MACHINE LEARNING II

Project Report on

News Article Recommendator

Submitted by

ADITYA

Registration No : 12217241

Programme Name : Btech. CSE (3rd Year)

SECTION KM008

COURSECODE: INT - 423

SCHOOL OF COMPUTER SCIENCE AND ENGINEERING

Lovely Professional University, Phagwara

DECLARATION

I hereby declare that the project work entitled “News Article Recommendator”
is an authentic record of my own work carried out as requirements of Project for the award of B. Tech degree in Computer Science and Engineering from Lovely Professional University, Phagwara, under the guidance of Dr. Jimmy Singla,during August to December 2024. All the information furnished in this project report is based on my own intensive work and is genuine.

Name of Student –

Aditya

Registration no : 12217241

ACKNOWLEDGEMENT

It is with my immense gratitude that I acknowledge the support and help of my Professor, Dr. Jimmy Singla, who has always encouraged me into this research. Without his continuous guidance and persistent help, this project would not have been a success for me. I am grateful to the Lovely Professional University, Punjab and the department of Computer Science without which this project would not have been an achievement. I also thank my family and friends, for their endless love and support throughout my life.

ABSTRACT

In the era of information overload, recommending relevant news articles has become essential for enhancing user engagement and satisfaction. This project presents a news article recommendation engine that leverages machine learning techniques to deliver personalized article suggestions based on user interests and reading history. The model analyzes content features, such as headlines and article categories, along with user interaction data, to generate recommendations tailored to individual preferences. Various recommendation algorithms, including content-based filtering and collaborative filtering, are explored to evaluate their effectiveness in improving recommendation accuracy. This report provides a detailed overview of the system's architecture, data processing techniques, model training and evaluation, and results from implementing the recommendation engine.

INTRODUCTION

With the rapid expansion of digital content, users face an overwhelming volume of news articles daily. This abundance of information, though beneficial, often leads to users missing relevant articles that align with their interests. Recommendation engines have emerged as a solution, helping users navigate content by suggesting articles that are likely to be of interest. This project focuses on developing a recommendation engine specifically for news articles, aiming to enhance user experience by providing personalized article recommendations.

The foundation of a successful recommendation engine lies in its ability to understand both the characteristics of articles and the preferences of users. In this project, we utilize two primary sources of data: the content of articles (including headlines, categories, and k eywords) and users' reading history. By combining these elements, the system learns patterns in user behavior, enabling it to recommend articles that align with their interests.

To achieve this, we employ various machine learning techniques, including content-based filtering, which recommends articles based on similarity to previously read content, and collaborative filtering, which considers the reading patterns of similar users. Hybrid approaches are also explored to enhance the accuracy and relevance of recommendations. This report details the dataset selection, feature engineering process, algorithm choice, model training, and evaluation metrics used to assess the system's effectiveness. The findings offer insights into the model's performance and the challenges of creating effective recommendations in the dynamic context of news articles.

Methodology

1. Importing Libraries

The analysis leverages Python libraries for data manipulation, visualization, text processing, and clustering, including libraries such as pandas, numpy, nltk, sklearn, and plotly.

2. Data Loading

The dataset comprises approximately two million records with six features, including headline, date, category, and author.

3. Data Preprocessing

Given the large size of the dataset, we selectively filter and clean the data for efficient processing:

3.1 Filtering Recent Articles

To streamline processing, only articles from the year 2018 are considered.

3.2 Removing Short Headlines

Headlines with fewer than five words are excluded to ensure headline quality and relevance, particularly after stop word removal.

3.3 Duplicate Removal

To avoid redundancy, articles with duplicate headlines are removed.

3.4 Missing Value Check

Missing values across all features are checked and handled appropriately.

4. Data Exploration

To gain initial insights, we conduct basic exploratory analysis on the filtered data:

4.1 Summary Statistics

We compute the number of unique articles, authors, and categories.

4.2 Category Distribution

A bar chart is plotted to visualize the distribution of articles across categories, with the "Politics" category appearing most frequently.

4.3 Monthly Article Distribution

The data is grouped by month to observe article distribution trends over time.

4.4 Headline Length Distribution

We analyze the probability distribution of headline length, which follows an approximately Gaussian distribution, with most headlines containing between 58 and 80 characters.

5. Text Preprocessing

Text preprocessing transforms headlines into a suitable format for similarity assessment:

5.1 Stop Word Removal

Stop words are removed to reduce noise and improve computational efficiency.

5.2 Lemmatization

Lemmatization is applied to reduce words to their base forms, enabling uniformity across different word inflections.

6. Headline-Based Similarity Measures

Headline similarity is evaluated through various text representation techniques. Lower distances between headline vectors indicate higher similarity.

6.1 Bag of Words Representation

Each headline is converted into a vector using the Bag of Words (BoW) model, where each word in the vocabulary represents a dimension.

6.2 TF-IDF Representation

TF-IDF weights each term based on its frequency within a document and across the corpus, prioritizing less frequent yet significant words. The TF-IDF formula is as follows:

weight(i,j)=TF(i,j)×IDF(i,D)$\text{weight}(i,j)=\text{TF}(i,j)\times\text{IDF}(i,D)$

where TF(i, j) measures the term frequency, and IDF(i, D) adjusts for the term's rarity in the corpus.

6.3 Word2Vec Embedding

For semantic similarity, we use Google's pre-trained Word2Vec model. This model represents each word as a 300-dimensional vector, and each headline vector is obtained by averaging the vectors of its constituent words.

7. Weighted Similarity Models

To refine recommendations, similarity measures incorporate headline, category, author, and publication date with variable weighting.

7.1 Headline and Category

Similarity is calculated based on headline and category features. Categories are one-hot encoded, and weights can be adjusted to prioritize articles within the same category.

7.2 Headline, Category, and Author

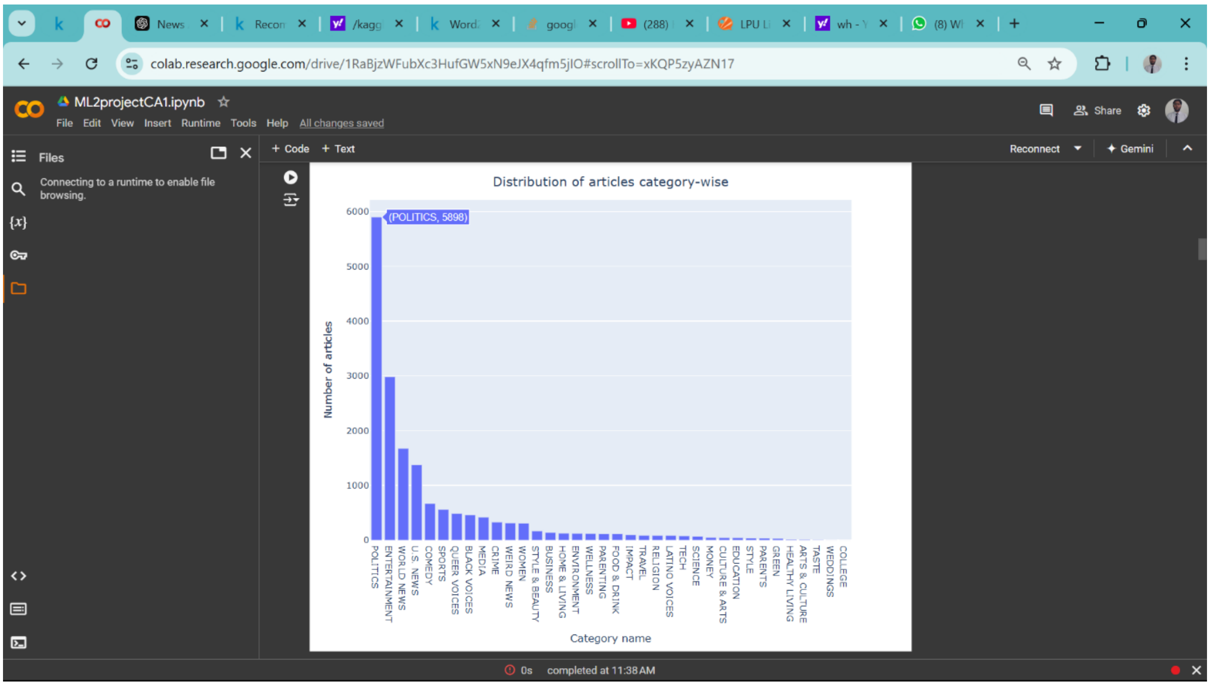
We extend similarity measures by including the author feature, also one-hot encoded, to recommend articles by the same author more accurately.

7.3 Headline, Category, Author, and Publishing Day

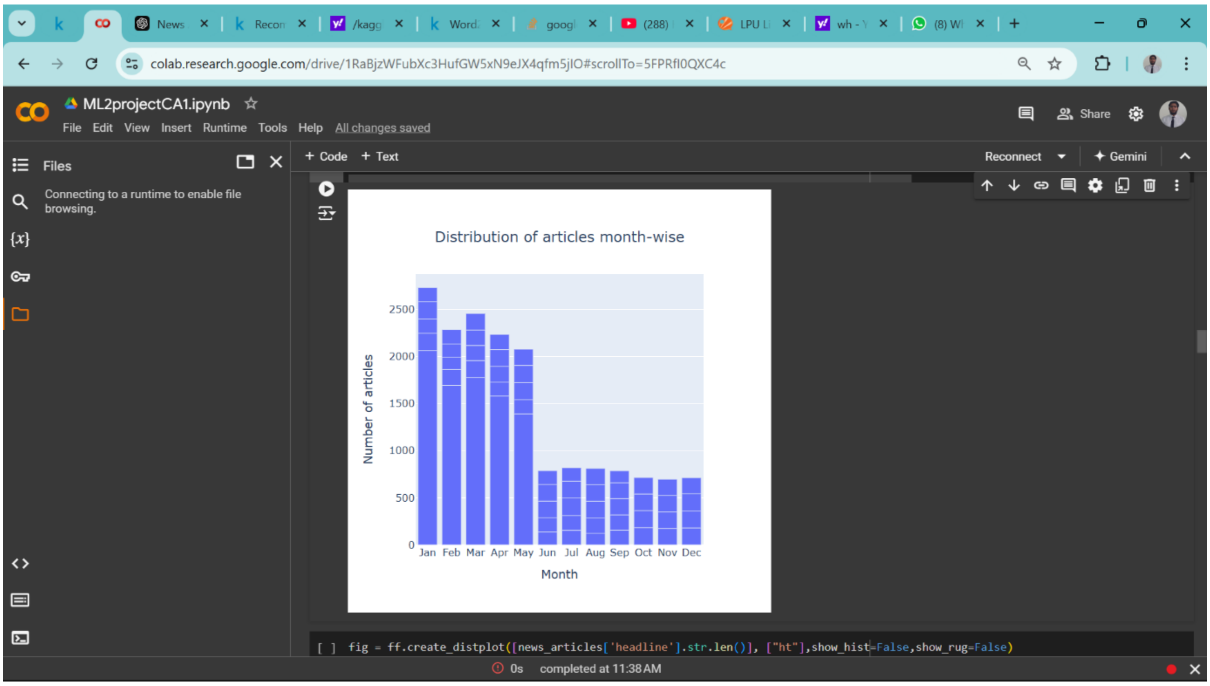
The model further incorporates the day of the week as a feature, one-hot encoded and weighted to emphasize articles published on the same weekday, author, and category.

Now let's see calcualte articles similarity based on the publishing **week day author** along with **headline**, **category** and **author**. Again, we are encoding this new feature through **onehot encoding**.

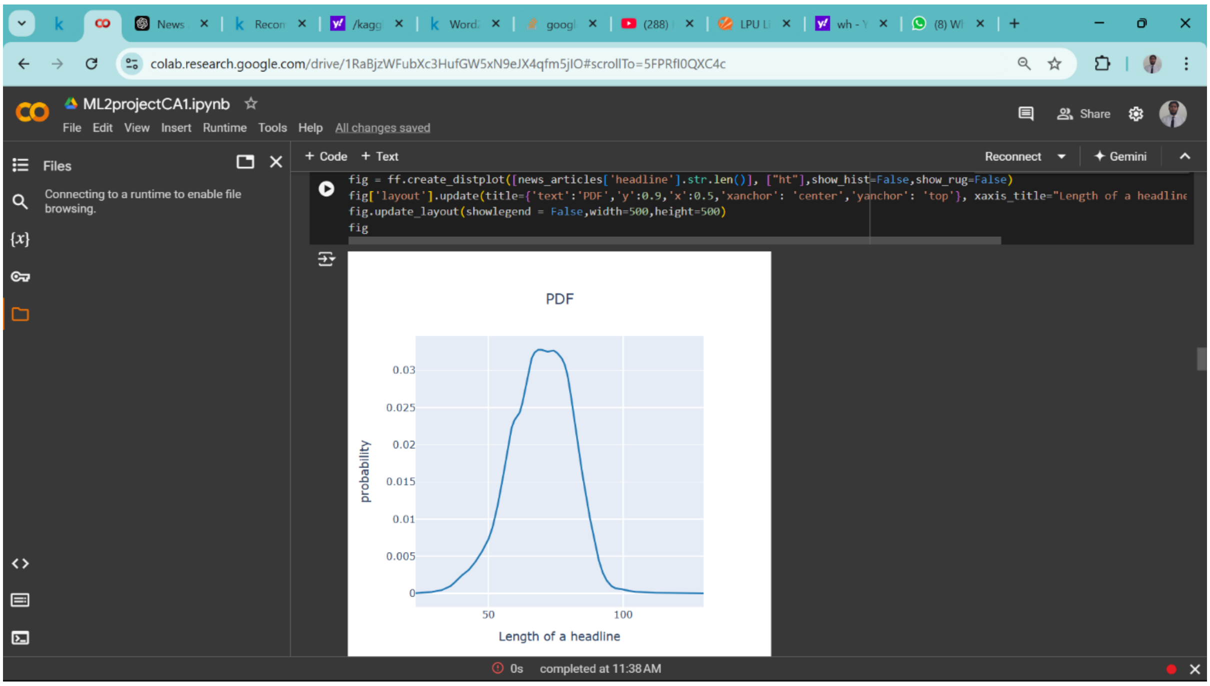
RESULTS :



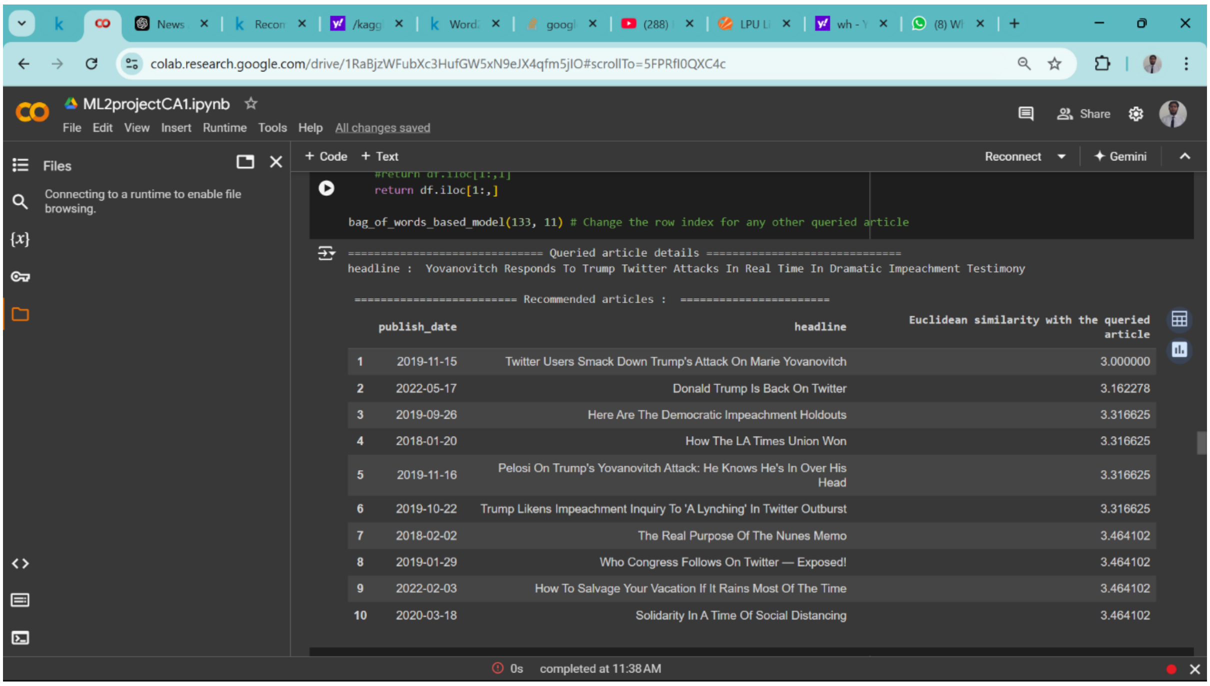
From the bar chart, we can observe that **politics** category has **highest** number of articles then **entertainment** and so on.



From the bar chart, we can observe that **January** month has **highest** number of articles then **March** and so on.



The probability distribution function of headline length is almost similar to a **Guassian distribution**, where most of the headlines are 58 to 80 words long in length.



Above function recommends **10 similar** articles to the **queried**(read) article based on the headline. It accepts two arguments - index of already read artile and the total number of articles to be recommended.

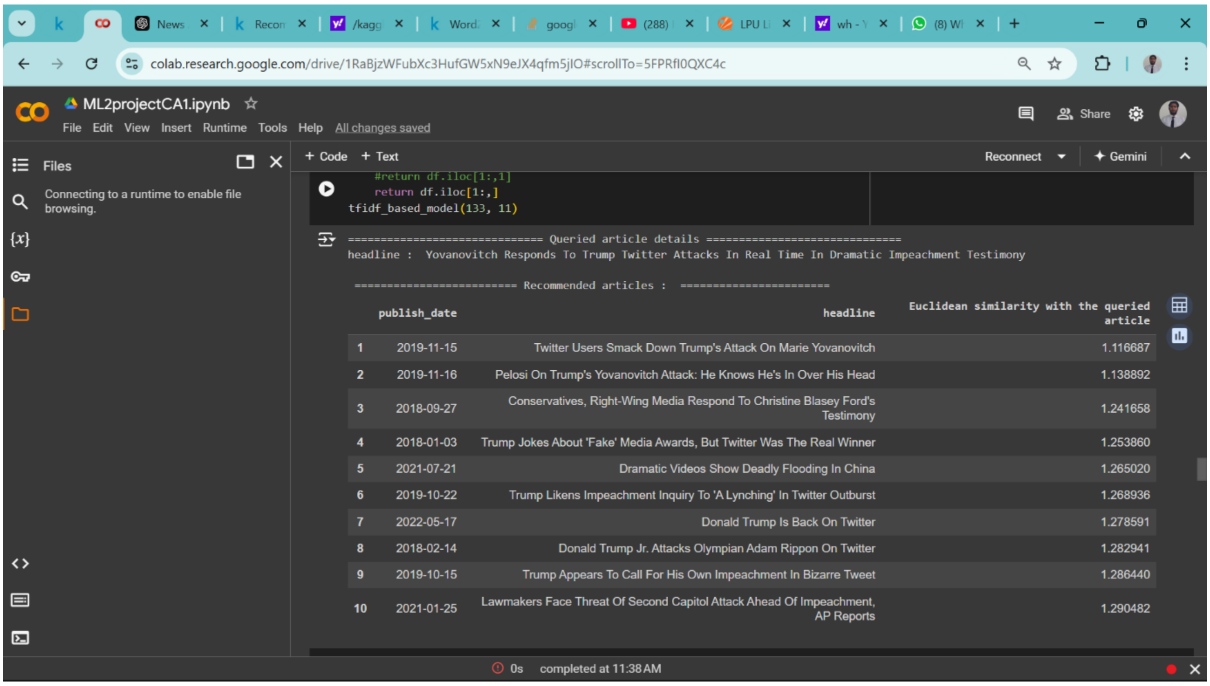
Based on the **Euclidean distance** it finds out 10 nearest neighbors and recommends.

Disadvantages

1. It gives very low **importance** to less frequently observed words in the corpus. Few words from the queried article like "employer", "flip", "fire" appear less frequently in the entire corpus so **BoW** method does not recommend any article whose headline contains these words. Since **trump** is commonly observed word in the corpus so it is recommending the articles with headline containing "trump".

2. **BoW** method doesn't preserve the order of words.

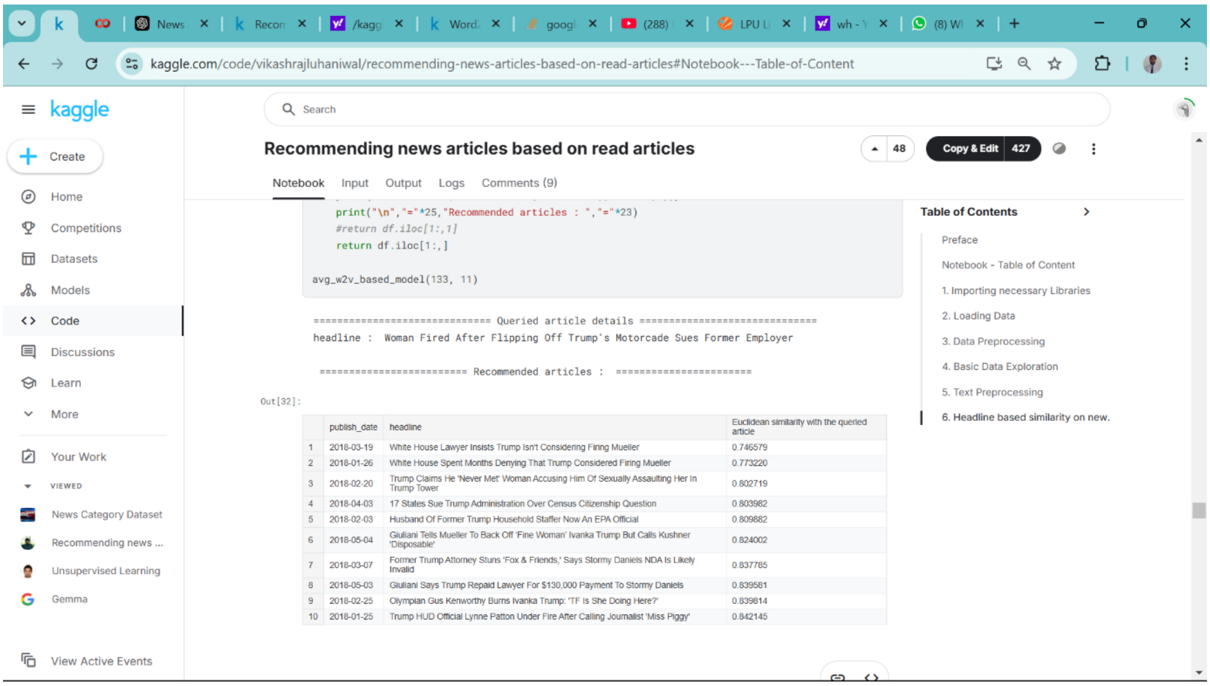
To overcome the first disadvantage we use **TF-IDF** method for feature representation.



Disadvantages :-

Bow and TF-IDF method do not capture semantic and syntactic similarity of a given word with other words but this can be captured using Word embeddings.

For example: there is a good association between words like "trump" and "white house", "office and employee", "tiger" and "leopard", "USA" and "Washington D.C" etc. Such kind of semantic similarity can be captured using word embedding techniques. Word embedding techniques like Word2Vec, GloVe and fastText leverage semantic similarity between words.




```
    print("\n","="*25,"Recommended articles : ","="*23)
    #return df.iloc[1:,1]
    return df.iloc[1:,]

avg_w2v_based_model(133, 11)
```

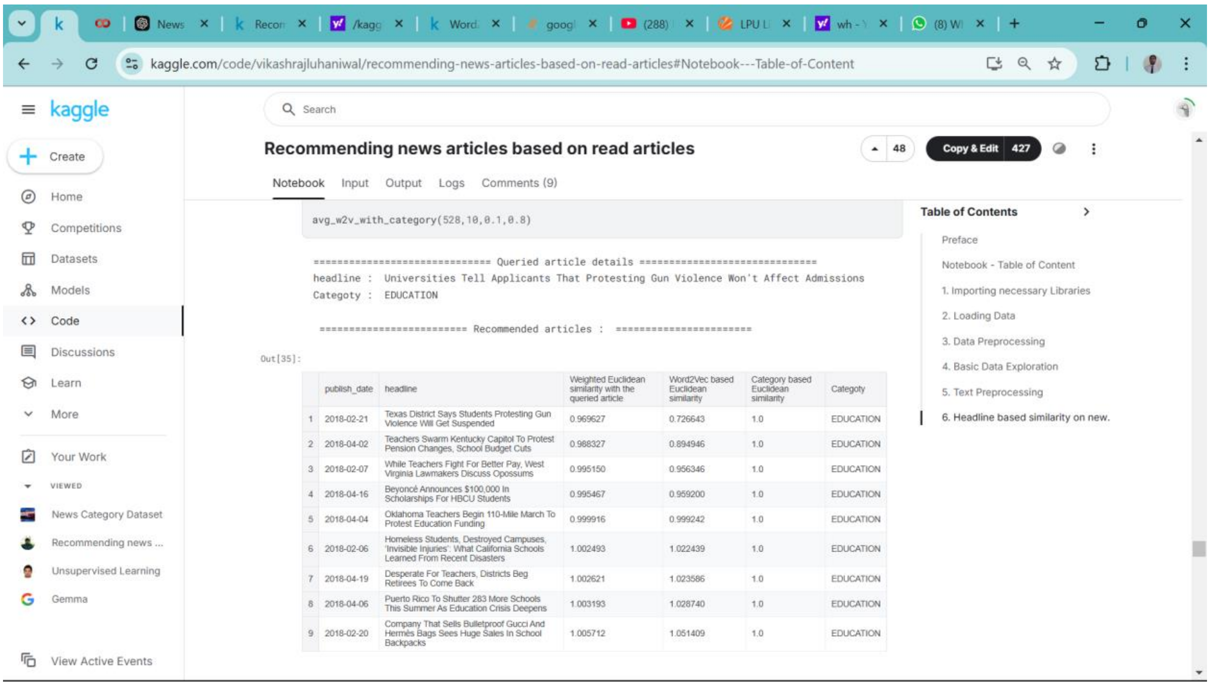
```
============================== Queried article details ==============================
headline :  Woman Fired After Flipping Off Trump's Motorcade Sues Former Employer

 ========================= Recommended articles :  =======================
```

Out[32]:

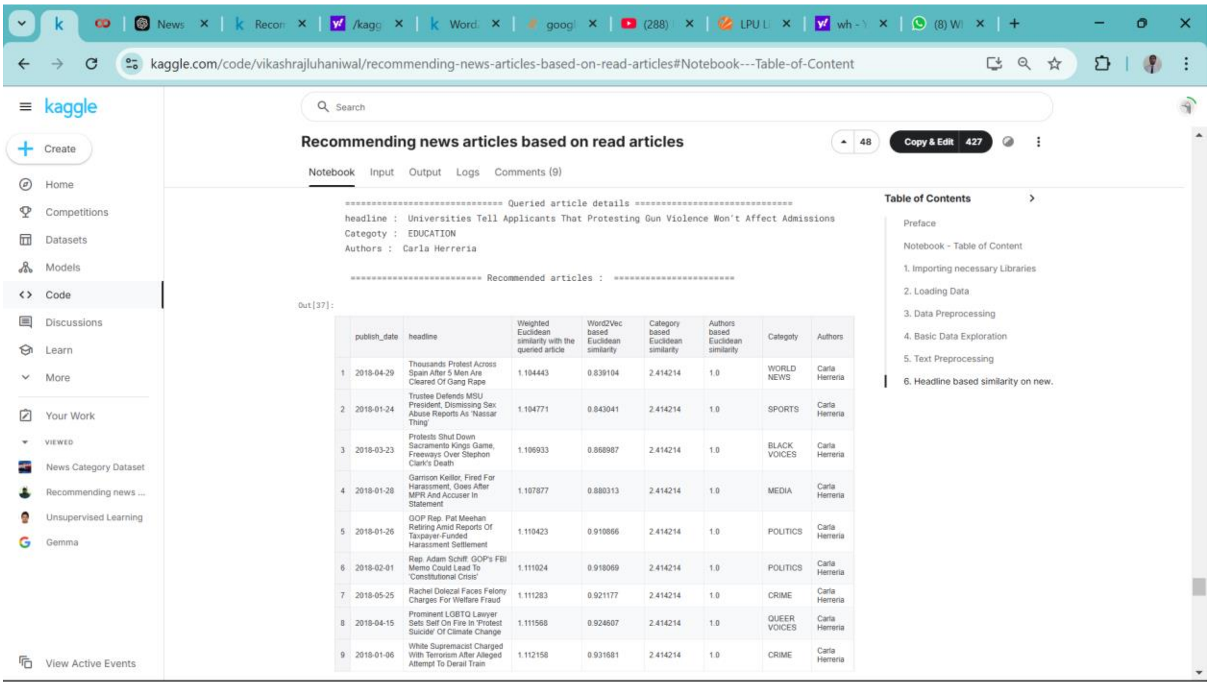
	publish_date	headline	Euclidean similarity with the queried article
1	2018-03-19	White House Lawyer Insists Trump Isn't Considering Firing Mueller	0.746579
2	2018-01-26	White House Spent Months Denying That Trump Considered Firing Mueller	0.773220
3	2018-02-20	Trump Claims He 'Never Met' Woman Accusing Him Of Sexually Assaulting Her In Trump Tower	0.802719
4	2018-04-03	17 States Sue Trump Administration Over Census Citizenship Question	0.803982
5	2018-02-03	Husband Of Former Trump Household Staffer Now An EPA Official	0.809882
6	2018-05-04	Giuliani Tells Mueller To Back Off 'Fine Woman' Ivanka Trump But Calls Kushner 'Disposable'	0.824002
7	2018-03-07	Former Trump Attorney Stuns 'Fox & Friends,' Says Stormy Daniels NDA Is Likely Invalid	0.837785
8	2018-05-03	Giuliani Says Trump Repaid Lawyer For $130,000 Payment To Stormy Daniels	0.839581
9	2018-02-25	Olympian Gus Kenworthy Burns Ivanka Trump: 'TF Is She Doing Here?'	0.839814
10	2018-01-25	Trump HUD Official Lynne Patton Under Fire After Calling Journalist 'Miss Piggy'	0.842145

Here, Word2Vec based representation recommends the headlines containing the word white house which is associated with the word trump in the queried article. Similarly, it recommends the headlines with words like "offical", "insist" which have semantic similarity to the words "employer", "sue" in the queried headline.

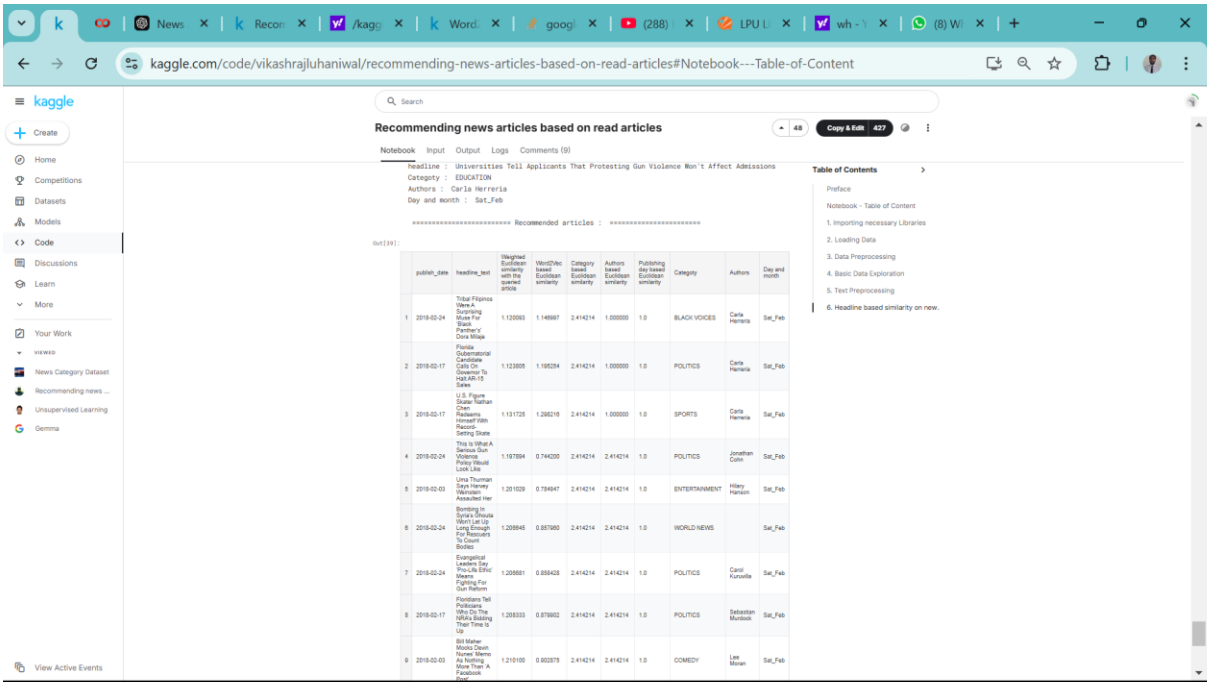
So far we were recommending using only one feature i.e. headline but in order to make a robust recommender system we need to consider multiple features at a time. Based on the business interest and rules, we can decide weight for each feature.
Let's see different models with combinations of different features for article similarity.



Above function takes two extra arguments **w1** and **w2** for weights corresponding to **headline** and **category**. It is always a good practice to pass the **weights** in a range scaled from **0 to 1**, where a value close to 1 indicates high weight whereas close to 0 indicates less weight.
Here, we can observe that the recommended articles are from the same **category** as the queried article **category**. This is due to passing of high value to **w2**.



Above function takes one extra weight argument **w3** for **author**.
In the ouput, we can observe that the recommended articles are from the same **author** as the queried article **author** due to high weightage to **w3**.



Above function takes one extra weight argument **w4** for **day of the week and month**.
In the ouput, we can observe that the recommended articles are from the same **day of the week and month** as the queried article due to high weightage to **w4**.

Conclusion

In this project, we developed a personalized recommendation system for news articles, aiming to improve user engagement by suggesting relevant articles based on individual interests. Through content-based and collaborative filtering methods, we examined various feature representations, including Bag of Words, TF-IDF, and Word2Vec, each contributing uniquely to understanding headline similarity. Additionally, we explored weighted similarity models incorporating multiple features, such as headline, category, author, and publication date, to refine recommendations and tailor them further to user preferences.

Our analysis demonstrated that the Word2Vec embedding technique, due to its ability to capture semantic similarity, produced superior results over the Bag of Words and TF-IDF methods, which do not account for contextual similarity. The weighted similarity models allowed for flexibility in prioritizing specific features, improving the model's relevance to users' reading behavior and preferences. Despite these advances, challenges remain in balancing feature importance and capturing evolving user interests, particularly in dynamic fields such as news.

The findings from this project underline the importance of combining multiple approaches in a recommendation engine to enhance its adaptability and effectiveness. Future improvements could include real-time personalization based on immediate user interactions and expanding the feature set to incorporate broader user data. By addressing these aspects, a more robust and responsive recommendation system can be developed to support user engagement in the ever-evolving landscape of digital news.

References

1. Jurafsky, D., & Martin, J. H. (2008). *Speech and Language Processing*. Pearson Prentice Hall.
2. Mikolov, T., Chen, K., Corrado, G., & Dean, J. (2013). Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.
3. Aggarwal, C. C. (2016). *Recommender Systems: The Textbook*. Springer.
4. Salton, G., Wong, A., & Yang, C. S. (1975). A vector space model for automatic indexing. *Communications of the ACM, 18*(11), 613–620.
5. Ramos, J. (2003). Using TF-IDF to determine word relevance in document queries. In *Proceedings of the First Instructional Conference on Machine Learning*.
6. Google News pre-trained Word2Vec model, available from Google Code Archive.
7. Lops, P., De Gemmis, M., & Semeraro, G. (2011). Content-based recommender systems: State of the art and trends. In *Recommender Systems Handbook* (pp. 73–105). Springer.
8. Koren, Y., Bell, R., & Volinsky, C. (2009). Matrix factorization techniques for recommender systems. *Computer, 42*(8), 30–37.
9. Lovely Professional University (2024). Course materials for INT-423 Machine Learning II, School of Computer Science and Engineering.