INFO-442-001 - SU 24-25
Data Science Projects

Kevin Shi, Richardson Chhin, Benjamin Leung, Kathryn Swatek
Professor Lei Wang

Final Project Report
News Article Recommendation System
Group 3

1 Introduction

1.1 Problem Statement

There are oceans of news articles online, with countless new ones getting published every day. However, most people only have a few minutes in their day to digest their news sources. News personalization is essential for everyday readers because it delivers information that they are looking for efficiently and fast, ensuring that they are not taking valuable time out of their day browsing for articles. This explains the need for research to enhance the way users read the news, though there are several challenges with news personalization and recommendations.

When it comes to news recommendation research, although there has been extensive development of recommendation systems in areas such as products and movies, there has not been a focus on specifically news recommendation. News data significantly differs from data from other domains. One challenge in recommending news content is that new stories appear frequently and older ones quickly become outdated, leading to the cold-start problem. Another challenge is that news articles are rich in textual information, and accurately capturing user interest requires understanding the semantic content of the articles themselves. User interest must also be inferred indirectly based on a user's click behavior, as news platforms generally lack explicit user feedback in the form of ratings.

Although there is less research in the field of news recommendation, leveraging the minimal number of small-sized public news datasets available can facilitate more research and studies to understand how to improve news recommendation in information retrieval.

1.2 Project Goals

The "MIND: A Large-scale Dataset for News Recommendation" paper presents an approach to news recommendation using advanced deep learning techniques and complex models such as multi-view neural networks and transformer-based architectures, which incorporate user behavior modeling, personalized news encoders, and attention mechanisms. These models are highly effective for large-scale recommendation tasks and achieve robust performance on the MIND dataset.

The goal of our project is to explore and understand the core principles of content-based recommendation systems by developing a news recommender system that effectively recommends articles to users. Rather than implementing complex deep learning architectures, our goal is to build and compare three content-based recommenders using relatively more basic text representation techniques: TFIDF, Word2Vec, and BERT. These models represent different levels of textual understanding from sparse, frequency-based vectors to dense semantic embeddings, and will allow us to examine how the choice of representation affects recommendation quality.

2 Dataset Description

The MIND dataset was compiled by randomly sampling anonymized behavior logs from Microsoft News of users who had at least five new clicks during a six-week span from October 12 to November 22, 2019. For the scope of this project, our recommendation model was built off the MIND-small dataset, which contained behavior logs of 50,000 randomly sampled users. The file storing user behaviors (called *behaviors.tsv*) contains five columns:

1. Impression ID: ID of impression
2. User ID: Anonymous ID of the user
3. Time: The impression time with format "MM/DD/YYYY HH:MM:SS AM/PM"
4. History: The news click history (ID list of clicked news) of this user before this impression
5. Impressions: List of news displayed in this impression and user's click behaviors on them (1 for click and 0 for non-click)

The *news.tsv* file contains information about all the articles that users had interacted with previously or during the data collection period. Within this dataset, there are 7 columns per news article, including:

1. News ID
2. Category: Labels manually tagged by editors
3. Subcategory: Labels manually tagged by editors
4. Title
5. Abstract
6. URL
7. Title Entities: Entities contained within news title
8. Abstract Entities: Entities contained within news abstract

The time period of the training set ranges from 2019-11-09 to 2019-11-14. The validation set for both the behaviors log and news articles was assembled from the last day of the fifth week in the training set (2019-11-15).

Lastly, the file names *entity_embeddings.vec* and *relation_embeddings.vec* both contain 100-dimension embeddings of entities and relations learned from the subgraph learned by the TransE method. The first column of this dataset is the ID of the entity/relation, and the other columns are the embedding vector values.

3 Methodology

3.1 Codebase Description

Links to our Hugging Face repo: Datasets and Models
Data Preprocessing:

1. data_preprocessing_train.ipynb
    - Handles preprocessing of the raw training data. Reads behaviors.tsv and news.tsv, performs data cleaning, tokenizes the textual content of the news articles, and then generates and saves embeddings using TF-IDF, Word2Vec, and BERT techniques
    - Inputs: behaviors.tsv, news.tsv
    - Outputs: behaviors_train_preprocessed.parquet, news_train_with_item_vectors.parquet, behaviors_train_user_profile_tfidf.parquet, behaviors_train_user_profile_w2v.parquet, behaviors_train_user_profile_bert.parquet
2. data_preprocessing_val.ipynb
    - Handles preprocessing of the validation data. Reads behaviors.tsv and news.tsv, performs data cleaning, tokenizes the textual content of the news articles, and then generates and saves embeddings using TF-IDF, Word2Vec, and BERT techniques
    - Inputs: behaviors.tsv, news.tsv
    - Outputs: behaviors_val_preprocessed.parquet, news_val_with_item_vectors.parquet, behaviors_val_user_profile_tfidf.parquet, behaviors_val_user_profile_w2v.parquet, behaviors_val_user_profile_bert.parquet

EDA:

1. Train_EDA.ipynb
    - Conducts exploratory data analysis on the preprocessed training dataset. It includes various data visualizations and statistical summaries to better understand the data's underlying patterns and distributions
    - Inputs: behaviors_train_preprocessed.parquet, news_train_with_item_vectors.parquet
2. Val_EDA.ipynb
    - Conducts exploratory data analysis on the preprocessed validation dataset. It includes various data visualizations and statistical summaries to better understand the data's underlying patterns and distributions
    - Inputs: behaviors_val_preprocessed.parquet, news_val_with_item_vectors.parquet

Model Training and Evaluation

1. recommender_tfidf_train.ipynb
    - Builds and evaluates the content-based recommender system using TF-IDF vectors for the training set. Calculates cosine similarity scores between user profiles and candidate news articles and computes various evaluation metrics
    - Inputs: behaviors_train_user_profile_tfidf.parquet, news_train_with_item_vectors.parquet
    - Outputs: all_scores_tfidf_train.pkl
2. recommender_tfidf_val.ipynb
    - Builds and evaluates the content-based recommender system using TF-IDF vectors for the validation set. Calculates cosine similarity scores between user profiles and candidate news articles and computes various evaluation metrics
    - Inputs: behaviors_val_user_profile_tfidf.parquet, news_val_with_item_vectors.parquet
    - Outputs: all_scores_tfidf_val.pkl
3. recommender_w2v_train.ipynb
    - Builds and evaluates the Word2Vec-based content recommender system on the training set. Follows a similar process to the TF-IDF notebook, but uses Word2Vec embeddings to represent the articles
    - Inputs: behaviors_train_user_profile_w2v.parquet, news_train_with_item_vectors.parquet
    - Outputs: all_scores_w2v_train.pkl
4. recommender_w2v_val.ipynb
    - Builds and evaluates the Word2Vec-based content recommender system on the validation set. Follows a similar process to the TF-IDF notebook, but uses Word2Vec embeddings to represent the articles
    - Inputs: behaviors_val_user_profile_w2v.parquet, news_val_with_item_vectors.parquet
    - Outputs: all_scores_w2v_val.pkl
5. recommender_bert_train.ipynb
    - Builds and evaluates a content-based recommender system using BERT embeddings for the training data. Process is similar to other recommender notebooks, but uses the powerful contextual embeddings from BERT
    - Inputs: behaviors_train_user_profile_bert.parquet, news_train_with_item_vectors.parquet
    - Outputs: all_scores_bert_train.pkl
6. recommender_bert_val.ipynb
    - Builds and evaluates a content-based recommender system using BERT embeddings for the validation data. Process is similar to other recommender notebooks, but uses the robust contextual embeddings from BERT.

- Inputs: behaviors_val_user_profile_bert.parquet, news_val_with_item_vectors.parquet
- Outputs: all_scores_bert_val.pkl

7. INFO442 Eval.ipynb
    - The final evaluation and comparison for all three notebooks. Loads the similarity scores from all models and generates comparative visualizations and statistical analysis of their performance
    - Inputs: all_scores_bert_train.pkl, all_scores_tfidf_train.pkl, all_scores_w2v_train.pkl, behaviors_train_user_profile_w2v.parquet, news_train_tokenized.parquet

3.2 Data Preprocessing

*See notebooks (*data_preprocessing_train.ipynb* and *data_preprocessing_val.ipynb*) for full data preprocessing steps

Data cleaning for behaviors_train.tsv:

- Get correct column types (e.g., convert time column to datetime)
- Fill NaNs for users with no history
- Convert history string to a list
- Parse impressions into (news_id, label) pairs: (candidate_news_id, clicked)
- Explode the impressions column into two columns: 'candidate_news_id' (news_id) and 'clicked' (label 0 or 1)

Data cleaning for news_train.tsv:

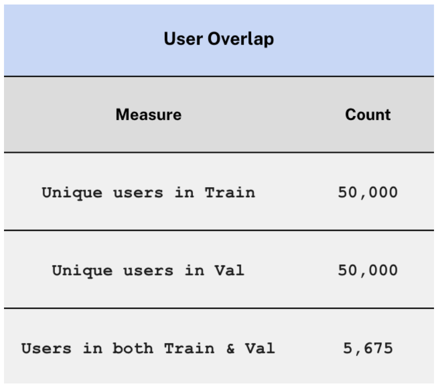
- Drop incomplete articles from news (i.e., rows with NaN in the 'abstract' column)
- Remove any references to dropped incomplete articles from:
    - The History column in behaviors_train.tsv
    - The Impressions column in behaviors_train.tsv
    - Remove rows with no impressions remaining
- Check for duplicate rows

Text preprocessing for news_train.tsv:

- Concatenate the content of the 'title' and 'abstract' columns into a column called 'text'
- Clean and tokenize text:
    - Remove all characters from the text that are not lowercase letters or whitespace (punctuation, numbers, special symbols)
    - Make text lowercase
    - Split the cleaned text into a list of individual words (tokens)
    - Lemmatization and stopword removal
    - Apply the above cleaning and tokenization to the 'text' column
- For TF-IDF: Join tokens back into text
- For Word2Vec: Keep the tokenized version, but make a plain list of strings
- For BERT: Combine title and abstract as a single string

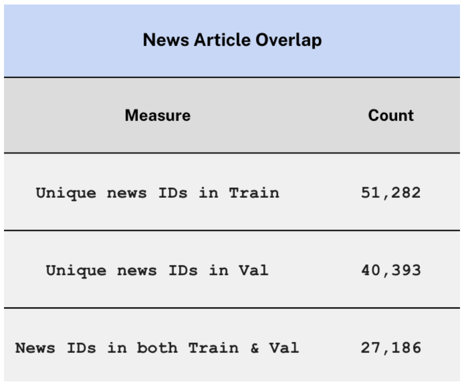
3.3 EDA

a. EDA Overview



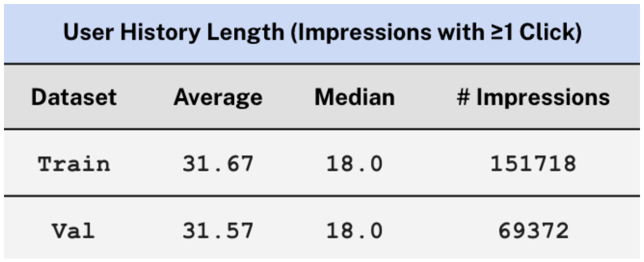
User Overlap	
Measure	**Count**
Unique users in Train	50,000
Unique users in Val	50,000
Users in both Train & Val	5,675

User base description: 50k unique users in both training and validation set, with 11% overlap



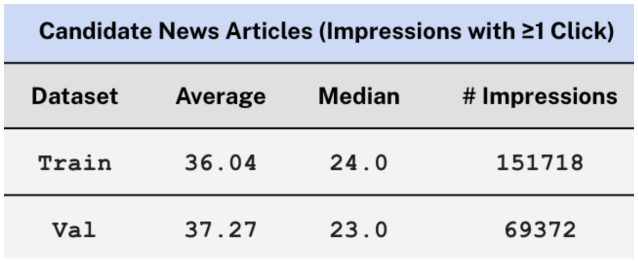
News Article Overlap	
Measure	**Count**
Unique news IDs in Train	51,282
Unique news IDs in Val	40,393
News IDs in both Train & Val	27,186

News articles description: 51k unique articles in the training set and 40.4k unique articles in the validation set, with 67% overlap



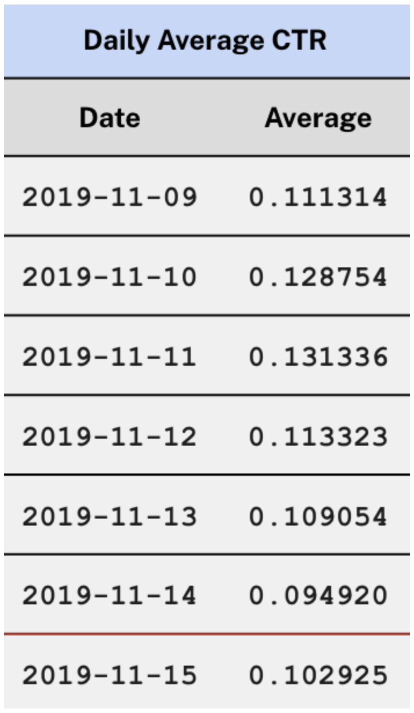
User History Length (Impressions with ≥1 Click)			
Dataset	**Average**	**Median**	**# Impressions**
Train	31.67	18.0	151718
Val	31.57	18.0	69372

User History Length Summary
Train and validation sets have similar averages and medians.



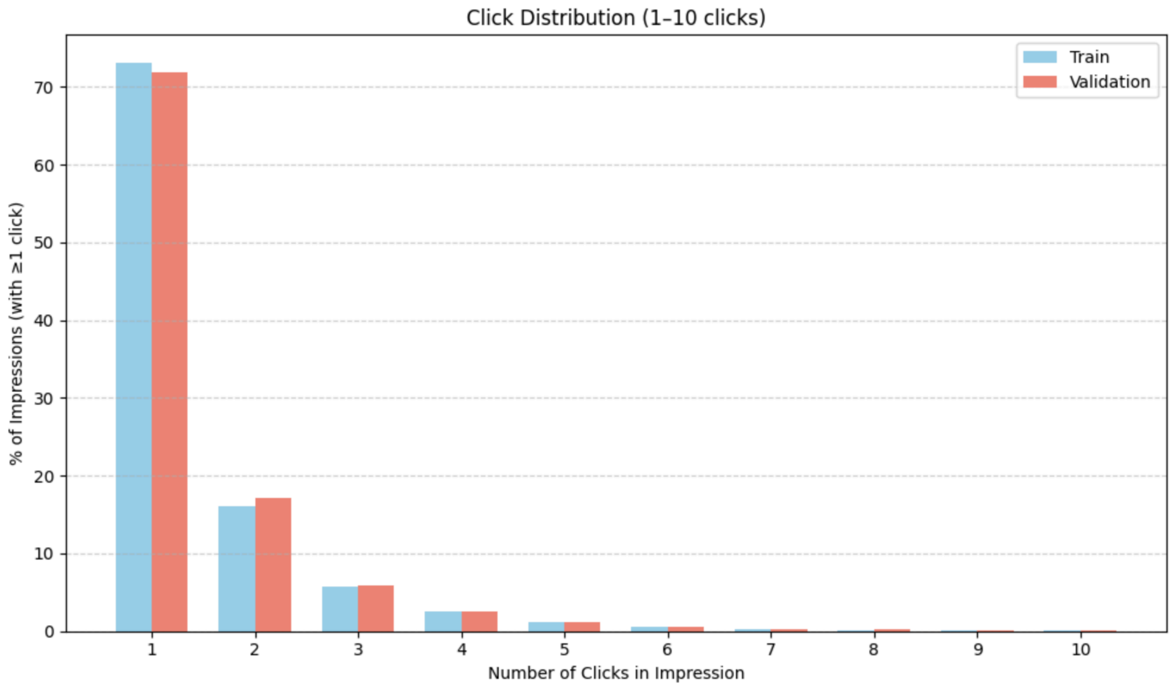
Candidate News Articles (Impressions with ≥1 Click)			
Dataset	**Average**	**Median**	**# Impressions**
Train	36.04	24.0	151718
Val	37.27	23.0	69372

Impression Length Summary
Train and validation sets have similar averages and medians.



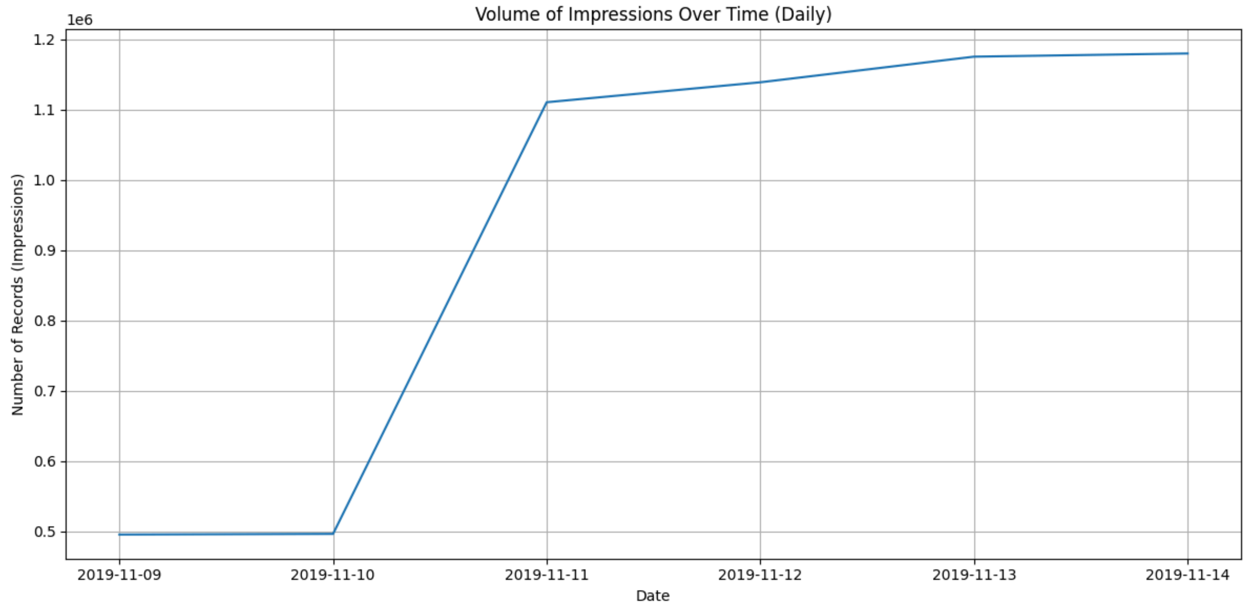
Daily Average CTR	
Date	**Average**
2019-11-09	0.111314
2019-11-10	0.128754
2019-11-11	0.131336
2019-11-12	0.113323
2019-11-13	0.109054
2019-11-14	0.094920
2019-11-15	0.102925

Click Through Rate (Clicks/Impressions) Averages by Day

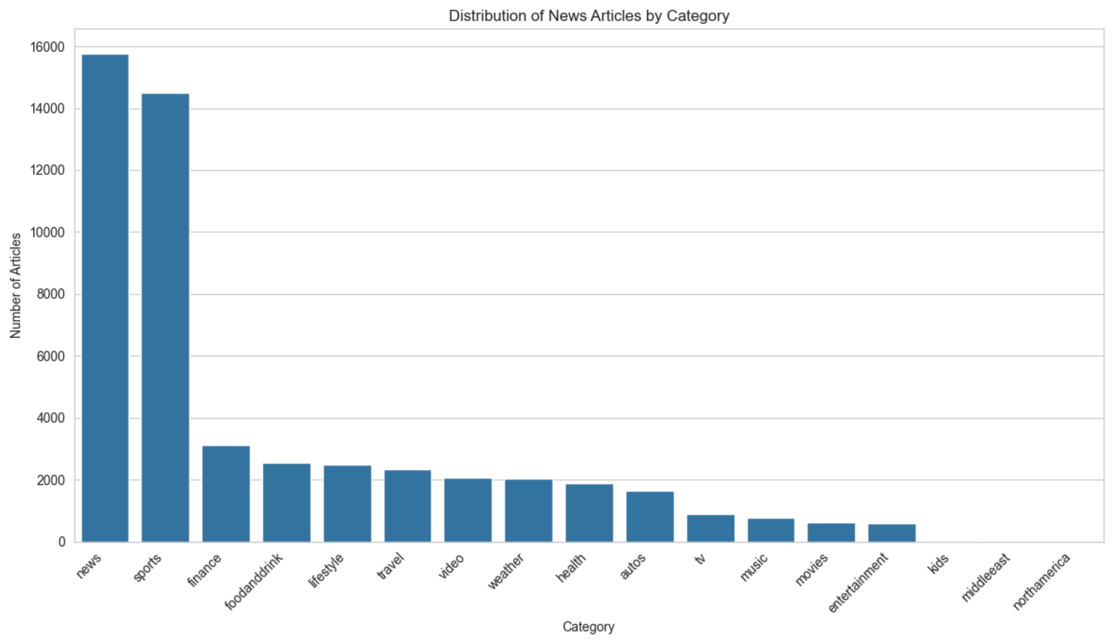


Train and validation sets have similar click distributions.

b. Training Set EDA

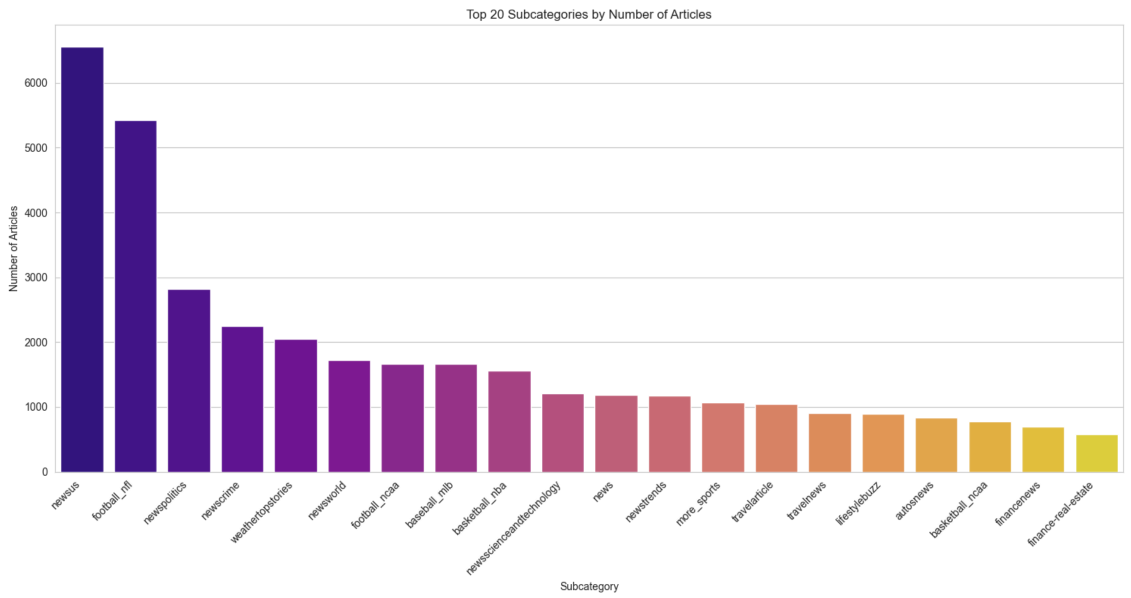


Impression count by day



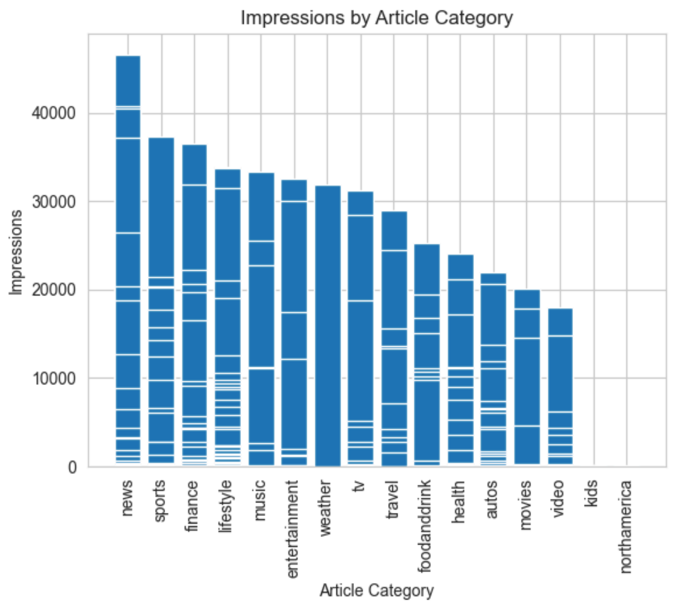
Article count by Category

17 distinct categories; Most common categories in the article dataset: news, sports, finance



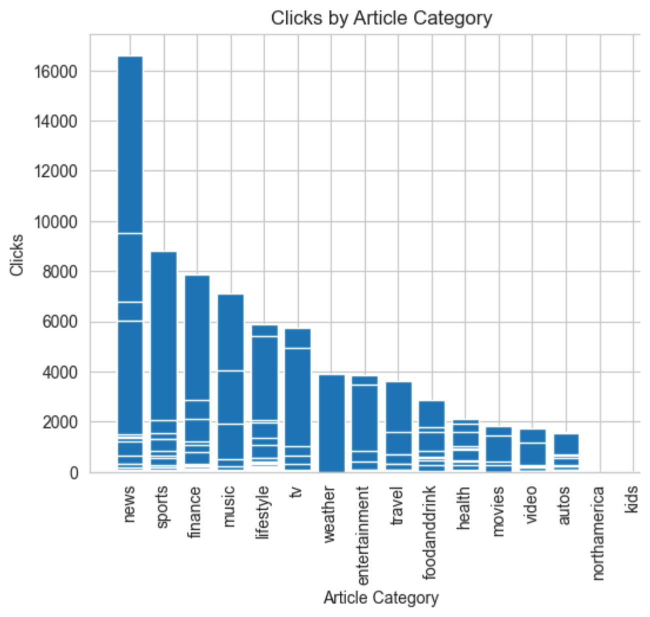
Article count by Subcategory (Top 20)

264 distinct subcategories; Most common subcategories in article dataset: newsus, football_nfl, newspolitics



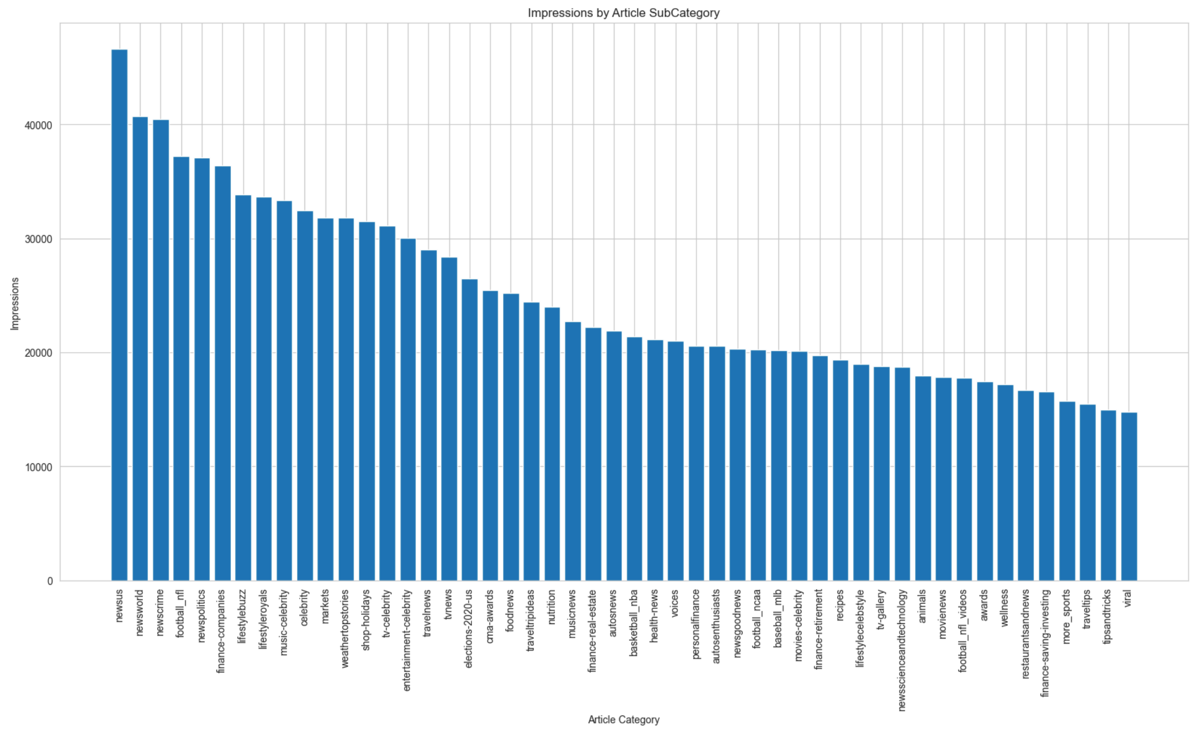
User Impressions by Article Category

Most viewed categories: news, sports, finance



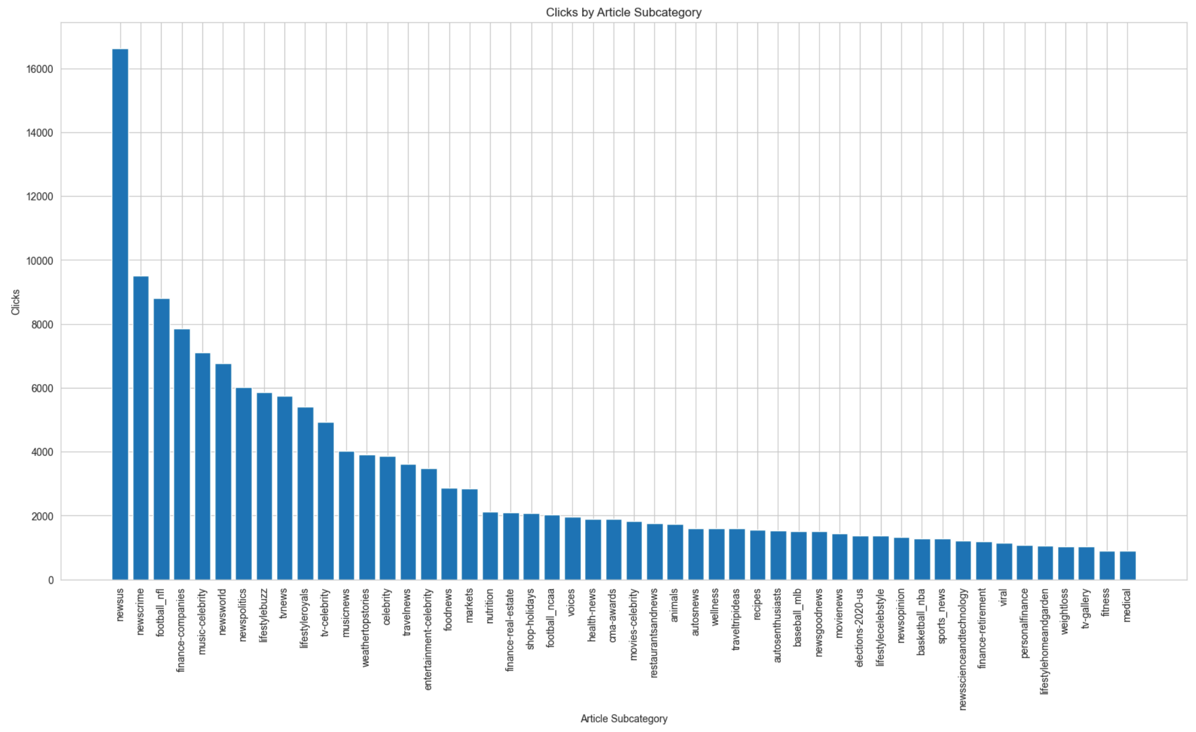
User Clicks by Article Category

Most engaged with categories: news, sports, finance



User Impressions by Article Subcategory

Most viewed subcategories: newsus, newsworld, newscrime

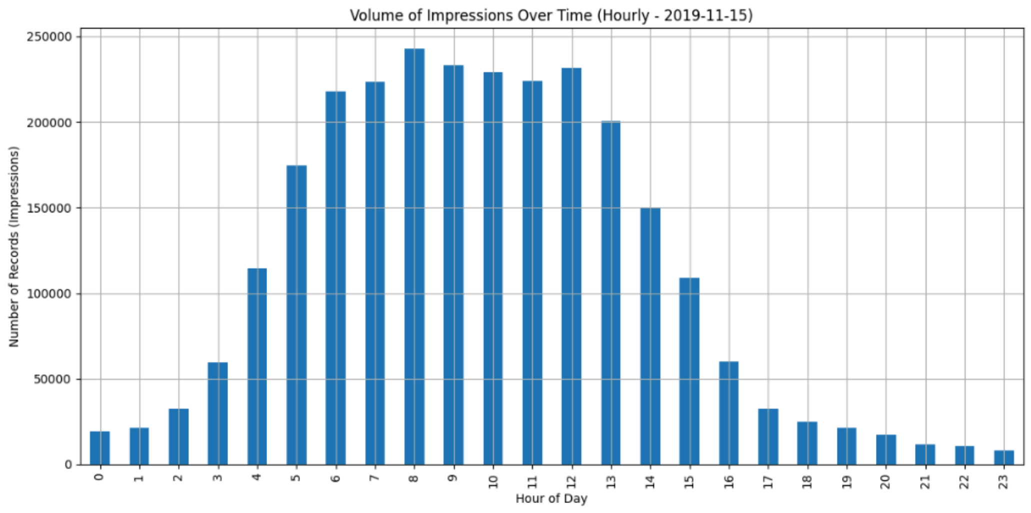


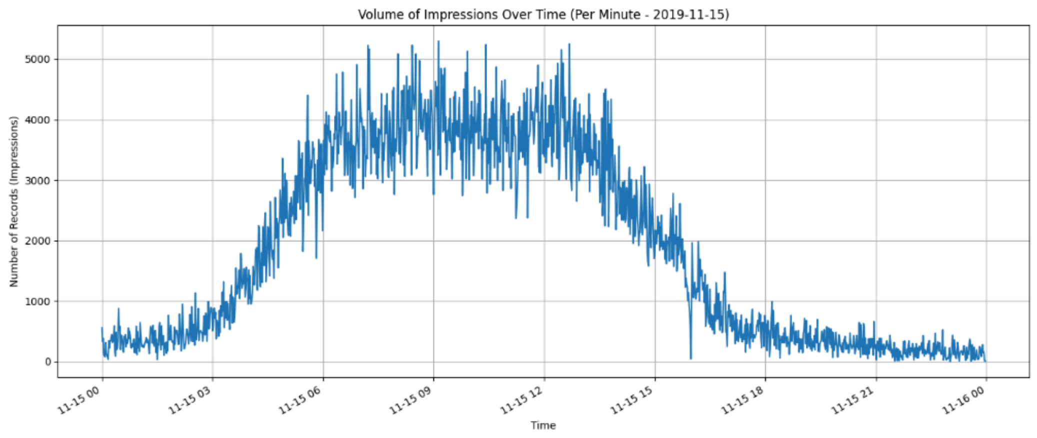
User Clicks by Article Subcategory

Most engaged with subcategories: newsus, newscrime, football_nfl

c. Validation Set EDA

Volume of Records Over Time (Hourly & Per Minute)

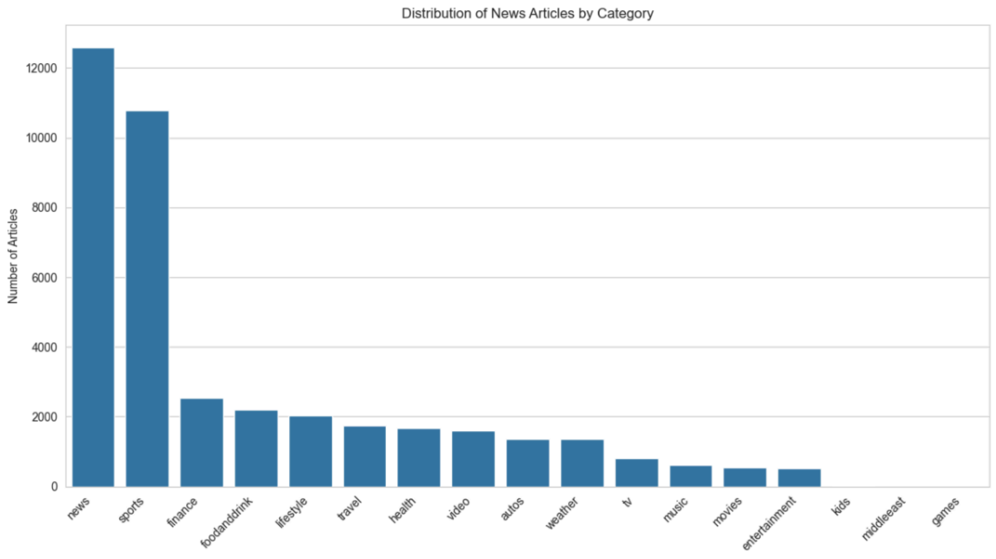




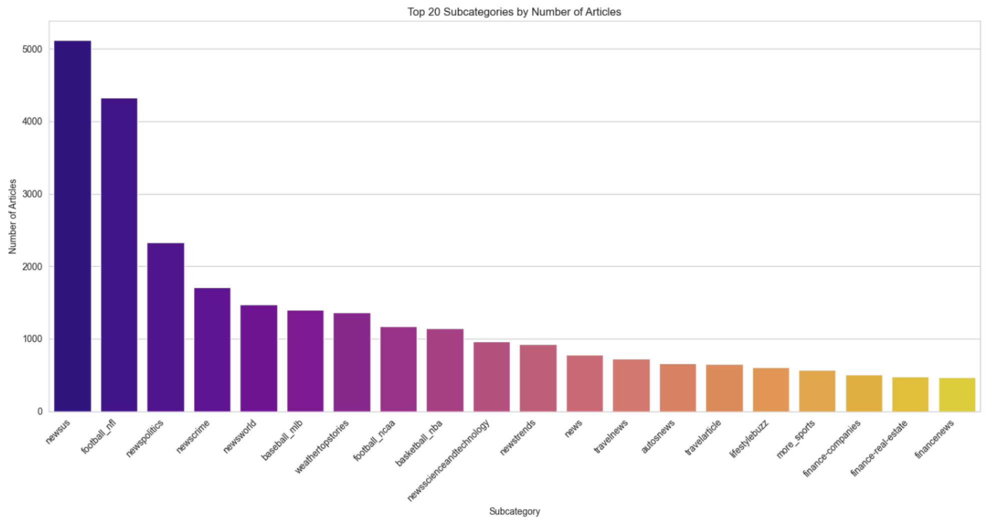
Top 10 Articles with the highest click rates with at least 100 impressions

	candidate_news_id	num_clicks	num_nonclicks	click_rate
3259	N47020	129.0	363.0	26.22
1265	N25064	27.0	89.0	23.28
3328	N47854	28.0	101.0	21.71
2192	N34973	26.0	94.0	21.67
3778	N53182	29.0	112.0	20.57
1765	N30598	142.0	558.0	20.29
402	N14802	99.0	447.0	18.13
988	N21679	759.0	3555.0	17.59
108	N11390	1717.0	8117.0	17.46
1908	N31958	8042.0	39243.0	17.01

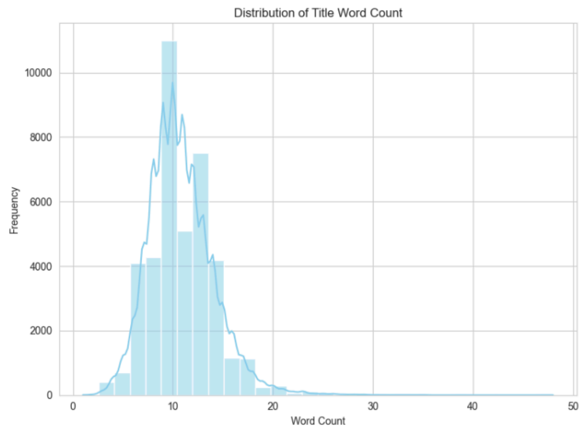
News Articles by Category

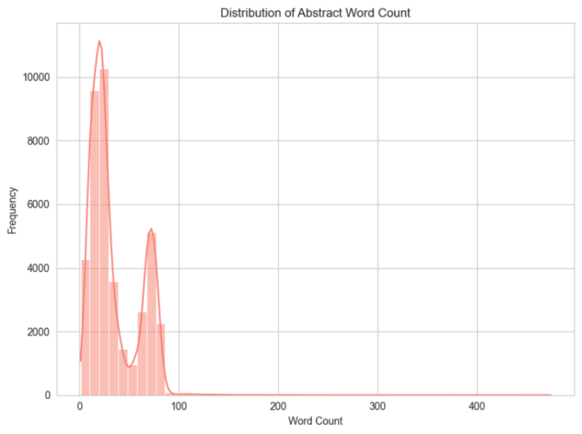


Top 20 Subcategories by Number of Articles

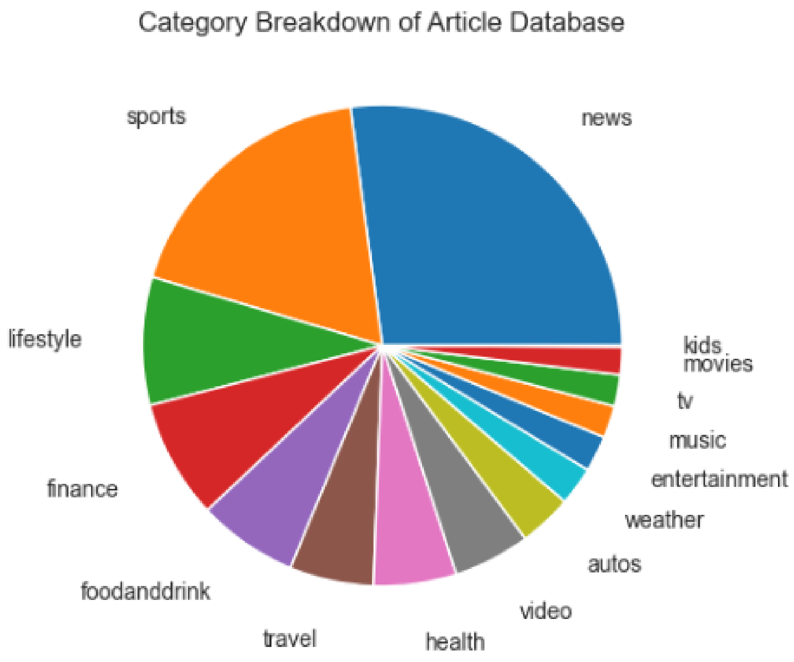


Distribution of Title & Abstract Word Counts

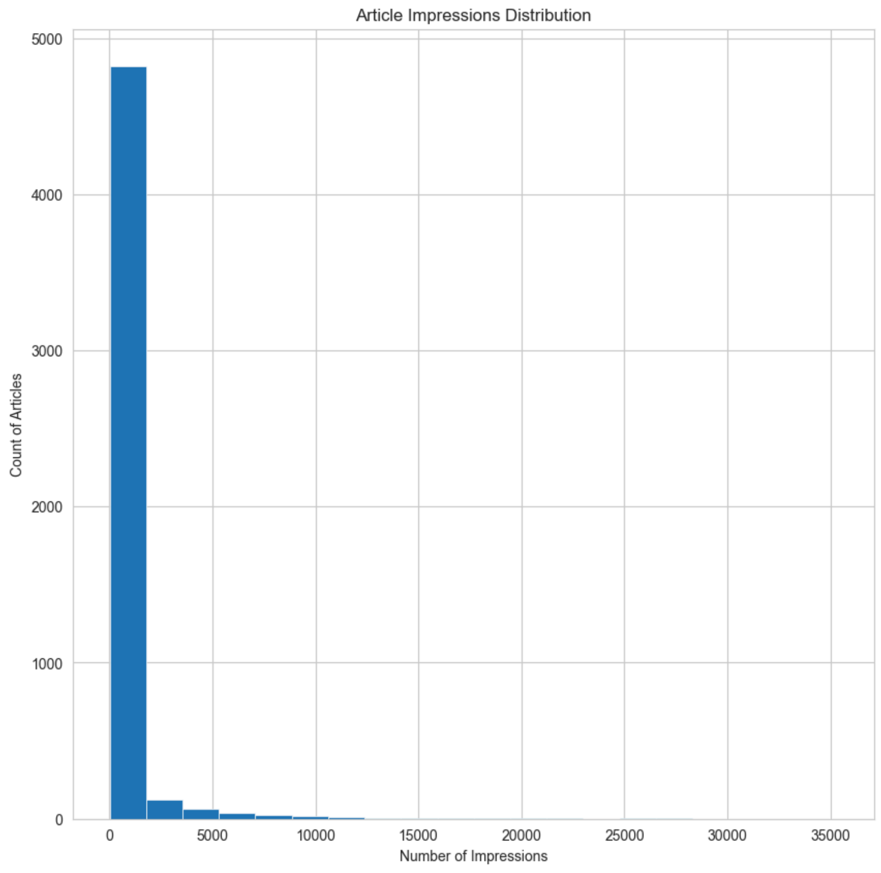




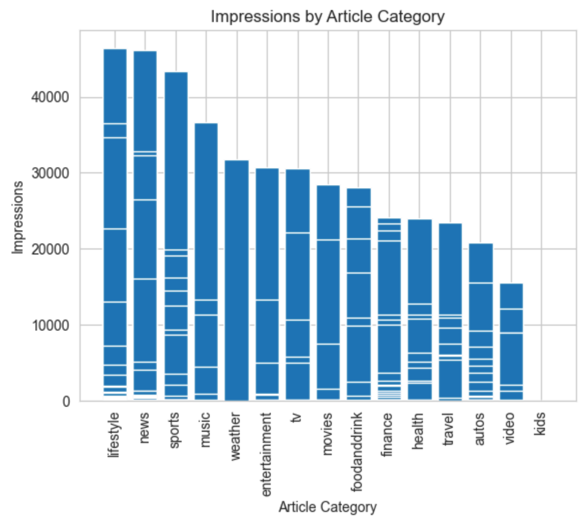
Article Database Category Breakdown



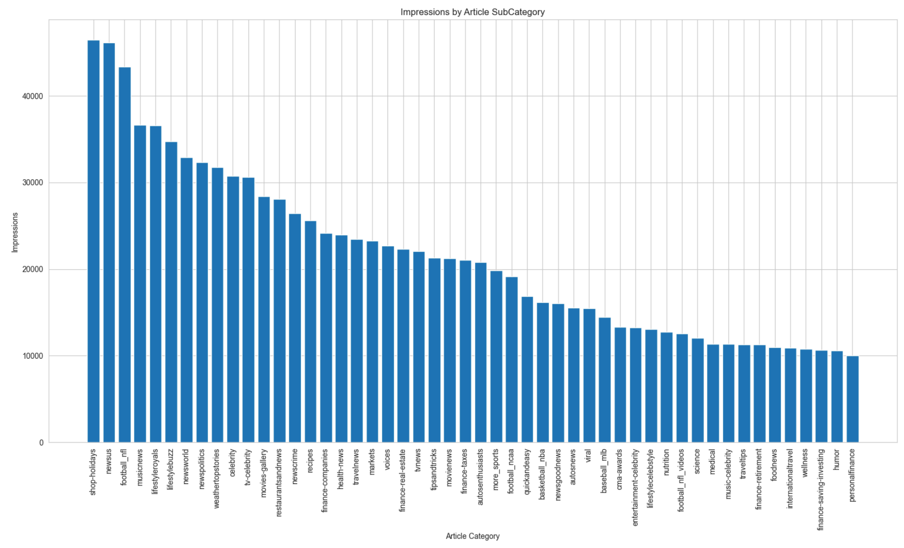
Article Impressions Distribution



Impressions by Article Category



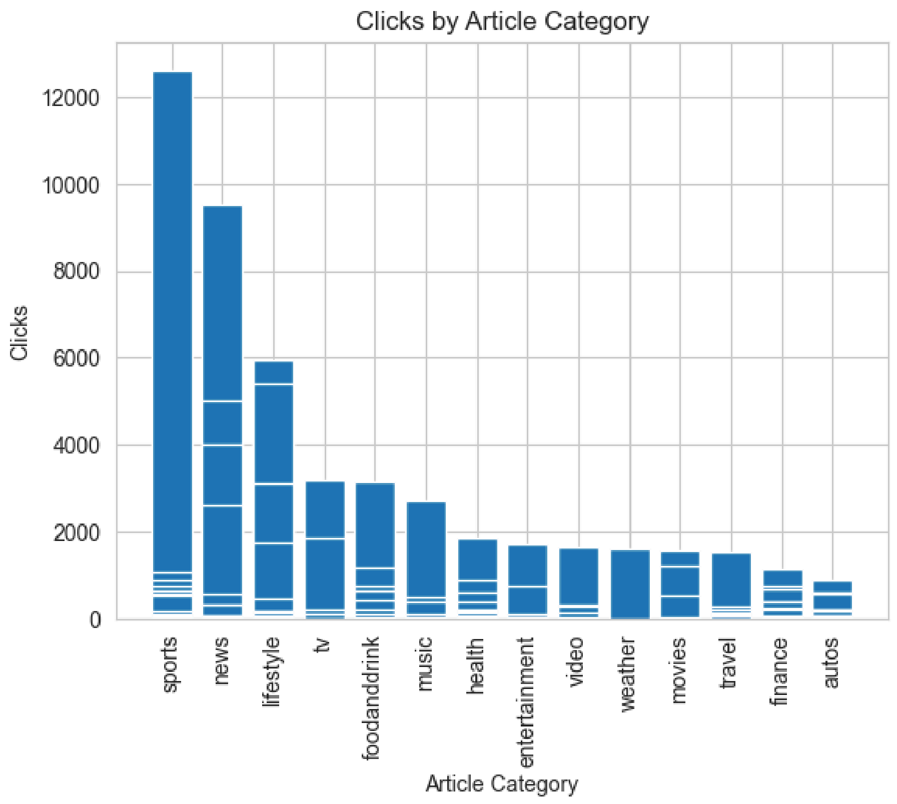
Impressions by Article SubCategory



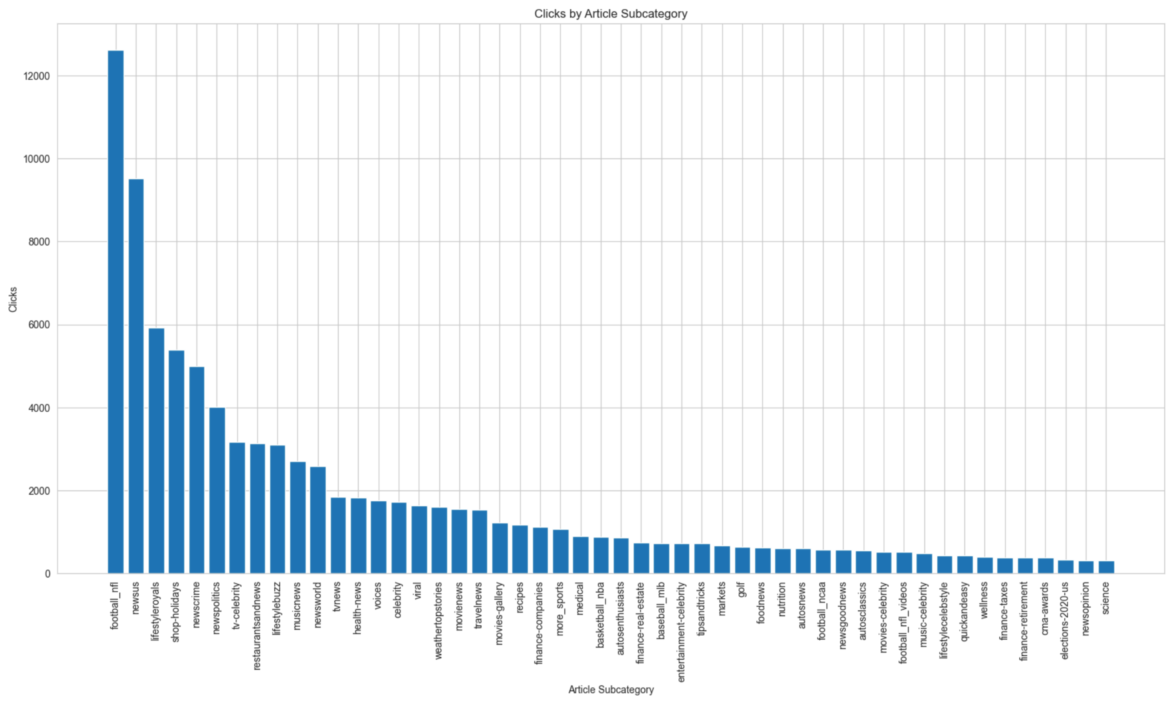
Top 10 Clicked Articles

	candidate_news_id	title	user_id	num_clicks
0	N31958	Opinion: Colin Kaepernick is about to get what...	7993	7993
1	N36779	South Carolina teen gets life in prison for de...	4649	4649
2	N5940	Meghan Markle and Hillary Clinton Secretly Spe...	4178	4178
3	N20036	30 Best Black Friday Deals from Costco	3719	3719
4	N30290	The Real Reason McDonald's Keeps the Filet-O-F...	2847	2847
5	N24802	3 Indiana judges suspended after a night of dr...	2232	2232
6	N5472	Report: Police investigating woman's death aft...	2129	2129
7	N53572	Taylor Swift Rep Hits Back at Big Machine, Cla...	1806	1806
8	N11390	Police find 26 children behind false wall at C...	1709	1709
9	N42844	Survivor Contestants Missy Byrd and Elizabeth ...	1611	1611

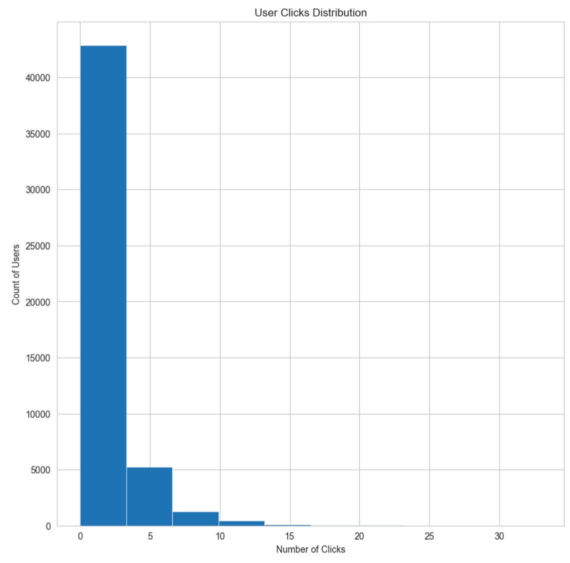
Clicks by Article Category



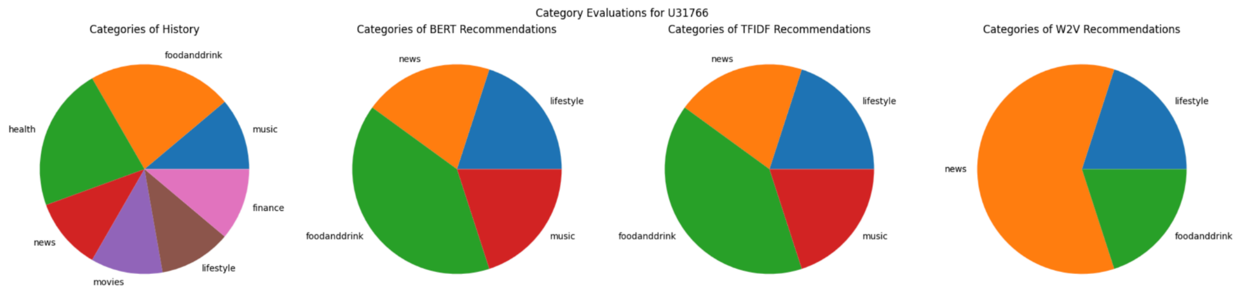
Clicks by Article Subcategory



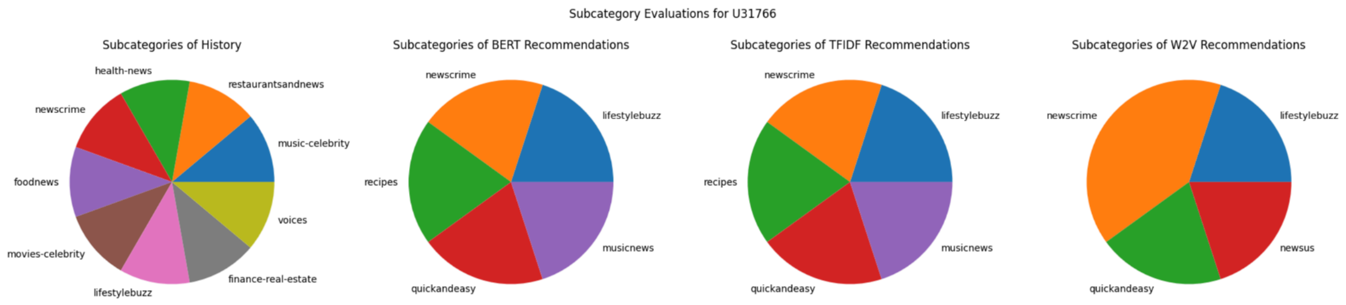
User Clicks Distribution



d. Recommendation Results EDA



Comparing categories of user history articles and the top 5 recommended articles per model for a random user



Comparing subcategories of user history articles and the top 5 recommended articles per model for a random user

3.4 Building the Recommender System

To build a content-based recommender system, we needed to compare an item profile and a user profile to recommend items to a user that are similar to items that they have previously interacted with. In our case, the items are news articles, and news articles that users have previously interacted with can be found in the user's click history.

1. Item Profile: First, we built a profile for each news article
    - The most important features of a news article are Category, Subcategory, Title, and Abstract
    - We one-hot-encoded Category and Subcategory to get binary vectors.
        - Might have looked something like: [0, 1, 0, …, 0, 0]
    - We used TFIDF, Word2Vec, and a BERT model to get vectors/embeddings for Title and Abstract.
        - Might have looked something like: [0.1, -0.36, 0.19, 0.7, -0.03, ..., 0.05]
    - To get the final item profile vector, we normalized each vector and then concatenated the vectors.
        - Might have looked something like (before normalization):

[0, 1, 0, …, 0, 0,	Category
0, 0, 1, …, 0, 0,	Subcategory
0.1, -0.36, 0.19, 0.7, -0.03, ..., 0.05]	Title + Abstract

        - We ended up with three item profile vectors, each constructed using a different method: TFIDF, Word2Vec, and BERT.
2. User Profile: Second, we built a user profile that represents user preferences and behavior
    - We used the clicked news IDs from behaviors.tsv (History column)
        - For each clicked article, we got its vector embedding from the item profile (for TFIDF, Word2Vec, and BERT, respectively).
        - We then averaged the embeddings of each clicked article in the user's history to get a single user profile vector (for TFIDF, Word2Vec, and BERT, respectively).
3. Third, we compared the user profile vector to the candidate article embeddings to assess similarity. We computed cosine similarity between the user profile vector and the candidate article embeddings, and then ranked all candidate articles by similarity score (for TFIDF, Word2Vec, and BERT, respectively). The candidate articles are recommendations contained in the Impressions column. The Impressions column includes a list of news displayed in the particular impression and users' click behaviors on them (1 for click and 0 for non-click).
4. Fourth, since we have ranked articles and relevance measures given by the Impressions column (1 for click and 0 for non-click), we calculated offline evaluation metrics to assess system performance (for TFIDF, Word2Vec, and BERT, respectively).
5. We proceeded through steps 1-4 for both the train and validation sets.

4 Evaluation

Our recommendation system was trained on the training set and evaluated on both training and validation splits using a range of offline evaluation metrics. When calculating the metrics, we skipped samples with less than three impressions in order to not consider small samples, and we skipped impressions with no clicks. Since the MIND dataset provides impressions with ground-truth relevance labels derived from user click logs, we were able to utilize the following metrics:

- ROC AUC: measures a model's ability to distinguish between classes
- Mean Reciprocal Rank (MRR): position of the first relevant recommendation
- nDCG@K: evaluates ranking quality by giving more weight to relevant items that appear higher up on the list
- Recall@K: fraction of relevant items retrieved at K
- Precision@K: fraction of retrieved items that are relevant at K
- MAP@K: overall ranking precision across positions at K

Results Summary:

	Train			Val		
	TFIDF	Word2Vec	BERT	TFIDF	Word2Vec	BERT
ROC AUC	0.5986	0.5978	0.5952	0.5506	0.5488	0.5612
MRR	0.3094	0.2989	0.3024	0.2880	0.2825	0.2892
nDCG@5	0.2846	0.2777	0.2797	0.2649	0.2593	0.2689
nDCG@10	0.3455	0.3393	0.3407	0.3209	0.3154	0.3244
Recall@5	0.4090	0.4080	0.4074	0.3747	0.3697	0.3853
Recall@10	0.5848	0.5854	0.5832	0.5357	0.5316	0.5445
Precision@1	0.1564	0.1412	0.1471	0.1503	0.1440	0.1464
Precision@5	0.1044	0.1037	0.1035	0.0973	0.0956	0.0999
Precision@10	0.0773	0.0772	0.0768	0.0717	0.0709	0.0728
MAP@10	0.2557	0.2478	0.2503	0.2394	0.2337	0.2410

Key Finding #1: BERT achieves the best performance

- BERT achieved the highest percentages across all metrics on the validation set. BERT has the strongest generalization on unseen data when compared to TFIDF and Word2Vec.
- It is likely that BERT's deep contextual embeddings capture the nuance of semantic relationships between articles that keyword or static vector approaches might miss.
- Because BERT takes the word meaning based on the surrounding context into consideration, it can relate different phrasings and synonyms to the same or similar concepts, which makes it particularly suited for diverse news articles. This allows it to recommend articles that are topically relevant even when the exact keywords differ, which is a major advantage for unseen articles in the validation set.

Key Finding #2: TFIDF achieves surprisingly strong performance

- Even though TFIDF is a simpler, sparse, and purely frequency-based approach, it scored 0.5506 ROC AUC on validation, which is only slightly behind BERT (0.5612) but better than Word2Vec (0.5488).
- This stronger performance could be due to the fact that news article content has high keyword-topic alignment, which means that readers often click articles containing similar keyword patterns as the ones they clicked before.
- TFIDF can remain competitive against more complex embedding methods when computational resources are limited.

Key Finding #3: Word2Vec needs further tuning

- Word2Vec consistently underperformed compared to TFIDF and BERT in both train and validation metrics.
- This weaker performance could be due to suboptimal hyperparameters (e.g., vector dimensionality, context window size, minimum word frequency threshold), training on a limited corpus, reducing the quality of learned semantic relationships, and/or averaging of word vectors, potentially losing contextual information important for news relevance.
- Future work could involve further tuning with larger vector sizes, different context windows, pre-trained Word2Vec models, etc.

Key Finding #4: Performance gap between train & validation set is acceptable

- The training set performs better than the validation set across all models, an expected outcome when moving from seen to unseen data.
- This difference of at most 5% is not large enough to indicate severe overfitting since our system uses cosine similarity ranking rather than direct supervised prediction. This means the embeddings are not trained to optimize for validation labels directly, so the concept of overfitting has less significance here.
- Overall, all models generalize well when applied to the validation set; this demonstrates stability of performance.

Key Finding #5: MRR suggests first relevant hit is in top 3-4

- MRR values around 0.28-0.30 suggest that on average, the first clicked and/or relevant article appears within the top 3-4 ranked recommendations.
- Finding a relevant article in the top 3-4 ranked recommendations is good for users.

Key Finding #6: nDCG@10 performs better than nDCG@5

- Across all models, nDCG@10 > nDCG@5, which indicates that the ranking quality improves when considering more positions.
- nDCG rewards systems that rank highly relevant items near the top. The higher nDCG@10 suggests that while the top-5 ranking is decent, relevant items are often still placed within positions 6-10, which the broader cutoff captures.

Key Finding #7: Recall@10 performs better than Recall@5

- Recall improves considerably from K=5 to K=10, which means that relevant items are often still placed within positions 6-10, which the broader cutoff captures.
- Recall@K measures the fraction of relevant items retrieved at K, and allowing the model to return more recommendations (top 10 instead of top 5) naturally increases the likelihood of including relevant items in the retrieved set.

Key Finding #8: Precision@1 > Precision@5 > Precision@10

- Precision@1 values (~0.14-0.15) indicate how often the very first recommendation is relevant.
- Precision@K equates the number of predicted and recommended relevant items as K. As expected, precision decreases as the number of recommendations increases because the denominator grows while the number of relevant items does not necessarily keep pace.
- Precision@K is not a strong metric to evaluate our system since it assumes exactly K recommendations are always shown and that they are equally important.

5 Conclusion

In this project, we explored content-based recommendation systems for news articles using three different text representation techniques: TFIDF, Word2Vec, and BERT. BERT, a contextual embedding model, achieved the best performance across all models. Meanwhile, TFIDF, despite its simplicity, delivered strong and interpretable results with minimal computational resources. Word2Vec underperformed compared to TFIDF and BERT in both train and validation metrics; further tuning for the Word2Vec model is needed.

Offline evaluation allowed us to assess recommendation quality. By focusing on precision, recall, and ranked retrieval-oriented metrics, we not only gained clearer insights into model behavior and trade-offs, but we also learned much about the nuances behind the theory and calculation of these metrics.

By focusing on building a simple content-based recommender system with basic text representation techniques, we have built a fundamental understanding of the complexities of news recommendation systems. Ultimately, this understanding serves as a basis for progressing to more complex approaches in future work.

Future Work:

- Hyperparameter tuning for TFIDF and Word2Vec
- Work with MIND-large dataset
- Utilize entity_embedding.vec & relation_embedding.vec
- Explore hybrid recommendation system approaches
- Integrate temporal considerations to account for news freshness and user interest shifts over time
- Address the cold start problem
- Consider evaluation such as: Catalog Coverage, User-space/Item-space Coverage, Diversity, Novelty, Serendipity, Expected Reciprocal Rank (ERR)
- Research news recommendation using advanced deep learning techniques such as those mentioned in the MIND research paper (i.e., multi-view neural networks, transformer-based architectures, user behavior modeling, personalized news encoders, attention mechanisms, etc.)

6 References

Fangzhao Wu, Ying Qiao, Jiun-Hung Chen, Chuhan Wu, Tao Qi, Jianxun Lian, Danyang Liu, Xing Xie, Jianfeng Gao, Winnie Wu and Ming Zhou. MIND: A Large-scale Dataset for News Recommendation. ACL 2020.