# News Recommending System Based on Collaborative Filtering

**Shengnan Duan (elenore@umich.edu), Xinhao Liao (theoliao@umich.edu)**

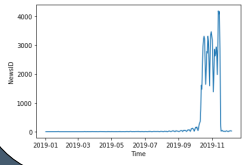
**In response to SI 671 Data Mining final project in 2020 Fall term**

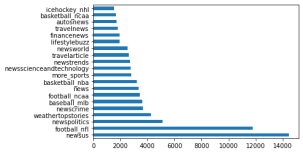
## Introduction

**Overview**: This paper designed a news recommender system mainly based on the item-item collaborative filtering and matrix completion techniques. The system is integrated with real-time scraping, user interest decay and web-front demonstration.

**Data Exploration**

There are 101,521 unique news and 711,222 users in the current database, who triggered 73,629,868 clicking events. The time range of data and the popular news categories are shown below:

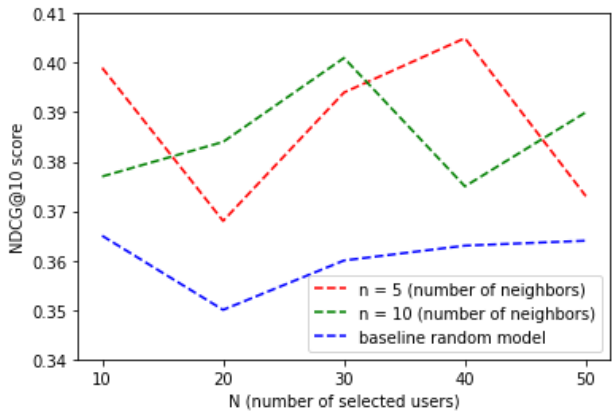



## Methodology and Results

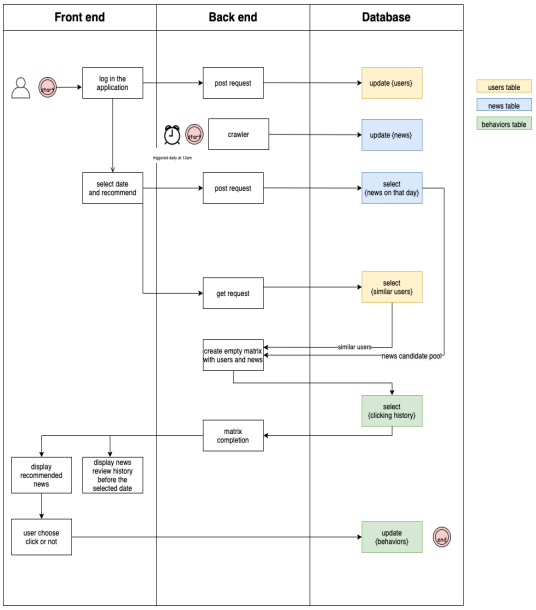
Item-item collaborative filtering is applied for normal cases where matrices are constructed for news clustered into categories and some users sampled. Ratings are estimated based on behavior records of users and similarities between clusters. The time decaying effect is also considered. For cold-start cases, with no user information we rank the candidate news by the popularity. Our model generally gives results significantly better than a baseline random model as shown below,

| | | Users | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| News | 1 | 78.80 | | 72.90 | 87.70 | 80.90 | | 80.60 | | 25.90 | 55.20 |
| | 2 | 73.00 | | | 73.70 | | 73.70 | 94.20 | 28.20 | | |
| | 3 | | | 66.80 | | 7.30 | | | 84.70 | 78.30 | |
| | 4 | 41.90 | 59.50 | | 15.30 | ? | 87.30 | | | | |
| | 5 | | 93.50 | | | | | | 46.40 | 64.10 | |
| | 6 | 40.30 | | | 88.00 | 21.80 | 67.10 | 70.40 | | | 83.50 |
| | 7 | | | 47.50 | | | | 99.50 | | | |
| | 8 | 75.80 | 39.40 | | 4.00 | 27.90 | 91.30 | | 28.90 | | 39.20 |
| | 9 | | | | 64.30 | | 42.10 | | | | |
| | 10 | 4.40 | | 24.90 | | 57.90 | 36.40 | | 2.10 | | 32.90 |

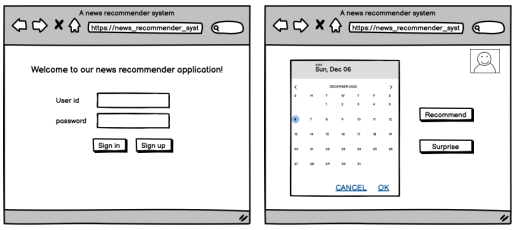


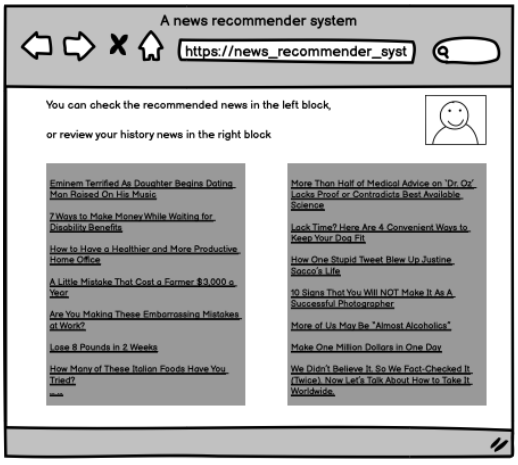

## Application Architecture

- Front end: HTML, JavaScript
- Back end: Python, Flask, SQL
- Database: Sqlite3, SQL




## User Interface Design



# News Recommending System Based on Collaborative Filtering

**Shengnan Duan**
School of Information
University of Michigan
elenore@umich.edu

Xinhao Liao
School of Information
University of Michigan
theoliao@umich.edu

## Abstract

With the increasing access to internet, more and more people switch from traditional newspaper to online platforms for news feed. This paper designs a news recommender system mainly based on the item-item collaborative filtering and matrix completion techniques. Furthermore, the system is integrated with real-time scraping, user interest decay and web-front demonstration. We are aiming to simulate a practical news recommender application that can be put in application. The study is based on MIcrosoft News Dataset (hereafter MIND), of which the user behavior data is used as ground truth and random ranking is used as the baseline to evaluate the system. The results show our approach could generate higher-quality recommendations compared with a random model. Although limitations existed, we pointed how the future improvements can be made.

Keywords: news recommender, collaborative filtering, similarity measurement

## 1 Introduction

With the increasing access to internet, more and more people switch from traditional newspaper to online platforms for news feed. This paper designs a news recommender system mainly based on the item-item collaborative filtering (hereafter CF) and matrix completion techniques. Furthermore, the system is integrated with real-time scraping, user interest decay and web-front demonstration. We are aiming to simulate a practical news recommender application that can be put in application. Inspired by Wu et.al (2020), we get the chance to start this study on the large-scale and high-quality English news dataset with detailed user clicking behaviors.

The paper will be structured as follow:

- Related work: Existing techniques and challenges of news recommender system will be discussed. The study will apply some of these techniques while try to solve some of the challenges.
- Data description: An introduction of the dataset will be demonstrated along with the data exploration analysis. The pre-processing and feature selection will also be included in this section.
- Architecture: Although the paper is primarily focused on the recommender system, we spend fair amount of effort to simulate a OLTP database and user interface. Therefore, the overall architecture of the application and the possible future roadmap will be introduced.
- Methodology of recommendation module: In this section, we will deep dive into a) the design of item-item collaborative system b) thoughts about cold start and c) evaluation methods. Challenges we encountered during the process will also be discussed.

- Results and analysis: We will show both recommending results compared with baseline and the web-end interface.
- Limitation and future improvement: Due to the time restriction, we believe there are some improvements could be made in the future. We will also discuss the potential risks of assumptions when building the model.
- Conclusion: The high level of our findings and future goal will be discussed.

## 2 Related works

One major purpose of building MIND dataset is to help solving some of traditional challenges news recommender system researches facing: a) timeliness nature of news is hard to tackle and brings a more severe cold-start problem than traditional recommender applications b) the exploration of nature language processing (NLP) of the news contents has few attentions and c) user behaviors are rather implicit compared with the more popular user-generated-content based collaborative filtering researches (Wu et al. (2020)). Raza and Ding (2020) introduced another challenge in terms of ethic that a highly personalized recommender system might limit the number of varieties of news user has exposure to. In that case, news will be just another kind of commodities and only sold to certain target group.

Raza and Ding (2020) address that to solve the timeliness problem, time-decay model is one of the conventional techniques that designed to give more weight to recent news and less weight to outdated news. We will use similar strategy in our study to not only cope with the timeliness but also user interest decay issue. Another solution is graph-based algorithm that treats user's reading behaviors as sequential data. However, this approach might fail to function in the large complex where there are many dynamic patterns. The third approach is popularity-based model, which is the easiest to implement. One obvious problem is user's access to news is biased. We will use this approach only to handle the user-cold start scenarios. Although Karimi et al. (2018) introduce a common approach that to extract user's history, location, social network etc. as additional features, we are afraid the potential privacy risks and would rather to go with popularity-based method. For item cold-start, content-based recommendation system is proven to solve a major part of the problem and it can lead to an alternative method that mix content analysis and user profile analysis Karimi et al. (2018).

The content-based recommendation system is also helpful to give more exploration to NLP techniques. In Raza and Ding (2020)'s discussion of deep learning models, we learned the TFIDF, hashing methods and topic modeling are often used for text data feature engineering. However, the conventional NLP methods focus on the counting statistics instead of the sequence of the words and make assumption of linear relationships. While deep learning models could solve some of the limitations, due to the time constrain, our study will not implement NLP methods.

Lastly, I don't think user's implicit behavior is necessary a huge challenge. Unlike CF on movies where user need to give explicit ratings, user's click behavior on news can somehow reveal their actual 'ratings'. News title and abstract are normally visible to users before clicking behaviors happened, and both of which are high-level summary of the news. Users firstly read the summary and decide to be interested. If interested, they choose to click. As for movie, users don't have access to the stories design, filming style, special effects, hidden clues unless they watch it (assuming no research on movies from outside data source). They only know if they are interested after clicking. Our assumption is made based on the news has qualitied title and abstract. If the assumption fails, then a user behavior modeling work need to be conducted.

## 3 Data Description

MIcrosoft News Dataset (MIND) is a large-scale dataset for news recommendation research. It was collected from anonymized behavior logs of Microsoft News website. The mission of MIND is to serve as a benchmark dataset for news recommendation and facilitate the research in news recommendation and recommender systems area. MIND contains about 160k English news articles and more than 15 million impression logs generated by 1 million users. Every news article contains rich textual content including title, abstract, body, category and entities. Each impression log contains the click events, non-clicked events and historical news click behaviors of this user before this impression.

Since the original data doesn't have news's time information, we scraped the urls provided in data to add the time feature. We believe it's necessary to solve the timeliness problem. Also, the data provided the 100-dimensional embeddings of the entities and relations learned from the subgraph (from WikiData knowledge graph) which will not be included in this study. Therefore, the database of application is structured as below:

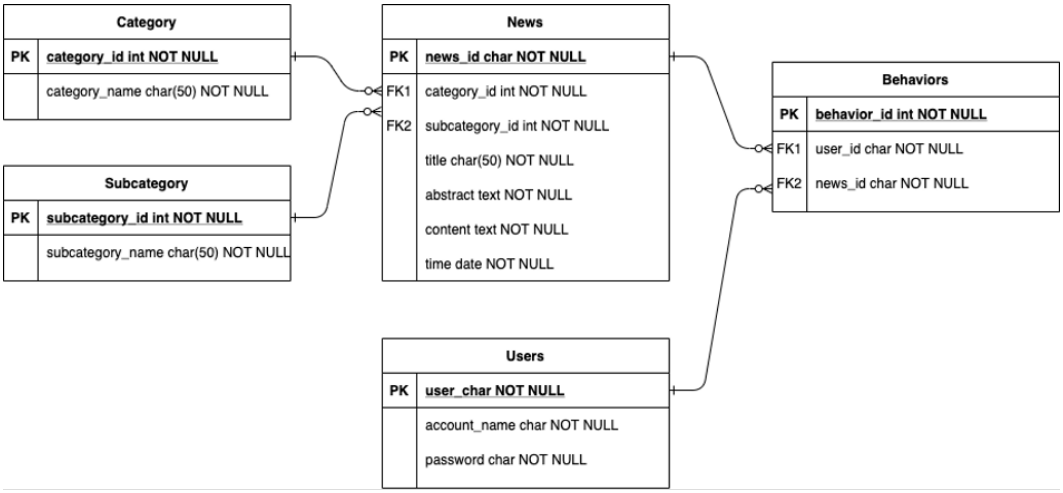



Figure 1: Database design

Since back-end programming is not our main focus here, we listed 'account name' and 'password' as example columns for 'users' table, which can be potentially expanded for future development. Similarly, the current 'behaviors' table is only used to build relationship between user and news (browsing history). The major changes between original data and current database are as below:

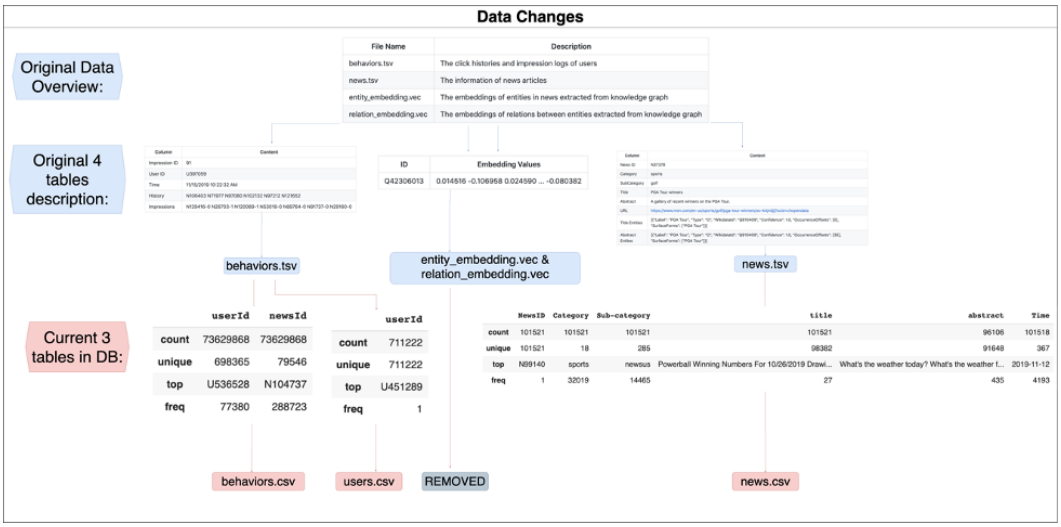



Figure 2: Data structure changes with original MIND data

There are 101,521 unique news and 711,222 users in the current database, who triggered 73,629,868 clicking events. There aren't many null values in the dataset, we only removed 3 missing news time.

There are 18 news categories and 285 subcategories. Sports and general news are the 2 most popular categories, and if we look at the subcategory distribution, more specific topics which have the largest audience will appear such as U.S news, politics, weather, crime and more detailed sports preference like football, baseball and basketball.

Next, we dig into both title and abstract and found abstract mostly have general words while title mostly uses eye-catching words like 'trump' 'will' 'win'. Furthermore, by extracting the overlapping

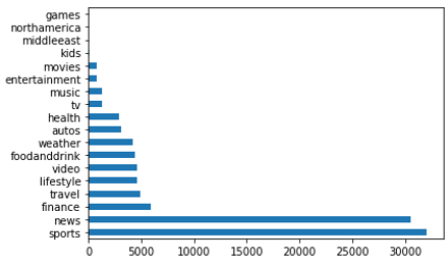



Figure 3: Distribution of Categories

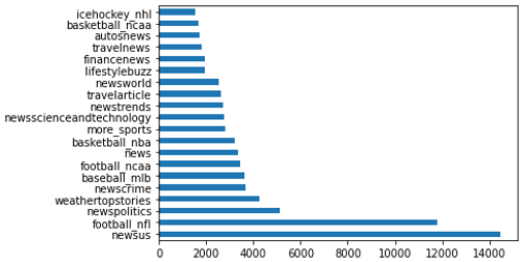
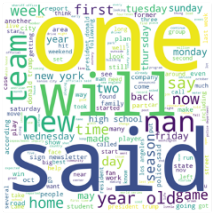
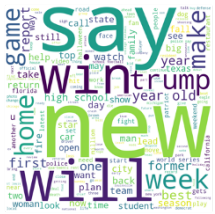



Figure 4: Distribution of Sub-Categories

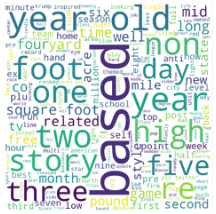
words, abstract has a more uniform distribution of frequent words except for word 'based', on the contrast, title's thesaurus has a skewed distribution where a few words have very high frequency ('ex', 'anti', 'post') and the rest have low frequency. Lastly, we have a look at the news trend over time. The data is very skewed that most of news are from 2019, especially from October to November.




(a) abstract




(b) title




(c) abstract after extraction


(d) title after extraction

Figure 5: Examples hand signal images our classifier succeeded and failed at classifying.

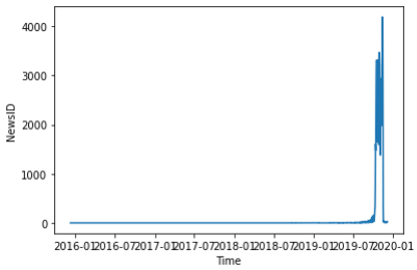



Figure 6: News distribution

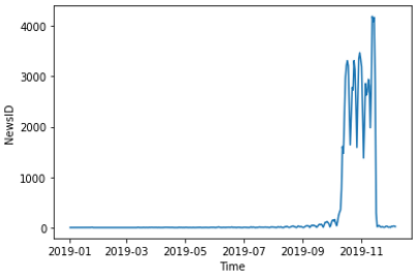



Figure 7: News distribution in 2019

## 4 System Architecture

There are 3 modules in our system: front end, back end and database. Technology stack we chose is as below:

- Front end: HTML, JavaScript
- Back end: Python, Flask, SQL
- Database: Sqlite3, SQL

The application has 2 starting points: on front end, user can start the app by logging in; on back end, a crawler is scheduled to scrap latest news daily at 12am. Users table and news table will be updated based on the 2 start events. Then, user can choose a date to for news search, by doing which, a post request will be triggered and retrieve the news of that day. The retrieving results will be put in the candidate pool. If the news amount is too large, we can filter this step further by news popularity.

Then, the **candidate pool** will be used to feed the **recommender module** as the news source; on the other side, the system will select users from users table and create the matrix for the item-item collaborative filtering method applied in **recommender module** where the columns are users and rows are subcategories (which will be dicussed in detail in section 5). After the **recommender module** computes the results, the system will send the results back to front end, where the user can see his/her news review history in the right block and news recommendations in the left block. The news links are provided and the clicking behaviors are recorded and updated in the database accordingly for future use. The design diagram is listed as below:

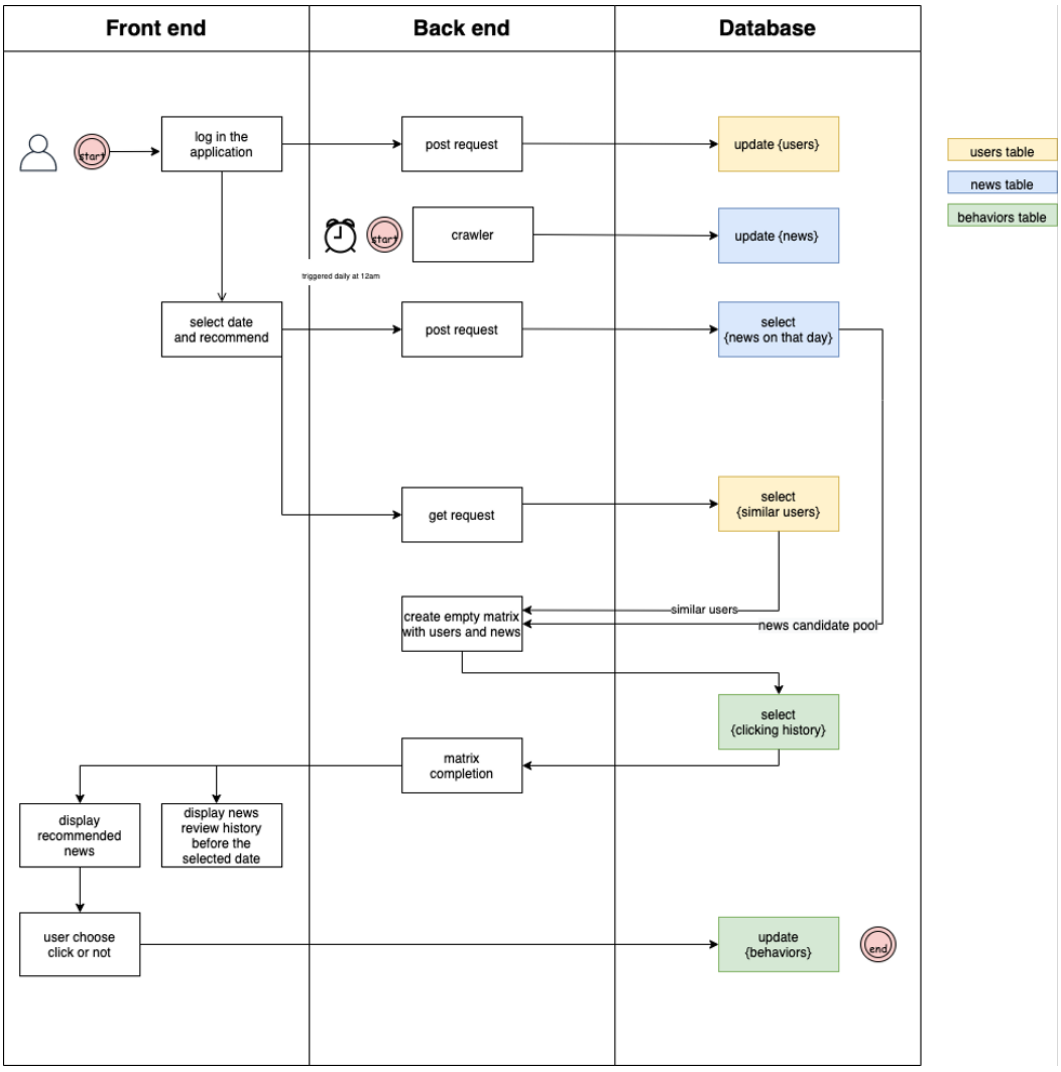



Figure 8: System architecture

## 5 Methodology

In this section, we will discuss the details of the recommender module and how we can evaluate its performances. The input of the recommender model involves the a pool of candidate news and the behavior records of the user, while the output is the list of candidate news reordered based on estimated ratings.

### 5.1 Item-item Collaborative Filtering Method

For the normal cases, we are going to recommend using the item-item CF which is a matrix-based method. With so many news and users existing, dealing with a matrix with every news corresponding to a row and every user corresponding to a column would be infeasible.

As for users, besides the user who is recommended for, $N-1$ other users are randomly selected for collaborative filtering. (Notice that clustering users might be another good option if there are enough extra user profile data available as shown in Das et al. (2007).)

Also noticing that news are already clustered into different categories and subcategories, and that news in the same category and subcategory are similar enough to some extent, it's natural to have each row corresponding to every tuple of category and subcategory. With as many as 308 tuples of Categories and Sub-categories (involving 18 categories and 285 sub-categories), users tend not to have enough records on every tuple and then ratings need to be estimated based on existing ratings on other similar tuples. Actually, those other tuples of category and subcategory with no news viewed by any selected user do not need to be considered since they would not contribute to the estimation in terms of the collaborative filtering method.

For the value in a cell of the matrix at row $c$ and column $c$, it represents user $u$'s rating on (Category, Sub-category) $c$ and is calculated based on user $u$'s behavior records with the effect of time decay considered as follows

$$r_{u,c} = C \cdot \frac{\sum_{t=0}^{\infty} count(u,c,t) \cdot d^t}{\sum_{k \in K} \sum_{t=0}^{\infty} count(u,k,t) \cdot d^t}$$

where $K$ is the set of all tuples of (Category, Sub-category), $count(u,k,t)$ represents the number of news in (Category, Sub-category) $k$ that user $u$ has viewed during the last $(t+1)$-th month, $d = 0.8$ is the decay rate, and $C = 100$ is a constant here. All other cells corresponding to unviewed subcategories can be labelled 0. In the assumption that all users share a roughly equivalent amount of overall enthusiasm for all the news, such normalized values implies how much of the user's interest is placed on a certain category and subcategory. The decay rate is used to reflect the fading of the user's enthusiasm for a certain area.

| | | Users | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| News | 1 | 78.80 | | 72.90 | 87.70 | 80.90 | | 80.60 | | 25.90 | 55.20 |
| | 2 | 73.00 | | | 73.70 | | 73.70 | 94.20 | 28.20 | | |
| | 3 | | | 66.80 | | 7.30 | | | 84.70 | 78.30 | |
| | 4 | 41.90 | 59.50 | | 15.30 | ? | 87.30 | | | | |
| | 5 | | 93.50 | | | | | | 46.40 | 64.10 | |
| | 6 | 40.30 | | | 88.00 | 21.80 | 67.10 | 70.40 | | | 83.50 |
| | 7 | | | 47.50 | | | | 99.50 | | | |
| | 8 | 75.80 | 39.40 | | 4.00 | 27.90 | 91.30 | | 28.90 | | 39.20 |
| | 9 | | | | 64.30 | | 42.10 | | | | |
| | 10 | 4.40 | | 24.90 | | 57.90 | 36.40 | | 2.10 | | 32.90 |

Figure 9: A matrix example

With the ratings computed in this way, we can create matrices as shown above and predict a user's rating on a tuple (Category, Sub-category) based on his ratings on the $n$ most similar neighbors (evaluated with the cosine similarities with the target tuple's vector). Then the estimated rating of user $x$ on target the tuple of (Category, Sub-category) $i$ is

$$r_{xi} = b_{xi} + \frac{\sum_{j \in n(i;x)} S_{ij} \cdot (r_{xj} - b_{xj})}{\sum_{j \in n(i;x)} S_{ij}}$$

where $n(i;x)$ represents $n$ tuples of (Category, Sub-category) that are most similar to $i$ and are rated by user $x$, $S_{ij}$ is the cosine similarity between tuples $i$ and $j$, $r_{xj}$ is the rating of user $x$ on $j$, and $b_{xi}$ is the baseline estimate for $r_{xi}$ and can be computed as follows

$$b_{xi} = \mu + b_x + b_i$$

where $\mu$ is the overall mean rating (value of cells in the matrix), $b_x$ is the rating deviation of user $x$, and $b_i$ is the rating deviation of tuple $i$.

After the user's ratings on different tuples of (Category, Sub-category) are estimated, the candidate news can be ranked accordingly. For news in the same category and sub-category, those viewed more would be ranked higher.

### 5.2 Cold-start Problem

When a new user comes, the item-item collaborative cannot be applied with no existing rating to any tuple of that user. Since there is no extra information about the user available from the dataset here, the candidate news are directly ordered by the popularity (i.e. how many times they are already views) in this case. Timeliness is not taken into account since all candidate news are chosen within the day and already considered timely enough.

### 5.3 Evaluation

The recommender module gives a ranking of all candidate news, and some of them will finally be clicked by the user (labelled 1) while the others will be not (labelled 0). With these ground truths available in $behaviors.tsv$ of MIND dataset, we can evaluate the rankings with NDCG@10.

By tuning the hyperparameters like $N$ (i.e how many users are randomly selected to create the matrix) and $n$ (i.e how many neighbors of tuples are used to estimate a rating), we can results corresponding to different hyperparameters with each other and with the result of a baseline model that always returns a random ranking of the candidates.

## 6 Results

### 6.1 Recommendation Results

The recommender module is evaluated on 500 records from the $behaviors.tsv$ of the dataset with different hyperparameters tested. The results are compared with each other and with the result of a baseline random model as shown below.

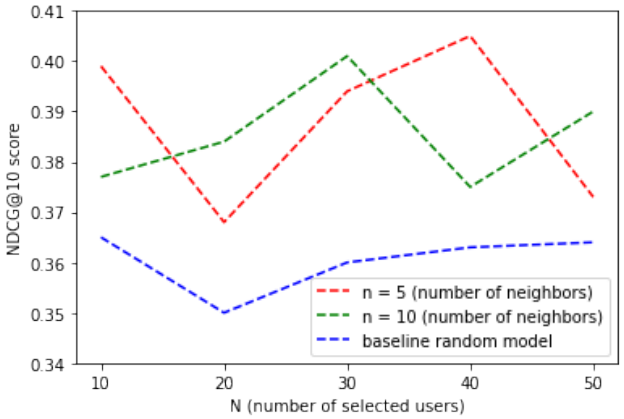
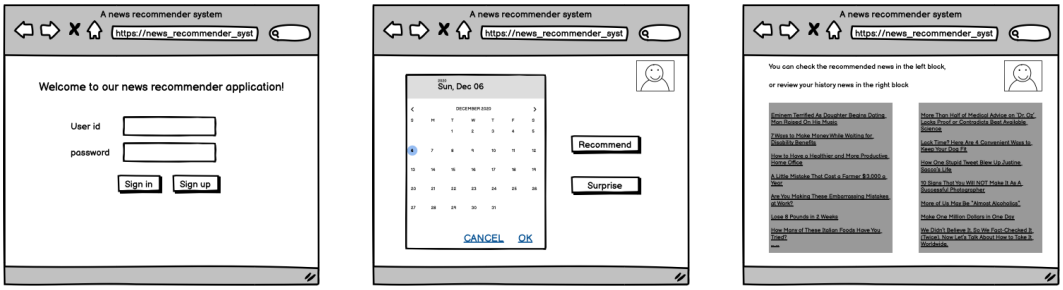



Figure 10: Results comparison

As we can see, the recommender module generally gives a result better than the baseline random model. However, different tested hyperparameters do not seem to show a significant difference on the results. Although we do expect to see some trend with the parameters (e.g. we might expect to have a better result when data of more users are applied), such potential differences might not occur unless a significant larger value is tested, which is infeasible with so much computation involved.

### 6.2 User interface

The current user interface is rather simple, and only support log in, select date & recommend and results page.




(a) user log in (b) select date and recommend (c) results page

Figure 11: User interface design

Notice that in a real system, we do not need to select a date and are only recommended with up-to-date news. However, with only limited outdated data given in the dataset, date selection is enabled in the demo system so that we can explore recommendation results for different users at different time points.

Our final demo page is shown below and some steps are simplified here. For instance, users do not need to actually register and login in the demo system. Histories will be fetched if the given user ID is found. Otherwise, a new user with that will be automatically created. All the stuff are organized in a single page just for demonstration.


Figure 12: Final user interface

## 7 Discussion & Roadmap

### 7.1 System Acceleration

One limitation of the demo system is that the matrix is updated every time a user select a new date, which will trigger a new candidate pool and therefore change the matrix structure. Ideally, we want to obtain a fixed amount of tuples of (category, Sub-category) and an optimized cluster of users to form a matrix, which will be updated daily. This can be achieved in a real-life system where a table of users' history on different categories and sub-categories in different month is maintained and only

data within a certain time period (e.g. a year) need to be considered. In this way, there is no need to fetch and count records for every user and tuple of category and sub-category every time so that the system can be significantly accelerated.

### 7.2 Candidates Selection

The news candidates pool needs to be small otherwise users might always be recommended with many news in the same category and subcategory. Although in the MIND dataset it's not a problem since only limited number of news are given every day, it could be a problem in a real-life large-scale system. In this case, we might filter the candidate news in advance based on a weighted sum of the timeliness (i.e. how long it has been since the news happens) and the popularity (how many time the news has been viewed) of the news to maintain a relatively small pool of candidate news.

### 7.3 Rating Estimation

Even for a tuple of (category, Sub-category) that is already viewed by a user, the rating is always estimated based on the user's calculated rating on the neighboring tuples (including itself as the most similar neighbor) and the similarities. Otherwise, those tuples with limited number of views by the user would tend not to be recommended if their ratings were directly calculated. Besides this, we might need even more strategies to avoid hiding certain news from certain users which might unintentionally polarize users' interests on different areas.

### 7.4 Method Improvement

As for the future improvements of the methodology, we might need to determine which parameter settings can optimize the recommending results as discussed in section 6.1. More parameters might need to be tuned in the future such as the time decay range, where we are putting a range of 0 to infinity in the current model since the data spans less than 12 months.

We also would like to try other methods including NLP word vectorization and deep learning models to fully discover the text potential. However, this requires a larger computation power. We will also keep expanding the current dataset, which only provides word embeddings for news title and abstract. A constant and timely scraping module will be designed and a bigger data storage, potential a cloud database will be involved.

## 8 Conclusion

News recommender system is a relatively new topic. Although researches were conducted since 10 years ago, public data of news articles with enriched user behavior and text contents is still in the process of establishment. Our solution proved collaborative filtering can largely improve the recommending results even without taking NLP into consideration and with the implicit user click behaviors. The limitations and improvement suggestions are made and a comparative study could be continued based on the current approach.

## References


Abhinandan S. Das, Mayur Datar, Ashutosh Garg, and Shyam Rajaram. 2007. Google news personalization: Scalable online collaborative filtering. In *Proceedings of the 16th International Conference on World Wide Web*. Association for Computing Machinery, New York, NY, USA, WWW '07, page 271–280. https://doi.org/10.1145/1242572.1242610.

Mozhgan Karimi, Dietmar Jannach, and Michael Jugovac. 2018. News recommender systems – survey and roads ahead. *Information Processing Management* 54(6):1203 – 1227. https://doi.org/https://doi.org/10.1016/j.ipm.2018.04.008.

Shaina Raza and Chen Ding. 2020. A survey on news recommender system – dealing with timeliness, dynamic user interest and content quality, and effects of recommendation on news readers.

Fangzhao Wu, Ying Qiao, Jiun-Hung Chen, Chuhan Wu, Tao Qi, Jianxun Lian, Danyang Liu, Xing Xie, Jianfeng Gao, Winnie Wu, and Ming Zhou. 2020. MIND: A large-scale dataset

for news recommendation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, Online, pages 3597–3606. https://doi.org/10.18653/v1/2020.acl-main.331.