NARS: News Aggregation and Recommendation System

R. S. Akalya
18cs103@psgitech.ac.in

K. R. Dharani Dharan
18cs206@psgitech.ac.in

Dr. R. Manimegalai
drrm@psgitech.ac.in

Department of Computer Science Engineering
PSG Institute of Technology and Applied Research

***Abstract* - The emergence of the web and the social web has resulted in a massive increase of news sources. The ease of access to these news sources provides a flood of information which is frequently contradictory and confusing. It's tough to tell the difference between true and false news when it circulates on the social web. Fake news may be used for various objectives, including political influence, financial gain, and religious beliefs. Same news articles may also repeat on multiple different sites. In order to overcome these problems, a new aggregator is developed to help the users distinguish between valid, false and duplicate news. In the proposed system, news from multiple news sites is scraped, validated and presented to the user. Spam filtering is done in order to eliminate false and duplicate news. The news aggregator also suggests relevant news to the user by predicting user preference using recommendation system.**

Index Terms – News Aggregator, Web scraping, Recommendation, Spam filtering

I. INTRODUCTION

Users have a plethora of options because of the fast rise of internet content and services. For example, news aggregation services like Google News, one of the most prominent web services, may deliver an overwhelming quantity of material that consumers can comprehend. To improve the user experience, customized online content recommendations are required. Content-based approaches, collaborative filtering- based methods, and hybrid methods are among the ways proposed to tackle the online customized news recommendation problem.

Deep learning models have recently become the new state-of-the-art approaches as an extension and integration of prior methods due to their capacity to simulate complicated user item (i.e., news) interactions. In recommendation systems, accurate user interest modelling is a major issue. Existing news recommendation algorithms often learn user representations based on past news click patterns. The present project's goals are to recognize and analyze the issues in the news recommendation area to improve the user experience by eliminating barriers and lowering the time and effort required to manage and maintain numerous news sites and articles.

Online news services like Google News and Microsoft News can aggregate information from a variety of sources and present it to readers in a consistent format. However, a great number of news pieces are published every day, making it difficult for consumers to obtain the information they are looking for. As a result, personalized news recommendations are essential for these news services to target user interests and reduce information overload.

In recommendation systems, accurate user interest modelling is a major issue. Existing news recommendation algorithms often learn user representations based on past news click patterns.

Recommendation engines are a type of machine learning that is used to rank or rate products and users. A recommender system, in a broad sense, is a system that predicts how a user would rate a certain item. After then, the predictions will be ranked and returned to the user.

They're frequently utilised by big-name companies like Google, Instagram, Spotify, Amazon, Reddit, Netflix, and others to boost user and platform engagement. Spotify, for example, will suggest tracks that are similar to those you've listened to or enjoyed previously so that you can continue to use their platform to listen to music. Amazon uses recommendations to propose things to different consumers depending on the information they've gathered about them. Recommender systems are frequently regarded as a "black box," with the models developed by these major corporations being difficult to decipher. The generated results are frequently recommendations for the user for things that they need or want but aren't aware of until they've been recommended to them.

There are a variety of approaches to developing recommender systems; some employ algorithmic and formulaic approaches such as Page Rank, while others use more modeling-centric approaches such as collaborative filtering, content-based, link prediction, and so on. The complexity of each of these approaches varies, but complexity does not imply "excellent" performance. Simple solutions and implementations frequently produce the best results. Large firms such as Reddit, Hacker News, and Google, for example, have promoted content on their platforms using simple formulaic implementations of recommendation engines. I'll give an intuitive and technical description of the recommendation system design, as well as the execution of a few distinct versions on a sample generated dataset, in this article.

A. Collaborative filtering

Collaborative filtering is the practise of predicting a user's interests based on several users' preferences and information. This is accomplished by using strategies including

collaboration among various agents, data sources, and other data sources to filter data for information or patterns. The core premise of collaborative filtering is that if customers A and B have similar tastes in one product, they are likely to have similar tastes in other products.

There are two common types of approaches in collaborative filtering, memory based and model-based approach.

1. Memory based approaches

Neighborhood collaborative filtering is another name for them. Ratings of user-item pairs are essentially predicted based on their neighbourhoods. This can be further divided into two types of collaborative filtering: user-based collaborative filtering and item-based collaborative filtering. User-based simply means that strong and similar recommendations will come from like-minded people. Item-based collaborative filtering suggests items based on their similarity, which is estimated based on user ratings.

2 Model based approaches

They are machine learning-based predictive models. The model's inputs are parameterized using features from the dataset in order to solve an optimization issue. Decision trees, rule-based techniques, latent factor models, and other model-based approaches are examples of model-based approaches.

B. Content Based Systems

Recommendations are generated by content-based systems depending on the user's preferences and profile. They strive to match users with goods they've previously liked. The degree of similarity between products is usually determined by the features of items that the user likes. Unlike most collaborative filtering models, which rely on ratings between the target user and other users, content-based models focus on the target user's own ratings. In essence, the content-based approach generates suggestions by combining several data sources.

The following data sources are required for the basic forms of content-based systems (these requirements can increase depending on the complexity of the system you're trying to build):

1. Item level data source

A reliable source of information about the item's characteristics. We have book pricing, num pages, published year, and other variables in our scenario. The more information you have about the object, the better for your system it will be.

2. User level data source

User comments on the item for which you're making recommendations. This type of feedback might be implicit or clear in nature. We're dealing with user ratings of books they've read in our sample data. The more user feedback you can collect, the better for your system it will be.

C. Hybrid Recommendation System

Each type of recommendation system has advantages and disadvantages. When applied alone, several of these techniques can appear to be limiting, especially when there are numerous data sources available for the problem. Systems that leverage a variety of available data sources to produce reliable inferences are known as hybrid recommender systems. Parallel and sequential designs are the two most common types of hybrid recommendation systems. A single output is produced by combining the recommendations from each of the numerous recommendation systems that are fed data from the parallel design. A single recommendation engine receives the input parameters from the sequential design, and the result is sequentially handed on to the next recommender.

II. Literature Survey

Wang, HC., Chen, CC. & Li, TW [1] Multimedia Tools and Applications has developed an HMM-based system that brings together news stories in chronological order and summarizes them. They claimed that the internet has altered how people get information. People nowadays prefer to get knowledge online rather than via actual paper resources, such as e-books, e-papers, and e-magazines.

They also mentioned that as time passes, more developments and subtheme reports may emerge. They also stated that in addition to the original news of the catastrophe, associated news about support, the rescue process, and casualties was regularly reported. Simultaneously, as an event unfolds, a large number of readers and netizens will enthusiastically discuss it, and the topic will continue to be of interest.

This study combines the time sequence and summary to construct the display of curated events and automates these aspects individually in the curation part. The topic model and an HMM were utilised in the time sequence step to represent each time interval based on the theme's strength and set conditions to determine the time intervals. To create the summary module for this study, extractive summarization method has been used. Theme strength and topic word distribution from the breakpoint are considered while selecting sentences from the module.

To compare and assess sentences, the detection module is employed. Readers are then given a summary of the event. Despite the fact that this HMM-based two-stage approach method yields excellent outcomes. It is not a scalable option as most of the summary are not accurate and usable. The authors of the paper have mentioned that they have handpicked the best sentences for the reference.

Jessica T. Feezell, John K. Wagner, Meredith [2] Computers in Human Behavior have done studies on political behaviour and polarisation of algorithmic news. They claim that non-algorithmic and algorithmic news have differing effects on political behaviour and polarisation. This paper's goal is to define and quantify the consequences of algorithmically generated news. They do so by separating two forms of algorithmically created news, socially driven and user-driven, and contrasting them with non-algorithmic news.

They observed that reading news from sites that utilise socially driven or user-driven algorithms is associated with higher levels of political participation than reading news from non-algorithmic sources. They also observed that neither non-algorithmic nor algorithmically picked news leads to an increase in political polarisation. The consequences of news consumption varied significantly depending on the manner of delivery, according to this study.

Guanjie Zheng, Fuzheng Zhang, Zihan Zheng[3] International World Wide Web Conferences Steering Committee propose a novel Deep Reinforcement Learning framework for news recommendation. They claim that dealing with dynamic changes in news suggestions is tough.

Traditional recommendation methods will reduce the user's reading options by recommending comparable items. Users will become bored as a result, and user happiness will suffer in the long run. To accomplish online personalised news recommendation, they present a DQN-based Deep Reinforcement Learning system. To forecast the possible reward, we utilise a multi-layer Deep Q-Network with a continuous state feature representation of users and a continuous action feature representation of items as the input.

There are two ways to demonstrate the dynamic shift in news recommendation. To begin with, news quickly becomes obsolete. Second, readers' interest in various news stories may change over time. Hybrid approaches are offered to improve user profile modelling by combining the benefits of the first two sets of methodologies.

Chuhan Wu, Fangzhao Wu, Yongfeng Huang & Xing Xie[4] CCF Transactions on Pervasive Computing and Interaction propose a neural news recommendation approach which can incorporate the implicit negative user feedback. They propose using a Transformer to capture behaviour relations and an additive attention network to select important news for user modelling to distinguish positive and negative news click behaviours based on reading dwell time, and learning separate user representations from positive and negative news clicks.

They subsequently make a click prediction. If a candidate news is comparable to the news that a user reads carefully, it is likely that this user will click on it. However, if it is very similar to news that a person has previously closed rapidly, it is unlikely that this user will click it. Finally, they introduce an interactive news modelling method that can learn unified news representations from title and body while taking their relationship into account. Extensive trials on real-world datasets suggest that adding negative user feedback improves the efficiency of user modelling for news recommendation, according to the authors.

III. Proposed News Aggregation System

In the proposed system, as shown in Figure 1, news articles are scraped from multiple news websites using Django dynamic scraper and stored in the database. The scraped data in the database is then analyzed for spam and duplicate news and those are discarded. Recommendation system using NLP tagging is used to categorize the data as per users' preference and the news articles are rendered to the user via web application.

A. Web Scraping using Django Dynamic Scraper

With Django Dynamic Scraper, scrapy scrapers can be defined dynamically using the Django admin interface, and scraped objects may be saved in the Django project's database (DDS). DDS isn't perfect for all scrapers because it simplifies things, but it's well suited for the fairly common scenario of scraping a website with a list of recently updated items (e.g., news, events, etc.) and then going into the detail page to scrape extra information for each item.

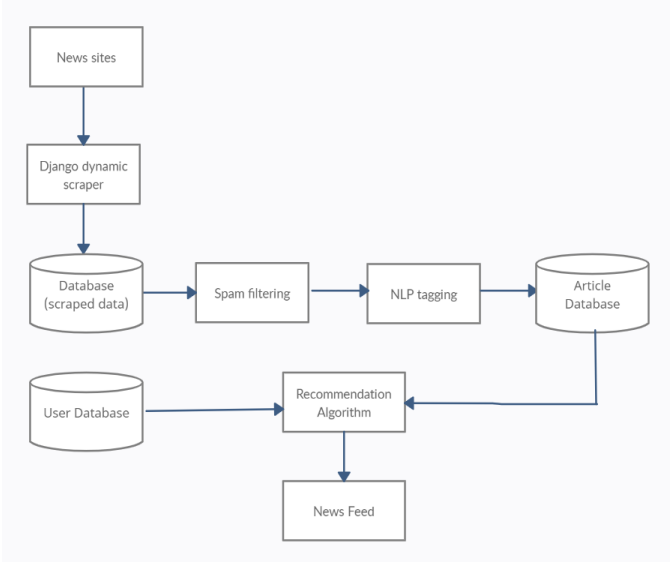

Figure 1. Block Diagram of Proposed News Aggregation System

Django Dynamic Scraper strives to keep its database data structure as independent as possible from the models in our app, therefore it includes its own Django model classes for creating scrapers, runtime information about our scraper runs, and classes for specifying the characteristics of the models we wish to scrape. Our Django models are mainly self-contained, with the exception of a few foreign key associations, and we won't have to update your model code every time DDS's model structure changes.

B. Django Web Application

Django is a Python web framework that encourages rapid development and practical design. It was designed by experienced developers to handle much of the web development work so that we can focus on constructing our app rather than reinventing the wheel. It's free and open source. A conventional data-driven website waits for HTTP requests from the web browser in a typical data-driven website (or another client). An application determines what is required when it receives a request based on the URL and maybe information in the POST or GET payload. Depending on what is required, it may then read or write data from a database or do additional actions to complete the request.

After that, the application will respond to the web browser by dynamically constructing an HTML page for the browser to view by inserting the retrieved data into placeholders in an HTML template. The code for each of these processes is usually separated into separate files in Django web applications. The following subsections provide a summary of the django web application block diagram depicted in Figure 2.

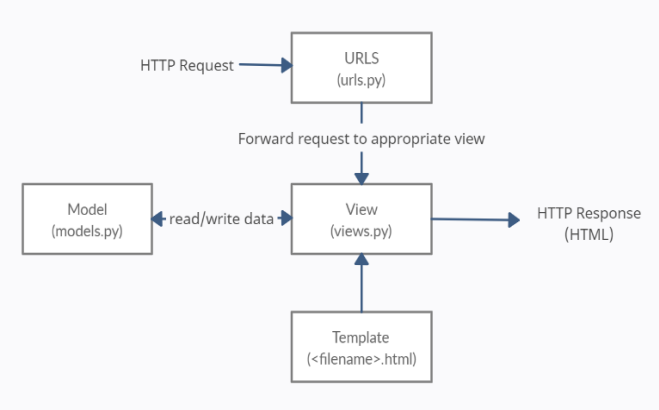

Figure 2. Block Diagram of Django Web Application

C. *URLs*

While it is feasible to process requests from all URLs with a single function, it is significantly more maintainable to write a distinct view function for each resource. A URL mapper is used to redirect HTTP requests to the appropriate page based on the request URL. A URL mapper can also match certain patterns of strings or digits in a URL and provide that information to a view function as data.

D. *View*

An HTTP request handler function that accepts and answers to HTTP requests is known as a view. Views get the data they need to fulfil requests from models, while templates take care of the return formatting.

E. *Models*

Models are Python objects that define an application's data structure and provide mechanisms for adding, changing, and deleting database items, as well as querying them.

F. *Templates*

A template is a text file that contains placeholders for actual content and dictates the structure and layout of a file (such as an HTML page). An HTML template can be used by a view to dynamically build an HTML page and populate it with data from a model. Not only can a template be used to define the structure of an HTML file, but it can also be used to describe the structure of any other type of file.

IV. Database and Backend

A. *Introduction*

Persistent connections eliminate the overhead of re-establishing a database connection with each request. The CONN MAX AGE option, which sets the maximum lifetime of a connection, is in charge of them. It can be customized for each database separately. The default value is 0, which keeps the database connection closed at the end of each request as it has been in the past. Set CONN_MAX_AGE to a positive integer of seconds to enable persistent connections.

Set it to None to allow for unlimited persistent connections. When Django performs its first database query, it establishes a connection to the database. This connection is kept open and reused in subsequent requests. Django terminates the connection when it reaches the CONN MAX AGE maximum age or when it is no longer usable.

Django automatically creates a database connection whenever it needs one and doesn't already have one — either because this is the initial connection or because the previous connection was closed. If the connection has surpassed its maximum age, Django terminates it at the start of each request. Set CONN MAX AGE to a lower value if your database server.

B. *Article Database*

The scraped news articles are stored in the article database. The schema of the article database is of the form as shown in Table I. News Website field holds the corresponding news website name from which news articles are scraped. Title field holds the news headline, Image field captures the image of the corresponding new article, Link field contains the URL of the detailed news. Checker Runtime field is a foreign key of Scheduler Runtime django class. It defines the duration between which the article's status is checked for in the news website.

TABLE I
Articles Database Schema

Articles	
Field Name	Datatype
News Website	ForeginKey(NewsWebsite)
Title	CharField
Image	ImageField
Link	URLField
Description	TextField
Checker Runtime	ForeignKey(Scheduler Runtime)

C. *News Website Database*

The news websites from which news articles are to be scraped are stored in the News Website database. The schema of the news websites database is of the form as shown in Table II. The Name field stores the name of the website, URL field holds the respective URLs of news sites, Category field is a foreign key of Category database. It stores the category within which the news website falls into. Scraper field contains the corresponding scraper which scrapes the data from the news website. Scraper Runtime field stores the duration between which the news site has to be scraped periodically.

TABLE II
NewsWebsite Database Schema

News Websites	
Field Name	Datatype
Name	CharField
URL	URLField
Category	ForeignKey(Category)
Scraper	ForeignKey(Scraper)
Scraper Runtime	ForeignKey(Scheduler Runtime)

D. *Category Database*

The different categories within which news websites and their corresponding articles fall into are defined in the Category database. The schema of the category database is of the form as

shown in Table III. Title field holds the name of the category and Image field contains the image used to represent each category.

TABLE III
CATEGORY DATABASE SCHEMA

Articles	
Field Name	Datatype
Title	CharField
Image	ImageField

V. RECOMMENDATION ALGORITHM

There are multiple methods to represent a text as a d-dimensional vector like Bag of words, TF-IDF method, Word2Vec embedding. Each method has its own advantages and disadvantages.

To carry out most of everything in the paper we implement all the mentioned methods and combine the recommendations to provide a master list.

A. Bag of Words Method

The occurrence of words within a document is represented by the Bag of Words (BoW) approach. Each headline can be thought of as a document, and the collection of all headlines is referred to as a corpus.

Each document is represented by a d-dimensional vector using the BoW technique, where d is the total number of unique words in the corpus. The vocabulary is made up of a collection of such unique terms.

The method is straightforward and adaptable, and it may be used to extract information from documents in a variety of ways. A bag-of-words is a text representation that describes the frequency with which words appear in a document. It entails two steps, A measure of the presence of recognised words, as well as a lexicon of known terms.

Because any information about the sequence or structure of words in the document is deleted, it is referred to as a "bag" of words. The model simply cares about whether or not recognised terms appear in the document, not where they appear. Based on the headline, the developed algorithm suggests ten comparable articles to the searched (read) item. It takes two arguments: the index of an article that has already been read and the total number of articles that should be recommended.

It finds and recommends the ten closest neighbours based on the Euclidean distance.The assumption is that documents with comparable content are similar. Furthermore, we can deduce something about the document's significance solely from its content.

Creating the bag-of-words can be as simple or as complex as possible. The difficulty arises from deciding how to create a vocabulary of known words (or tokens) as well as how to rate the existence of known terms.

The bag-of-words paradigm is simple to learn and use, and it provides a lot of customization options for your individual text data.

It has a number of flaws, including the following:

1. *Vocabulary*

   The vocabulary must be carefully designed, particularly in order to manage the size, which has an impact on the document representations' sparsity.

2. *Sparsity*

   Sparse representations are more difficult to model, both in terms of computational difficulty (space and time complexity) and information complexity (models must harness so little information in such a huge representational space).

3. *Meaning*

   Ignoring word order misses the context and, as a result, the meaning of the document's words (semantics). Context and meaning may determine the difference between the same words organised differently ("this is fascinating" vs "is this interesting"), synonyms ("old bike" vs "used bike"), and much more if they are modelled.

B. Term Frequency and Inverse Document Frequency Method

The TF-IDF approach is a weighted measure that gives less frequent terms in a corpus greater weight. It gives each term (word) in a document a weight based on Term Frequency (TF) and Inverse Document Frequency (IDF) (IDF).

Term frequency analyses the frequency of a term you're interested in in relation to the rest of the document. There are several methods for determining frequency:

- The number of times a word is used in a document (raw count).
- Term frequency has been modified to account for the document's length (raw count of occurences divided by number of words in the document).
- Frequency that is logarithmically scaled (e.g. log(1 + raw count)).
- Frequency of Boolean expressions (e.g. 1 if the term occurs, or 0 if the term does not occur, in the document).

$$TF(i,j) = \frac{Number\ of\ times\ word\ i\ appears\ in\ document\ j}{Number\ of\ words\ in\ document\ j} \quad \ldots(1)$$

Inverse document frequency examines the frequency (or rarity) of a term in the corpus. The IDF is calculated as follows: t is the term (word) whose commonness we want to assess, and N is the number of documents (d) in the corpus (D).. The numerator is just the number of papers that include the phrase t.

$$IDF(i,D) = \log\left(\frac{Number\ of\ documents\ in\ the\ corpus\ D}{Number\ of\ documents\ containing\ word\ i}\right) \quad \ldots(2)$$

The essential assumption behind TF-IDF is that a term's relevance is inversely proportional to its frequency across documents. TF informs us about the frequency with which a term appears in a document, while IDF informs us about the relative rarity of a term in the collection of documents. We can get our final TF-IDF value by multiplying these numbers together.

$$weight(i,j) = TF(i,j) * IDF(i,D) \quad \ldots(3)$$

As a result, if a word appears more times in one document but fewer times in all others, its TF-IDF score will be high. The higher the TF-IDF score, the more important or relevant the

term is; the lower the TF-IDF value, the less relevant the term is.

The TF-IDF approach does not capture the semantic and syntactic similarity of a given word to other words, but Word embedding may. Word embedding algorithms like Word2Vec, GloVe, and fastText take advantage of word semantic similarity.

C. *Word2Vec Embedding*

Google invented Word2Vec, one of the strategies for semantic similarity, in 2013. During training, it looks for patterns in a corpus and represents each word with a d-dimensional vector. We need a large corpus to get better outcomes.

Because our corpus is tiny, we'll use Google's pre-trained model on news items from Google. This standard model includes vector representations for billions of words, which were learned from millions of fresh articles. Each word is represented by a 300-dimensional dense vector in this instance.

We need to obtain vector representations for each headline because the model provides vector representations for each word, but we need to determine the distance between headlines. The average can be calculated by first accumulating vector representations of all the words in the headline and then calculating the average. The average Word2Vec model is another name for it.

D. *Creating the Final Recommendation List*

For providing the final recommended list, Introducing the aggregate algorithm. Which combines the above-mentioned BoW, TF-IDF, Word2Vec models and updates the data set with only latest news to get the results. This ensures that the user gets up to date news only. The Aggregate function $\boldsymbol{A}$ in equation (4) performs a union of all the values in the models.

$$\boldsymbol{A} = (\boldsymbol{BoW}) \cup (\boldsymbol{TFIDF}) \cup (\boldsymbol{Word2Vec}) \quad \ldots(4)$$

Then the output $\boldsymbol{A}$ is added to the database to be recommended to the user. When performing the Aggregate function, the total recommendable articles increase. So, a priority and randomize function is used such that Word2Vec has highest priority, TF-IDF and BoW has lower. About 30% of the articles are randomized to make the list much more consistent and not polarizing.

VI. Experimental Results

After implementing all the methods and creating the aggegation system,

After extensive testing and multiple tries, It is shown that by using about 30% randomization provides much more neutral and more explorable readings for the user.

The following Figures 3-6 provide the experimental results for dofferent models.

A. *Bag of Words Model*

```
###### Bag of Words #######
Query: Woman Fired After Flipping off Trump's Motorcade Sues Former Employer
###### Recommended Article #######
1. The Supreme Court Just Made It A Lot Harder For You To Sue Your Employer
2. The Trump Administration Is Suing California Again
3. Lou Dobbs Flips Out on Live TV, Urges Trump To 'Fire The sOB' Robert Mueller
4. Cardi B's Former Manager Sues Her For $10 Million
5. A Third Woman Is Suing To Break A Trump-Related Nondisclosure Agreement
6. Former RNC Chair Fires Back At claim He Was Only Hired Because He Was Black
7. State Employer Side Payroll Taxes And Loser Liberalism
8. Democrats Flip Kentucky State House Seat Where Trump Won Overwhelmingly
9. Big Tax Game Hunting: Employer Side Payrol1 Taxes
10. Democrats Flip 2 More GOP-Held State House Seats
```

Figure 3. Output of Bag of Words model

B. *TF-IDF Model*

```
###### TF-IDF #######
Query: Woman Fired After Flipping off Trump's Motorcade Sues Former Employer
###### Recommended Article #######
1. The Trump Administration Is Suing California Again
2. Texas Sues Trump Administration To End DACA
3. Stormy Daniels Suing Trump Over Nondisclosure Agreement
4. Trump: Mueller Should Never Have Been Appointed
5. Spanish Woman Looks More Like Trump Than The Donald Himself
6. What You Should Know About Trump's Nihilist Budget
7. The Caliphate Of Trump And A Planet In Ruins
8. Trump Ally Sues Qatar For Hacking His Email
9. All They Will Call You Will Be Deportees
10. Pursuing Desegregation In The Trump Era
```

Figure 4. Output of TF-IDF Model

C. *Word2Vex Embedding*

```
###### Word2Vec #######
Query: Woman Fired After Flipping off Trump's Motorcade Sues Former Employer
###### Recommended Article #######
1. White House Lawyer Insists Trump Isn' t Considering Firing Mueller
2. White House spent Honths Denying hat rump Considered iring Mueiler
3. Trump Claims He "Never Met Woman Accusing Him Of Sexually Assaul ting Her In Trump Tower
4. 17 States Sue Trump Administration Over Census Citizenship Question
5. Husband of Former Trump Household Staffer Now An EPA Official
6. Giuliani Tells Mueller To Back off "Fine Woman Ivanka Trump But Calls Kushner Disposable
7. Former Trump Attorney Stuns Fox & Friends, Says Stormy Daniels NDA Is Likely Invalid
8. Giuliani Says Trump Repaid Lawyer For $138, 000 Payment To Stormy Daniels
9. olympian Gus Kenworthy Burns Ivanka Trump: "TF Is She Doing Here?
10. Trump HUD Official Lynne Patton Under Fire After calling Journalist Miss Piggy
```

Figure 5. Output for Word2Vec Embedding system

D. *Aggregation Output*

```
###### Aggeregation Function #######
Query: Woman Fired After Flipping off Trump's Motorcade Sues Former Employer
###### Recommended Article #######
1. Democrats Flip 2 More GOP-Held State House Seats
2. 17 States Sue Trump Administration Over Census Citizenship Question
3. Texas Sues Trump Administration To End DACA
4. White House Lawyer Insists Trump Isn' t Considering Firing Mueller
5. Former Trump Attorney Stuns Fox & Friends, Says Stormy Daniels NDA Is Likely Invalid
6. The Trump Administration Is Suing California Again
7. Democrats Flip Kentucky State House Seat Where Trump Won Overwhelmingly
8. What You Should Know About Trump's Nihilist Budget
9. olympian Gus Kenworthy Burns Ivanka Trump: "TF Is She Doing Here?
10. Pursuing Desegregation In The Trump Era
```

Figure 6. Final output of the Aggregation Algorithm.

In Figure 7 shows the module implemented in Django server Backend and the final output of the project in which the webpage output is presented with Aggregation function providing the articles.

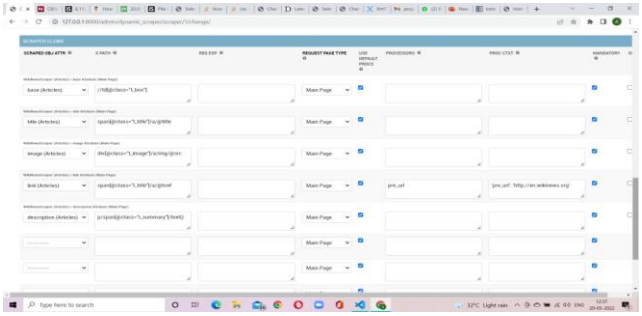

Figure 7.Django server implementing scrapers

VII. Conclusion

There are advantages and disadvantages on all the above implemented algorithms. The BoW and TF-IDF models do not recommend based on the context. While The Word2Vec is based on individual personalised feed recommendation. Which at times gets repetitive and boring. So, by incorporating both versions, this paper has tried to compensate both. Individual user based as well as room to explore topics for the user. In Future this model can incorporate even more different types of algorithms like BERT and GloVe techniques. By using Django as backend, the admins can easily add or remove the pages that needs to be dynamically scraped and processed.

References

[1] Wang, HC Chen, "Automatic content curation of news events," Multimedia Tools Application," https://doi.org/10.1007/s11042-022-12224-4, Vol: 81 (2022).

[2] Wu, C., Wu, F., Huang, Y. et al. "Neural news recommendation with negative feedback," CCF Trans. Pervasive Comp. Interact. https://doi.org/10.1007/s42486-020-00044-0, Vol: 2, Pages 178–188 (2020).

[3] Morteza Zihayat, Anteneh Ayanso, Xing Zhao, "A utility-based news recommendation system, Decision Support Systems," ISSN 0167-9236, https://doi.org/10.1016/j.dss.2018.12.001, Volume 117, Pages 14-27, 2019.

[4] S. Khan, T. Khan, C. Prasad, A. Khatri and I. Khan, "Intelligent News Aggregator and Validator," International Conference on Nascent Technologies in Engineering (ICNTE), https://doi.org/10.1109/ICNTE44896.2019.8945945, Pages 1-5, 2019

[5] Shreesha, M., S. B. Srikara, and R. Manjesh, "A Novel Approach for News Extraction Using Web scraping," 3rd National Conference on Image Processing, Computing, Communication, Networking and Data Analytics, https://doi.org/10.21467/proceedings.1.56, Volume 3, 2018.

[6] Chang HT, Liu SW, Mishra N, "A tracking and summarization system for online Chinese news topics," Aslib Journal of Information Management, https://doi.org/10.1108/AJIM-10-2014-0147, Volume 67, Issue 6 Pages 687–699, 2018.

[7] Hou SL, Lu RQ, "Knowledge-guided unsupervised rhetorical parsing for text summarization," Information Systems, https://doi.org/10.1016/j.is.2020.101615, Volume 94, 2020.

[8] Chen, C., Meng, X., Xu, Z., Lukasiewicz, T., "Location-aware personalized news recommendation with deep semantic analysis," IEEE Access, https://doi.org/10.1109%2FACCE SS.2017.2655150, Volume 5, Pages 1624–1638, 2017.

[9] Ren, R., Zhang, L., Cui, L., Deng, B., Shi, Y., "Personalized Financial News Recommendation Algorithm Based on Ontology," Procedia Computer Science, https://doi.org/10.1016%2Fj.procs.2015.07.151, Volume 55, Pages 843–851, 2015

[10] Zihayat, M., Ayanso, A., Zhao, X., Davoudi, H., An, A. "A utility-based news recommendation system," Decision Support Systems, https://doi.org/10.1016%2Fj.dss.2018.12.001, Volume 117, Pages 14–27, 2019.