Dataset:

Features in a dataset are the individual measurable properties or characteristics of the data points. They provide essential information about each data point, helping to understand the underlying patterns and relationships within the data. Features serve as the building blocks for understanding and classifying the data, as they contain valuable insights that can be used to distinguish between different data points or categories.

By analyzing the features, one can identify important patterns, trends, and correlations, which are crucial for making informed decisions or predictions. Features play a key role in machine learning models, where they are used to train the model to recognize patterns and make accurate predictions or classifications based on new, unseen data.

Key Features in the Crime Articles Dataset:

The crime articles dataset contains features like crime types, locations, and dates, which are crucial for identifying patterns and similarities in crime-related content. These features help the model recommend articles that share similar crime types, locations, or dates, enhancing the relevance of recommendations for users interested in crime-related content.

- **Crime Types:** This feature categorizes articles based on the type of crime discussed, such as theft, assault, or fraud. It helps the model recommend articles that discuss similar types of crimes, providing users with a cohesive reading experience on specific crime topics.

- **Locations:** The location feature specifies the place where the crime occurred. It enables the model to recommend articles related to crimes in similar locations, helping users stay informed about criminal activities in specific areas of interest.
- **Dates:** The date feature indicates when the crime took place or when the article was published. It allows the model to recommend articles based on temporal relevance, suggesting recent articles or articles from specific time periods.

Key Features in the General Articles Dataset:

On the other hand, the general articles dataset includes features like article topics, publication dates, and authors, providing insights into the broader context of the articles. These features contribute to the diversity of recommendations, ensuring that users receive a varied selection of articles from different topics and authors.

- **Article Topics:** This feature categorizes articles based on their topics, such as technology, sports, or entertainment. It helps the model recommend articles from similar topics, ensuring that users receive a diverse selection of content based on their interests.
- **Publication Dates:** Like the crime articles dataset, the publication date feature indicates when the article was published. It allows the model to recommend recent articles or articles from specific time periods, ensuring that users receive up-to-date content.
- **Authors:** The author specifies the author or authors of the article. It helps the model recommend articles from specific authors, allowing users to explore content from their favorite writers or experts in certain fields.

5.2 Data Preprocessing:

In the dataset preprocessing stage, several key steps were applied to clean and standardize the data for further analysis. The first step involved converting all text to lowercase to ensure uniformity in the dataset, as this helps avoid discrepancies that might arise from inconsistent casing. Next, HTML tags were removed from the text to clean it and ensure that the content was in a readable format. Following this, punctuation was removed from the text to focus on the textual content itself, disregarding punctuation that does not contribute to the meaning of the text. Additionally, stop words, which are common words that do not add much meaning to the text (e.g., "the," "and" "is"), were removed to reduce noise in the dataset and improve the quality of the analysis.

Another crucial step in the preprocessing stage was stemming, which involves reducing words to their base or root form. This step aids in text normalization, ensuring that different variations of the same word are treated as the same word. For example, "running" and "ran" would both be stemmed to "run." This process helps improve the performance of the model by reducing the complexity of the text data.

After applying the preprocessing steps, the dataset was ready for exploratory data analysis (EDA) and model building. EDA involves analyzing and visualizing the data to gain insights and understand the characteristics of the dataset. For example, visualizing the distribution of word counts in the dataset helps understand the length of the articles, which can be useful in determining the complexity of the language used. Similarly, visualizing the average word lengths can provide insights into the language structure and complexity of the articles.

In conclusion, the dataset preprocessing stage plays a crucial role in preparing the data for analysis in an article recommendation system. By applying steps such as converting text to lowercase, removing HTML tags and punctuation, removing stop words, and stemming, the dataset was cleaned and standardized, ensuring that it was suitable for further analysis.

5.3 Model Building and Fitting:

Key Features in the Datasets

In our project, we identified key features in the two datasets, crime articles, and general articles, which are essential for building an effective recommendation system. For the general article's dataset, the key features include the article title and text, as these elements encapsulate the main content and topic of the articles. Similarly, in the crime articles dataset, the key features are the heading and content summary, which provide a concise overview of the article's main points. These features are significant because they help in identifying the main themes and topics of the articles, which are crucial for recommending similar articles to users based on their interests.

Text Data Transformation with Count Vectorizer

We used the Count Vectorizer from the scikit-learn library to convert the text data into numerical vectors. Count Vectorizer is used to tokenize the text data and convert it into a matrix format, where each row represents a document (article) and each column represents a unique word in the corpus. This transformation allows the model to understand the textual content of the articles and calculate cosine similarity scores between articles based on their word frequencies. We chose Count Vectorizer because it is a simple yet effective method for converting text data into a format suitable for machine learning models.

```
cv=CountVectorizer(max_features=5000, stop_words="english")
```

1. **Tokenization**: The CountVectorizer tokenizes the text data, which means it splits the text into individual words or tokens. It removes punctuation and converts all text to lowercase to ensure uniformity.
2. **Vocabulary Building**: It builds a vocabulary of all the unique words in the text data. Each word becomes a feature, and the position of the word in the vocabulary is its index in the feature matrix.

3. **Vectorization**: For each document (article), Count Vectorizer counts the frequency of each word in the document and fills the corresponding cell in the feature matrix. This results in a matrix where each row represents a document, and each column represents a unique word in the vocabulary.
4. **Stop Words Removal**: The stop_words="english" parameter removes common English stop words like "and", "the", "is", etc. These words are often removed because they do not carry much information for the model.
5. **Limiting Features**: The max_features=5000 parameter limits the number of unique words (features) to 5000. This can help reduce the dimensionality of the feature matrix and improve the performance of the model, especially if there are many words in the vocabulary.

Serialization with Pickle

Serialization is the process of converting an object into a byte stream, which can be saved to a file or transmitted over a network. We use the pickle library to serialize and save the trained model. Pickle allows us to save the trained model to a file, which can be loaded later to make predictions without retraining the model. This is particularly useful for saving time and resources, especially when dealing with large datasets and complex models.

```
import pickle

pickle.dump(cv,open('cv','wb'))

cv = pickle.load(open('cv','rb'))
```

The pickle module in Python is used for serializing and deserializing Python objects. Serialization is the process of converting a Python object into a byte stream that can be saved to a file or transmitted over a network. Deserialization is the reverse process, where the byte stream is converted back into a Python object. In your code, pickle.dump(cv, open ('cv', 'wb')) serializes the Count Vectorizer object cv and saves it to a file named 'cv' in binary mode ('wb' stands for write binary). This allows you to save the trained Count Vectorizer object for later use without having to retrain it every time.

Model Training and Fitting

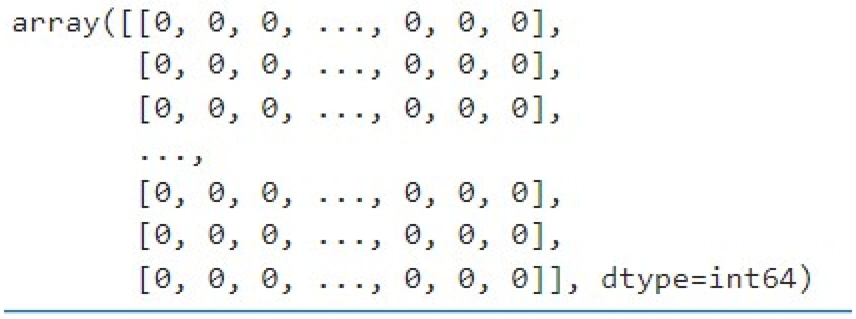
To fit the model, we use the transformed data from the Count Vectorizer. The model is trained to learn the patterns and relationships between the article titles and contents, represented as numerical vectors. This training enables the model to calculate cosine similarity scores for each pair of articles in the dataset. These similarity scores are then used to recommend similar articles to users based on their interests.

```
vectors_en=cv.fit_transform(df_en["title"]).toarray()
vectors=cv.fit_transform(df["heading"]).toarray()
```

In this code snippet, is using the Count Vectorizer object cv to transform the text data from the "title" column of the Data Frame df_en into a numerical matrix. This transformation converts each title into a vector of word counts, where each element in the vector represents the count of a unique word in the title. The result is a sparse matrix, where each row corresponds to a title and each column corresponds to a unique word in the corpus. This matrix is then converted to a dense array using toarray() to make it easier to work with and visualize.

```
vectors_en.shape
(2211, 4800)
```

The shape of the vectors_en array, (2211, 4800), indicates that there are 2211 samples (rows) and 4800 features (columns) in the array. Each sample represents a title from the English article's dataset, and each feature represents a unique word in the corpus. This shape is the result of applying the Count Vectorizer transformation to the title data, where each title is converted into a numerical vector based on the frequency of each word in the title.



```
array([[0, 0, 0, ..., 0, 0, 0],
       [0, 0, 0, ..., 0, 0, 0],
       [0, 0, 0, ..., 0, 0, 0],
       ...,
       [0, 0, 0, ..., 0, 0, 0],
       [0, 0, 0, ..., 0, 0, 0],
       [0, 0, 0, ..., 0, 0, 0]], dtype=int64)
```

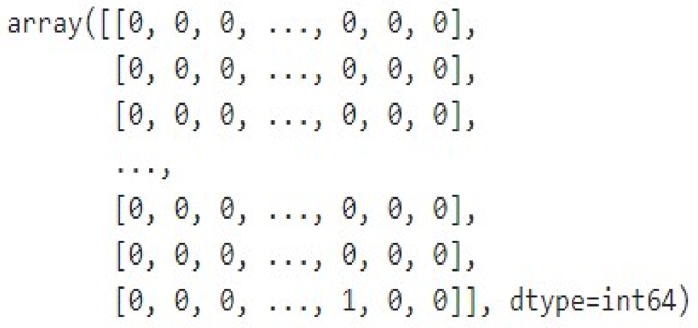
Fig:21 vectors of general english articles

vectors_en is a 2D array containing the numerical representations of the titles from the English article's dataset. Each row represents a title, and each column represents the count of a specific word in that title. This array is used for training machine learning models to recommend similar articles based on their titles.

vectors.shape

(7617, 5000)

vectors.shape indicates that the numerical vectors generated for the crime dataset have 7617 rows (titles of crime articles) and 5000 columns, representing the top 5000 most frequent words in the titles of crime articles. Each value in the array represents the count of a specific word in a title, allowing for the comparison and recommendation of similar crime articles based on their titles.



```
array([[0, 0, 0, ..., 0, 0, 0],
       [0, 0, 0, ..., 0, 0, 0],
       [0, 0, 0, ..., 0, 0, 0],
       ...,
       [0, 0, 0, ..., 0, 0, 0],
       [0, 0, 0, ..., 0, 0, 0],
       [0, 0, 0, ..., 1, 0, 0]], dtype=int64)
```

Fig:22 vector of crime articles

Vectors are a NumPy array representing the transformed numerical vectors for the crime dataset. Each row corresponds to the title of a crime article, and each column represents a unique word in the corpus. The values in the array indicate the count of each word in the corresponding article title. These numerical vectors are essential for calculating the cosine similarity between article titles, enabling the recommendation of similar crime articles based on their titles.

In conclusion, selecting key features, transforming text data with Count Vectorizer, and serializing the trained model with pickle are crucial steps in building an effective article recommendation system. These steps enable us to convert text data into a format suitable for machine learning, train the model to understand relationships between articles, and save the trained model for future use. Additionally, these steps help in providing personalized recommendations to users, enhancing their experience and engagement with the platform.

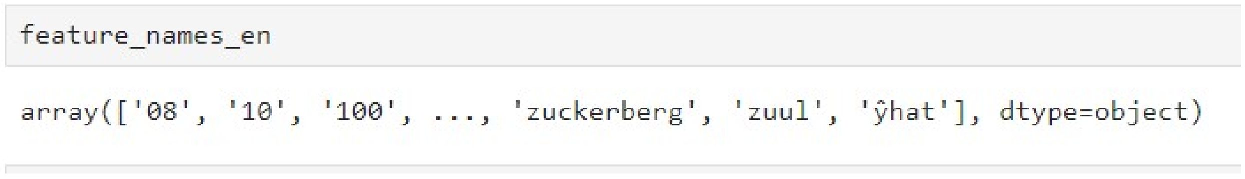
Extracting Feature Names for Content Understanding and Recommendation:

Extracting feature names from the CountVectorizer transformation is essential for understanding the underlying structure and content of the text data. These feature names represent the unique words present in the corpus of articles and serve as the basis for calculating the similarity between articles. By obtaining the feature names, we can identify the key words and topics that are important for each article, enabling us to make more informed recommendations based on the content similarities between articles. In the context of our recommendation system, the feature names help us understand the main themes and subjects of the articles, allowing us to recommend articles that are closely related in topic and content.

```
# Finding most commonly used 5000 words in our corpus

feature_names_en = cv.get_feature_names_out()
```

In the general article's dataset, we extracted the most commonly used 5000 words from the corpus using the Count Vectorizer. The feature_names_en array contains these words, representing the vocabulary learned from the text data.



```
feature_names_en

array(['08', '10', '100', ..., 'zuckerberg', 'zuul', 'ŷhat'], dtype=object)
```

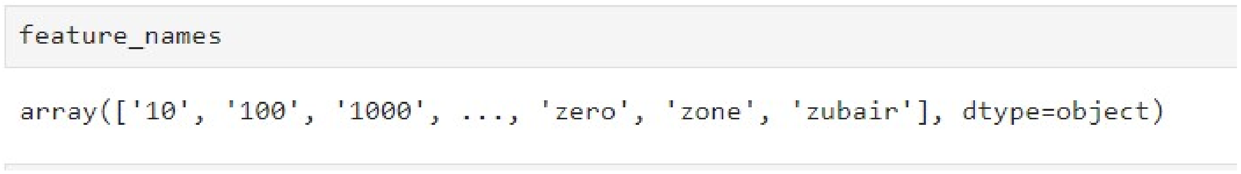
Fig:23 feature names in general article

Each word in the array is a unique term found in the corpus, ranging from numerical values like '08' and '100' to more descriptive terms like 'zuckerberg' and 'ŷhat'. These words are essential for understanding the content and themes present in the articles. By extracting these words, we create a foundational vocabulary that allows the model to learn the language patterns and frequencies in the dataset, aiding in the recommendation process.

```
# Finding most commonly used 5000 words in our corpus

feature_names = cv.get_feature_names_out()
```

In the crime articles dataset, the feature_names array contains the most commonly used 5000 words in the corpus



```
feature_names

array(['10', '100', '1000', ..., 'zero', 'zone', 'zubair'], dtype=object)
```

Fig:24 feature names in general article

The array includes terms such as '10', '100', '1000', 'zero', 'zone', 'zubair', and many others. These words represent the vocabulary learned from the crime articles and are essential for understanding the content and themes present in the dataset. By extracting these words, we create a foundational vocabulary that allows the model to learn the language patterns and frequencies in the crime articles, which is crucial for recommending similar articles based on their textual content.

Calculating Cosine Similarity:

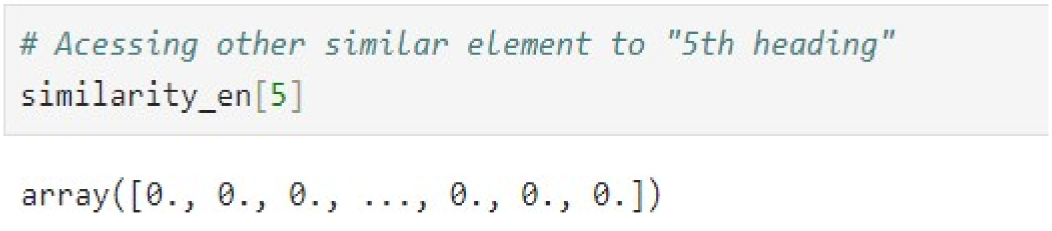
Cosine similarity is a metric used to measure the similarity between two vectors in an inner product space. In the context of our project, we use cosine similarity to calculate the similarity between the vectors representing the articles in our dataset. The cosine similarity score is calculated based on the cosine of the angle between two vectors, where a score of 1 indicates that the vectors are identical, a score of 0 indicates that the vectors are orthogonal (i.e., completely unrelated), and a score of -1 indicates that the vectors are diametrically opposed.

In our implementation, we use cosine similarity to calculate the similarity between the vectors representing the article titles. This allows us to determine how similar two articles are based on the words they contain. By calculating the cosine similarity scores for each pair of articles in our dataset, we can identify which articles are most similar to each other. This information is crucial for building a recommendation system, as it allows us to recommend articles that are most relevant to a given article based on their textual content.

```
similarity_en=cosine_similarity(vectors_en)
```

This code calculates the cosine similarity between each pair of articles in the general article's dataset based on their titles. Cosine similarity is a metric used to measure the similarity between two vectors, in this case, the vectors representing the titles of the articles.

The resulting similarity_en matrix contains pairwise similarity scores, where each element (i, j) represents the similarity score between the title of the ith article and the title of the jth article.



```
# Acessing other similar element to "5th heading"
similarity_en[5]
```

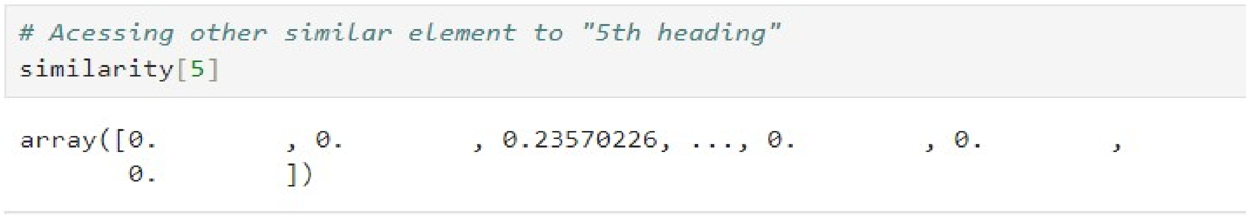
```
array([0., 0., 0., ..., 0., 0., 0.])
```

Fig:25 cosine similarity calculation for the 5th heading in the general article dataset

Therefore, by calculating the cosine similarity for the 5th heading, we can determine which other headings in the dataset are most similar to it based on the words they contain. This information can be used to recommend articles with similar headings to users who are interested in the topic of the 5th heading.

similarity=cosine_similarity(vectors)

This code calculates the cosine similarity between each pair of articles in the crime article's dataset based on their titles.



```
# Acessing other similar element to "5th heading"
similarity[5]
```

```
array([0.        , 0.        , 0.23570226, ..., 0.        , 0.        ,
       0.        ])
```

Fig:26 cosine similarity calculation for the 5th heading in the general article dataset

For the 5th heading, the array represents the cosine similarity scores between the 5th heading and all other headings in the dataset. Each element in the array corresponds to a specific heading, with the index of the array indicating the heading's position in the dataset.These scores can be used to identify articles with headings that are most similar to the 5th heading, helping to recommend relevant articles to users.

Recommendation Function:

```
def recommend(tag):
  if not df[df["heading"]==tag].empty:
    try:
      heading_index=df[df["heading"]==tag].index[0]
      distances=similarity[heading_index]
      heading_list=sorted(list(enumerate(distances)),reverse=True,key=lambda x:x[1])[1:6]

      for i in heading_list:
        print(df.iloc[i[0]].heading)
    except IndexError:
      pass
  else:
    if not df_en[df_en["title"]==tag].empty:
      try:
        heading_index = df_en[df_en["title"]==tag].index[0]
        distances = similarity_en[heading_index]
        heading_list = sorted(list(enumerate(distances)), reverse=True, key=lambda x: x[1])[1:6]

        for i in heading_list:
          print(df_en.iloc[i[0]].title)
      except IndexError:
        Pass
```

The recommendation function is designed to provide article recommendations based on a given input tag, which can be either a crime article heading or a general article title. The function first checks if the input tag exists in either the crime articles dataset (df) or the general articles dataset (df_en). If the tag is found in either dataset, the function calculates the cosine similarity between the input tag and all other tags in the respective dataset. The function then identifies the most similar tags and returns the top 5 recommendations.

The function takes a single parameter, tag, which represents the input article heading or title for which recommendations are to be generated. The function checks if the input tag exists in either dataset and proceeds to find similar articles based on cosine similarity scores.

Finding Similar Articles

For crime article tags, the function first identifies the index of the input tag in the crime articles dataset (df). It then retrieves the cosine similarity scores between the input tag and all other tags in the dataset. The function sorts these scores in descending order and selects the top 5 most similar articles, excluding the input tag itself.

For general article titles, the function follows a similar process, identifying the index of the input title in the general articles dataset (df_en) and calculating cosine similarity scores with all other titles. Again, the function selects the top 5 most similar articles based on these scores.

Output

The function outputs the top 5 recommended article headings (for crime articles) or titles (for general articles) that are most similar to the input tag. These recommendations can be used to suggest relevant articles to users based on their interests or search queries.

User Interface

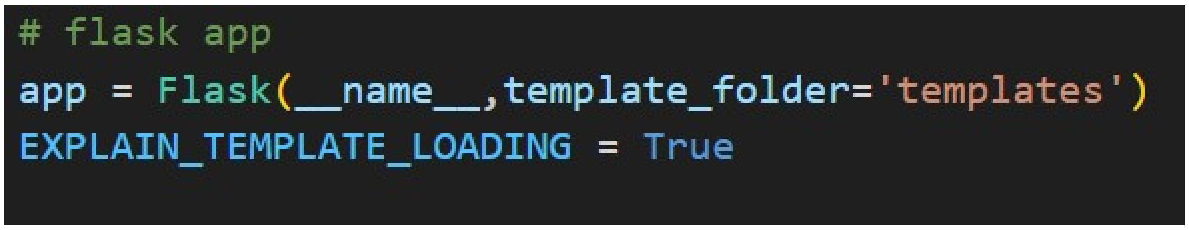
Now we built the model and it is giving accurate values for user input we need to make a user interface for the users to make it more interactive with the user because people can't understand how the model works we need to create such that the user needs to give an input when the given input is valid it shoes the predicted values and display it in the user interface or if the entered input is not valid then I should not show any result rather show the same page.

To create the user interface we will be using HTML, CSS, BOOTSTRAP, and FLASK. Flask is a microweb framework written in Python classified as a microframework because it does not require particular tools or libraries. It has no database abstraction layer

Integrating model with UI using Flask:

1. Import Flask and necessary libraries

   We need to import the Flask,render_template, request from Flask and numpy, pandas, and pickle
2. Creating Flask app



```
# flask app
app = Flask(__name__,template_folder='templates')
EXPLAIN_TEMPLATE_LOADING = True
```

We need to initialize an application for the flask to communicate with the HTML files and how we do this is Templates folder where we store the HTML files and give them to template_folder such that we don’t need to give a path for rendering the HTML file instead we can just give the name of the HTML file

3. Load dataset

   We need to read the datasets using pandas

4. Load Model

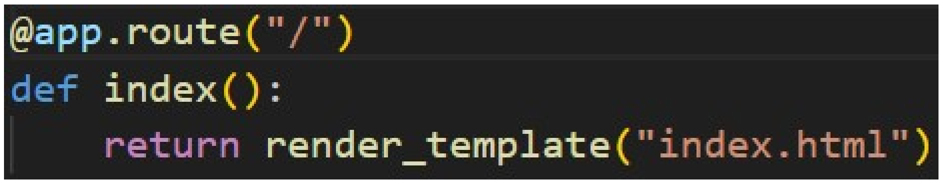
   Using pickle we need to load the model as we observed in the previous implementation section by using the load function

   vc = pickle.load(open('vc.pkl','rb'))

5. Using helper function and model prediction function are used in the model creation and the same needs to be used in the app.py file

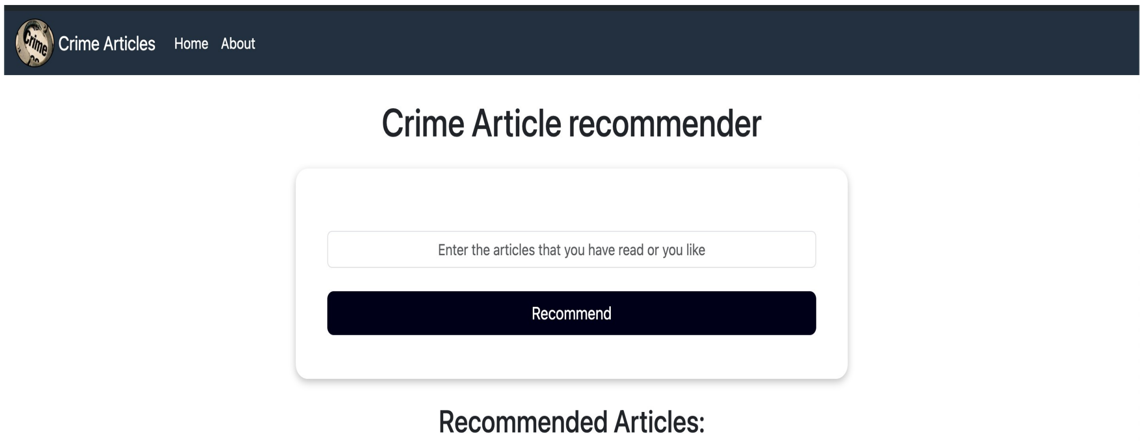
6. Creating routes

   For each route we can render a new HTML template for example



```
@app.route("/")
def index():
    return render_template("index.html")
```

Output:



Crime Articles Home About

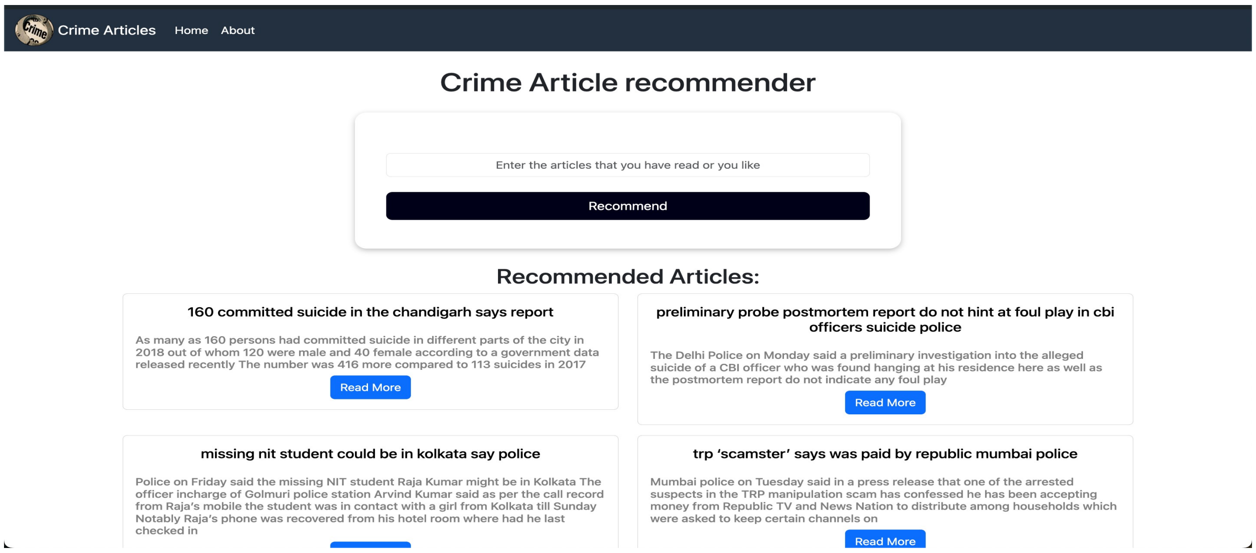
Crime Article recommender

Enter the articles that you have read or you like

Recommend

Recommended Articles:

UI, when a user enters input and presses, enter



Crime Articles Home About

Crime Article recommender

Enter the articles that you have read or you like

Recommend

Recommended Articles:

160 committed suicide in the chandigarh says report

As many as 160 persons had committed suicide in different parts of the city in 2018 out of whom 120 were male and 40 female according to a government data released recently The number was 416 more compared to 113 suicides in 2017

Read More

preliminary probe postmortem report do not hint at foul play in cbi officers suicide police

The Delhi Police on Monday said a preliminary investigation into the alleged suicide of a CBI officer who was found hanging at his residence here as well as the postmortem report do not indicate any foul play

Read More

missing nit student could be in kolkata say police

Police on Friday said the missing NIT student Raja Kumar might be in Kolkata The officer incharge of Golmuri police station Arvind Kumar said as per the call record from Raja's mobile the student was in contact with a girl from Kolkata till Sunday Notably Raja's phone was recovered from his hotel room where had he last checked in

trp 'scamster' says was paid by republic mumbai police

Mumbai police on Tuesday said in a press release that one of the arrested suspects in the TRP manipulation scam has confessed he has been accepting money from Republic TV and News Nation to distribute among households which were asked to keep certain channels on

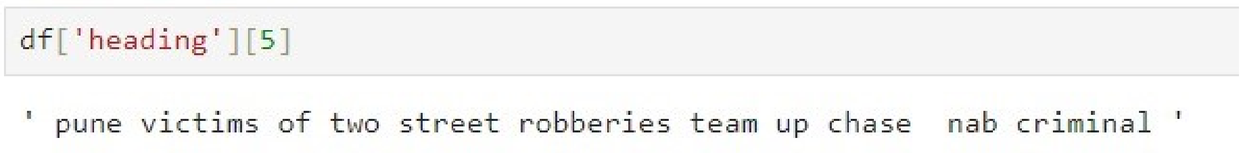
Read More

When read more is clicked then we get redirected to the link of the article.

6. Results and Discussion:

We need to read other datasets using pandas and implement the logic for the predictions by taking user input which is comma-separated text. We created a helper function and a model prediction function.

The entered user input is first sent to the prediction function and it returns the disease that the model predicted the predicted disease will be sent to the helper function and the helper functions will return the Articles with their title, summary and their link to their full article.

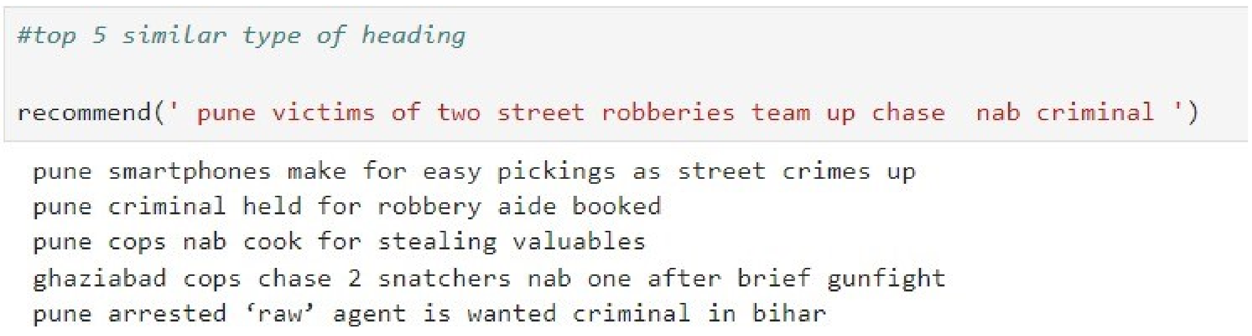


```
df['heading'][5]
```

```
' pune victims of two street robberies team up chase  nab criminal '
```

This is an article in our data set which is an input to our recommendation system.

Now we are going to recommend articles similar to this one using the cosine similarity for all the other data points in our dataset.

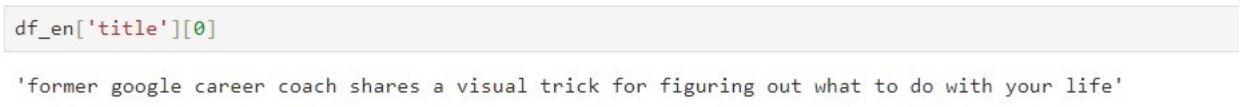


```
#top 5 similar type of heading

recommend(' pune victims of two street robberies team up chase  nab criminal ')
```

```
pune smartphones make for easy pickings as street crimes up
pune criminal held for robbery aide booked
pune cops nab cook for stealing valuables
ghaziabad cops chase 2 snatchers nab one after brief gunfight
pune arrested ‘raw’ agent is wanted criminal in bihar
```

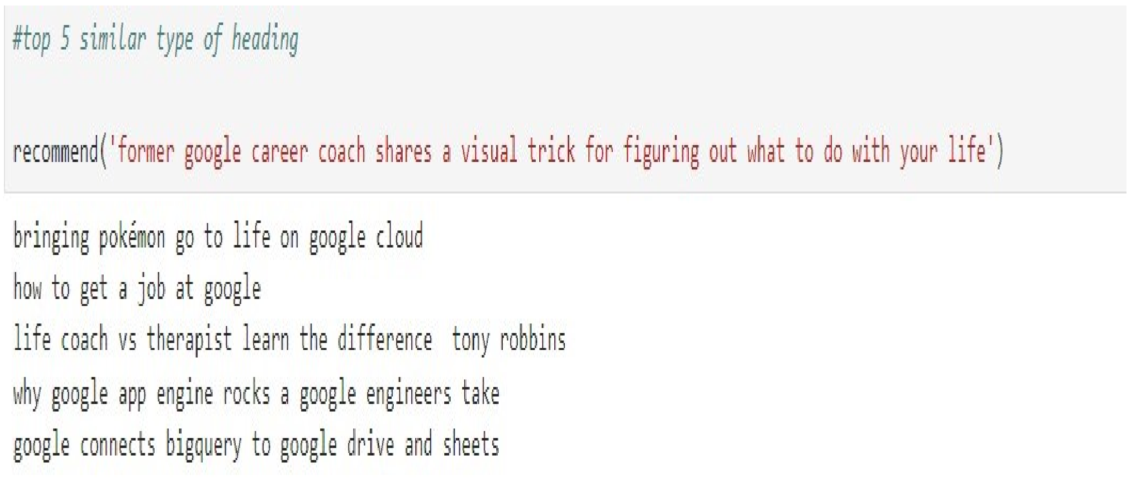
These are the top 5 recommended articles similar to the input.



```
df_en['title'][0]
```

```
'former google career coach shares a visual trick for figuring out what to do with your life'
```

This is the results:



```
#top 5 similar type of heading


recommend('former google career coach shares a visual trick for figuring out what to do with your life')
```

```
bringing pokémon go to life on google cloud
how to get a job at google
life coach vs therapist learn the difference  tony robbins
why google app engine rocks a google engineers take
google connects bigquery to google drive and sheets
```

These are the 2 results based on the input given based on the input article given by the user.

7.Conclusion

The articles recommendation system, designed to facilitate access to crime-related content, has proven effective in aligning recommendations with user interests, demonstrating robust capabilities in understanding and responding to user queries. The adaptability shown by the system is crucial for maintaining user engagement and ensuring the relevance of the content provided. Its architecture, built on a Flask backend and utilizing machine learning algorithms for recommendations, is both flexible and scalable. This scalability is essential as it permits the system to handle an expanding dataset and a growing user base without sacrificing performance.

The system's design allows for seamless upgrades and integrations, accommodating an increasing number of users and a larger data volume. This flexibility is particularly vital as it supports the continuous improvement of system functionalities, which is critical to sustaining user satisfaction and system utility over time.

The success of a recommendation system relies on continuous monitoring, evaluation, and optimization to improve recommendation quality and user satisfaction. It also requires integration with business logic and rules to align recommendations with business goals and constraints.

8.Future Work

Future work will focus on optimizing the system for larger datasets, improving the handling of complex queries, and continuously enhancing the user interface. The potential integration of machine learning models for better text understanding and context extraction is also being considered to enhance the accuracy of recommendations.

Overall, the implementation of this recommendation system offers significant benefits in terms of user engagement and satisfaction, positioning it as a valuable tool for readers and researchers interested in related articles based on the input of the user.

9.Reference

Recommendation Systems: Ricci, F., Rokach, L., Shapira, B., & Kantor, P.B. (2015). Recommender Systems Handbook. Springer.

Adomavicius, G., & Tuzhilin, A. (2005). Toward the Next Generation of Recommender Systems: A Survey of the State-of-the-Art and Possible Extensions. IEEE Transactions on Knowledge and Data Engineering, 17(6), 734-749.

S. Bahulikar, "Analyzing recommender systems and applying a location based approach using tagging", *2017 2nd International Conference for Convergence in Technology (I2CT)*, 2017.

Recommendation Systems: Ricci, F., Rokach, L., Shapira, B., & Kantor, P.B. (2015). Recommender Systems Handbook. Springer.https://link.springer.com/article/10.1007/s10115-023-01901-x

Flask Documentation https://flask.palletsprojects.com/en/3.0.x/