News Recommendation System Project Report

Pavan Thanay IMT2020024
Krushikar Reddy IMT2020043
Chaithanya Reddy IMT2020054
Samarth Gattu IMT2020062
Sougandh Krishna IMT2020120

I. Introduction

In today's era of the internet, people are being bombarded with an overwhelming amount of information, especially in the form of news. With so much news content available online, it has become increasingly difficult for users to find the relevant news that they are interested in. A news recommendation system can be used to assist users in sorting through this enormous amount of news content and providing them with the most relevant and personalized news.

II. Objective

The main objective of this project is to develop a news recommendation system that can suggest relevant news articles to users based on their reading history. We have used different Machine Learning Algorithms and different NLP techniques to recommend news articles.

III. Dataset

We have used the MIND dataset for our analysis. It contains two .tsv files: behaviour.tsv and news.tsv.

- behaviour.tsv file contains the click history of the user, all the impressions shown to the user and the impression time, and the user clicks. An impression log records the news articles displayed to a user when the user visits the news page at some specific time.
- news.tsv file contains the description of each news article. News-id, URL, Category, Sub-category, Title, and Abstract are its columns.

A. Exploratory Data Analysis

- The news dataset contains 17 categories and 264 subcategories.
- 60 % of the news articles belong to either news or sports category.
- The dataset consists of the user behavior records between the period November 9, 2019, to November 14, 2019.
- Users click only approximately 4 % of the news articles shown to him.
- Approximately more than 30,000 news articles (60%) are not even displayed to the user in this period.
- Each user in the dataset has at least 17 news article clicks in this period.

IV. PREPROCESSING

- We removed the URL column from news.tsv as the links are not accessible and also removed the Impression ID column from behaviour.tsv
- Then, we replaced all the NULL/NaN values with an empty string (" ").
- Next, we concatenated the category, sub-category, Title and Abstract columns of the News dataset.
- Then we converted the words to lowercase, removed symbols, and the stop words, and applied tokenization and lemmatization.
- We created a click and non-click list for all the impressions.

V. USER and ITEM REPRESENTATIONS

A. *Item Embedding*

Item, refers to the news articles. We formed news article embeddings in 3 different ways.

- The preprocessed string, is passed into a pre-trained BERT model, from which we obtain a 384 dimension vector. This is the final item embedding.
- We treat each of the columns as a different entity and pass each of the columns into the Bert model to form 4 different 384 dimension vectors. Then we can take a weighted average of these 3 vectors, with weights being user choice. (User can give importance to a particular column which he wishes to). The obtained weighted average is the final item embedding.
- The 4 vectors obtained in the previous method will be concatenated to form a 384 X 4 dimension vector. In this method, the importance of each column is reserved.

B. *User Embedding*

There are 3 different ways in which we have got the user embedding. For all the 3 methods we have used the user's click history of the behavior dataset to obtain the embedding. Let us assume that there are $'n'$ news articles in the user's click history. Let $V_1, V_2, V_3, ..., V_n$ be their embeddings.

- The user embedding is the average of the news embedding of the click history.

$$u = \frac{\sum_{i=1}^{n} V_i}{n}$$

  By taking the simple average, we are giving equal importance to all the news articles in the click history.
- Let 'k' be defined as the average news embedding of the click history. Let us consider the weights to be the dot product of k with the news embeddings of the click history. Then we take the weighted average of the embeddings of the click history with the above-formed weights to form the user embedding.

$$k = \frac{\sum_{i=1}^{n} V_i}{n}$$

$$u = \frac{\sum_{i=1}^{n} (k.V_i)V_i}{\sum_{i=1}^{n} (k.V_i)}$$

  In this representation, we are giving more importance to those categories which are more in number in the click history.
- We formed n different vectors, by taking the weighted average, with weights, being the dot product of V_i and $V_j, \forall j = 1, ...n$. Then we take the average of the n vector to form the final user embedding.

$$V_i^{'} = \frac{\sum_{j=1}^{n} (V_i.V_i)V_j}{\sum_{j=1}^{n} (V_i.V_j)}$$

$$u = \frac{\sum_{i=1}^{n} V_i^{'}}{n}$$

  In this representation, there is the notion of Attention that is given to each news article of the click history.

For the analysis of all the developed recommendation models described later, we have used the 1^{st} type of User and news article embeddings. We can choose to use other user and news article representations depending upon the exact use case.

VI. MODELS

A. *Simple recommendation system using news and user embeddings*

Initially, the user will be asked to choose some news articles from a randomly selected list of 25 news articles. In this model, we represent each user as the average of all the news embeddings present in his click history. The click history will be updated as the user views more and more news articles. News articles are then recommended to the user based on the highest similarity between the user representation and each of the news article embedding.

One more way of providing recommendations is randomly selecting a few articles from the user's click history and for each of the news articles selected provide the news article with the highest similarity as the recommendation.

Similarity or closeness is calculated based on the L-2 norm. Lower the value of the L-2 norm between two entities, implies higher similarity or closer the entities.

Algorithm Analysis: This is a very simple model, which is not capable of giving diverse recommendations. This algorithm always gives those news articles as recommendations that are almost the same (similar) as the news articles in the user's click history.

B. *K-Means Clustering*

Similar to the previous method, users are represented as the average of all the news embeddings present in his click history. A new user can be represented as the average of all the user embeddings. Similarity or closeness used in this method is also the same as defined previously.

Clustering can be done in 3 ways:

- **Method 1 (Clustering Users):**
  After obtaining the user representations, the users are clustered using the K-Means algorithm where the loss function is the similarity between the user representations. Now for a given user, we find the cluster he belongs to and assume his temporary representation as the centroid of this cluster. Now we find the similarity between each news article and this temporary user representation (cluster centroid), the news articles with the highest similarity are given to the user as news recommendations.
- **Method 2 (Clustering News articles):**
  In this method each user is treated as a different entity, instead, we only cluster the news articles in the dataset. The news articles are clustered using the K-Means algorithm where the loss function is the similarity between the news article embeddings. Now for a given user, we assign the user to a news article cluster with highest similarity. Now a randomly selected list of 25 news articles from this cluster is given as the news recommendation to the user.
- **Method 3 (Clustering Users and News articles):**

In this method, both users and news articles are clustered. The clustering of users and news articles is done in the same way as done in Method 1 and Method 2. Now for a given user, we find the cluster he belongs to in the same way as mentioned in Method 1. Now this cluster centroid is sent as the user representation for the recommendation model used in Method 2 to assign the user to the closest news article cluster. Now a randomly selected list of 25 news articles from this cluster are given as the news recommendation to the user.

Algorithm Analysis: This algorithm (Method 3) is definitely better than the simple recommendation model used before, since it not only gives recommendation based on the user's click history, but also considers the click history of similar users, and the news articles are clustered based on the similarity instead of treating them as independent entities. This model on average takes around 30 minutes for training on a fixed number of clusters. So, to find the optimal number of clusters (both users and news article clustering) using the elbow point method, it takes a lot of time for training. Since in news applications the user representation or click histories must be frequently updated, this model is not feasible in practice.

C. *Trending*

Trending news articles recommendations mainly depend on the publication time of the news article, number of views/clicks for the news article and geographic location. The given dataset doesn't give any information related to geographical location or publication time. Assumption: The publication time of the news article is the time of the impression it first appeared in. Using this assumption we created a new news article dataset which also consists of the publication time. Now at a given instant, we recommend the top news articles which have the highest clicks in the past 48 hrs. This model has no personalization since the recommendation is exactly the same for all the users.

Analysis: For further improvements in the trending recommendation system, we can also consider the user's geographical location and give different importance to a news article based on the exact publication time of the news article i.e. the trending recommendation algorithm must give less importance to an article A compared to an article B which was published after article A, given both the articles A and B have the same number of clicks, since the article B got the same number of clicks as article A in less time period. Hence the trending algorithm must be a function of the publication time of the news article, the number of clicks of the news article, the user's geographical location, and also the user's interests (to provide some user personalization).

D. *Bayesian Personalized Ranking (BPR)*

Dataset creation:

This dataset has 4 columns User Id, Item 1, Item 2, label. The label is assigned value 1 if the user likes Item 1 more than Item 2 (The item is a news article), else label value is 0. We consider the user likes Item 1 compared to Item 2 only if the user views Item 1 and didn't choose Item 2 provided both the news articles are provided with the same impression. Suppose user U_1 was shown articles a_1, a_2 and a_3 and U_1 clicks a_1 only, then we insert rows into the dataset randomly as $(u_1, a_1, a_2, 1)$ or $(u_1, a_2, a_1, 0)$ and $(u_1, a_1, a_3, 1)$ or $(u_1, a_3, a_1, 0)$.

Example: Consider a User whose User Id is 120 with the following impressions:

1^{st} Impression: Clicks = 1 ; Non-clicks = 2,3
2^{nd} Impression: Clicks = 4,5 ; Non-clicks = 6

TABLE I
EXAMPLE DATASET

User Id	Item 1	Item 2	Label
120	1	2	1
120	3	1	0
120	4	6	1
120	6	5	0

The above table is an example of a single user, the actual dataset must include the comparisons for all the users in the system. Both train and test datasets have been generated in a similar way. The train dataset has 51,567 rows corresponding to 250 different random users. The test dataset has 22,487 rows, corresponding to 120 different random users.

- **Approach 1:**
  Given a row of the dataset, i.e., (u, a_1, a_2, l), we do the following:
  1. We maintain a weight vector w.
  2. Let $v = u \odot (a_1 - a_2)$
  3. Let $J = l * ln(\sigma(w \cdot v)) + (1–l) * ln(\sigma(-w \cdot v))$

  Where $\sigma(x) = \frac{1}{(1+e^{-x})}$
  w is learned during training in such a way that J is maximized over all training examples.
- **Approach 2:**
  Given a row of the dataset, i.e., (u, a_1, a_2, l), we do the following:
  1. Transform u linearly using W_u, i.e., $u' = W_u u$
  2. Transform a_1 and a_2 linearly using W_a
  3. $a_1' = W_a a_1$, $a_2' = W_a a_2$
  4. Let $v = u' \cdot (a_1'–a_2')$
  5. Let $J = l * ln(\sigma(v)) + (1–l) * ln(\sigma(-v))$

  W_a and W_u are learnt during training in such a way that J is maximized over all training examples.

Training the Model:

The models corresponding to two different approaches have been trained for 20 and 10 epochs respectively. In

approach 2, the dimension of the transformed user and news embeddings has been set to 10. The threshold with which a model predicts if a given example has label 1 or 0 is different for both the approaches; they are set as 0.457 and 0.418 for approaches 1 and 2, respectively, after doing trial and error in such a way that respective F1 scores are maximized on the train dataset.

Ranking:

- Suppose we have 'k' articles from which we need to provide the impression for the user then we need to compute the k x k matrix where (i, j) cell indicates whether the user likes the news article 'i' compared to the news article 'j', this must be predicted using the previously trained model.
- Now for each article 'i' maintains the vector of count of articles 'j' such that the user likes the news article 'i' compared to the article 'j' according to the model prediction.
- The articles with top count values are given as the final impression to the user.

Results:

TABLE II
ACCURACY METRICS

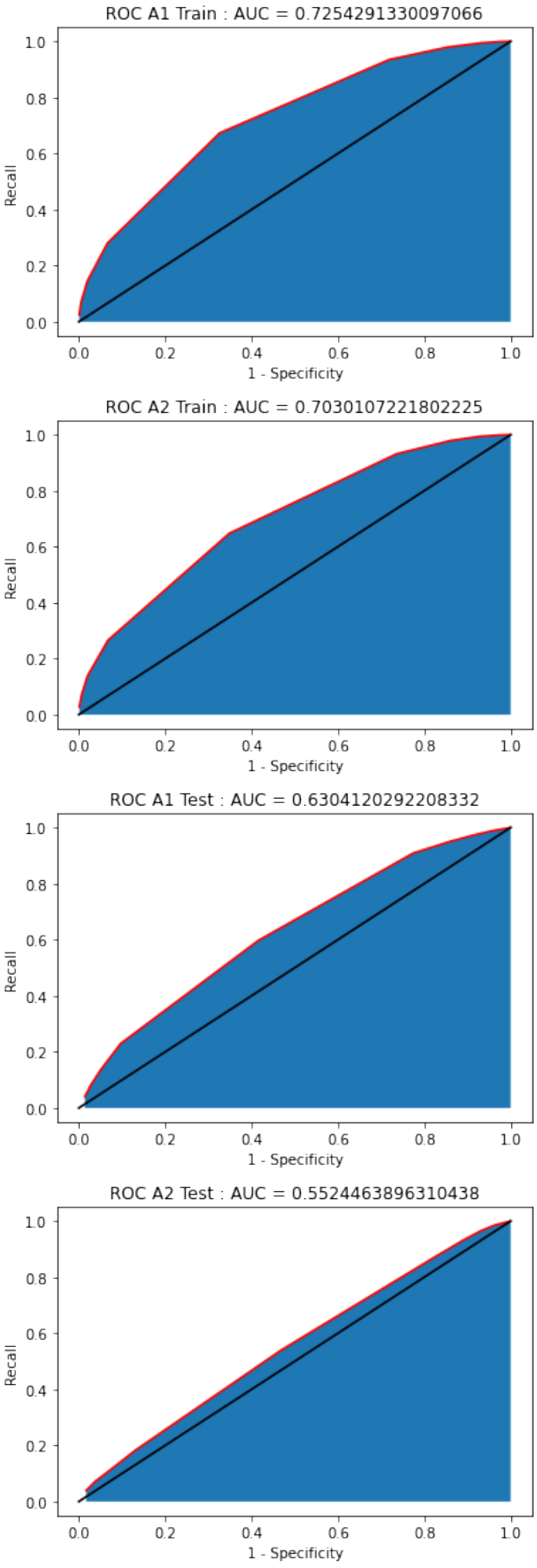
Models	Train		Test	
	Aprch 1	**Aprch 2**	**Aprch 1**	**Aprch 2**
Threshold	0.457	0.418	0.457	0.418
Precision	0.6028	0.5693	0.5561	0.5177
Recall	0.8564	0.9148	0.8200	0.8564
F1 Score	0.7076	0.7018	0.6628	0.6453

As it is evident from the results (test accuracy metrics), approach 1 has better precision and F1 score, whereas approach 2 has better recall. We can use different approaches based on what we want:

- Suppose we want to show very critical or sensitive articles to users, then we need to make sure that number of false positives is less. Then, precision is the accuracy metric which we want to be maximum and hence, we should go for approach 1, since it gives better precision.
- Suppose a business company wants its news to reach all its potential customers, then the number of false negatives must be less. In that case, recall is the accuracy metric which we want to be maximum and hence, we should go for approach 2, since it gives better recall.
- If we want to strike a balance between the number of false positives and false negatives, i.e., get a balanced classifier, we can choose an approach that maximizes the F1 score. Hence, we can go for approach 1 in that case.

By doing a visual inspection several times (code), the ranking of articles from a fixed set of articles, for one particular user is more or less similar.

Following are the ROC curves corresponding to the two classifiers, corresponding to train and test datasets.



Based on AUC - ROC from the above plots, we can say that approach 1 yields an 'overall' better classifier than approach 2.

E. Thompsons Sampling

We divided all news articles into 4 main categories (Life, entertainment, world, and video). There are 4 arms and each arm represents these main categories.

TABLE III
ARMS

Arm	Categories
Life	Lifestyle, Health, Weather, Food and Drink, Travel, Kids
Entertainment	Entertainment, TV, Movies, Sports
World	News, Finance, Middle-east, North America, Autos
Video	Video

- We maintained 4 beta distributions (prior distribution) and sampled a data point from each distribution. Based on this data point, an arm will be picked.
- Based on the user rating for this particular article, the selected arm prior to distribution will be updated.

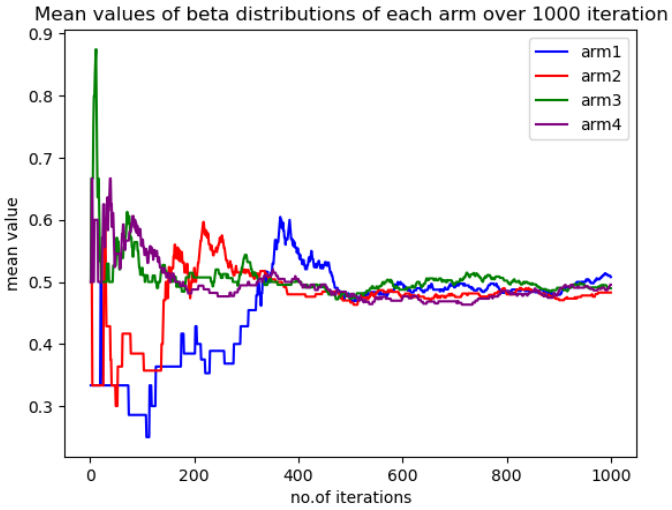
Thompson sampling balances exploration and exploitation by allowing the algorithm to continue exploring new options while exploiting. This algorithm can quickly learn which actions are optimal and maximize the cumulative rewards.

This model is user-specific, so we need to maintain one model for every user. There is no training time for this model. This model gets updated on the go.

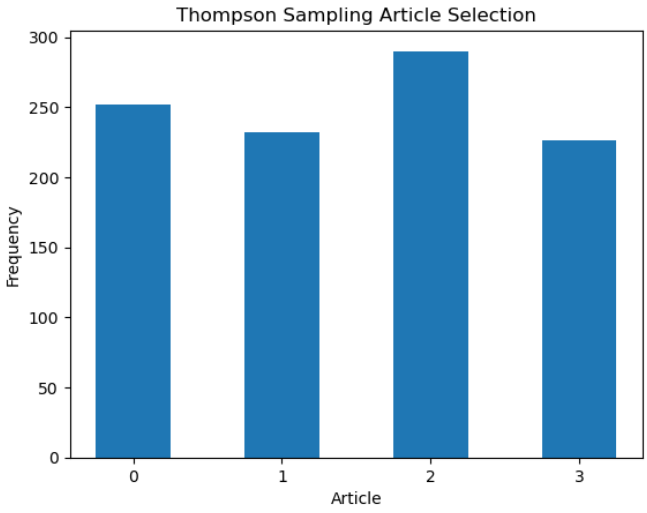
We simulated user interactions with the recommended articles to test the Thompson sampling algorithm without taking input from the real user. We simulated user interactions by randomly generating a click probability for each category article. The click probability is just a number between 0 and 1 that represents the probability that the user will click on the article. We generated a reward that is either 1 or -1, depending on whether the user clicked on the article.

Like probability of a user for each arm is [0.5,0.5,0.5,0.5]

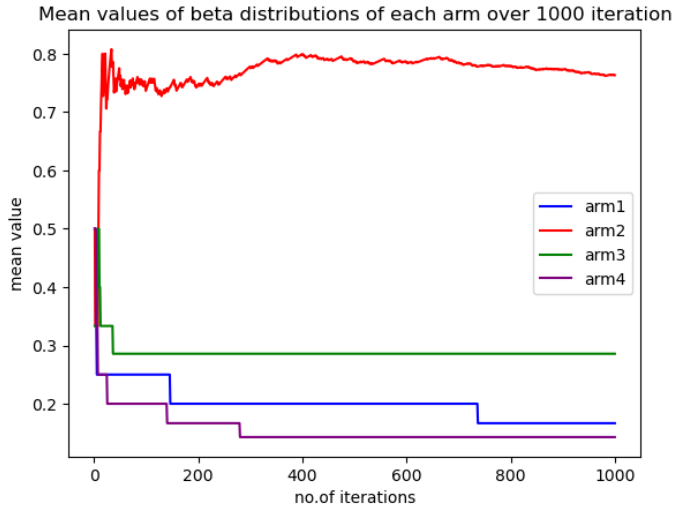
Below graph shows how the mean of each arm's beta distribution changes over the iterations. We can see that after some iterations all mean values converge at 0.5. It shows that our algorithm is working.

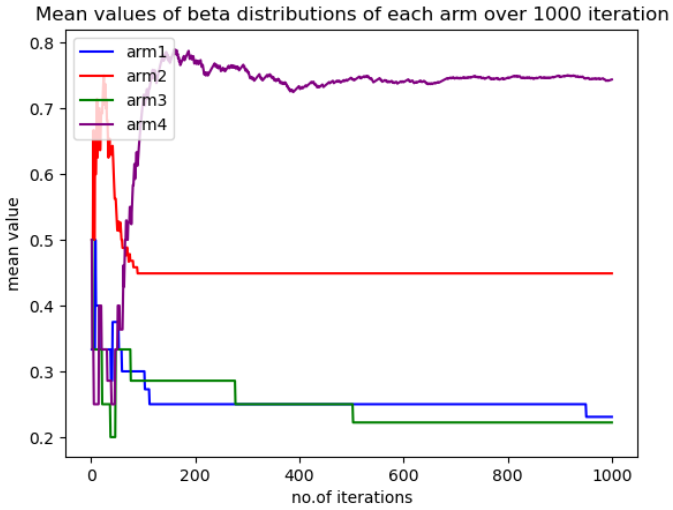


The below graph shows the frequency with which each category article was selected by Thompson sampling algorithm. Since all the categories have equal click probability, all the frequencies are close.



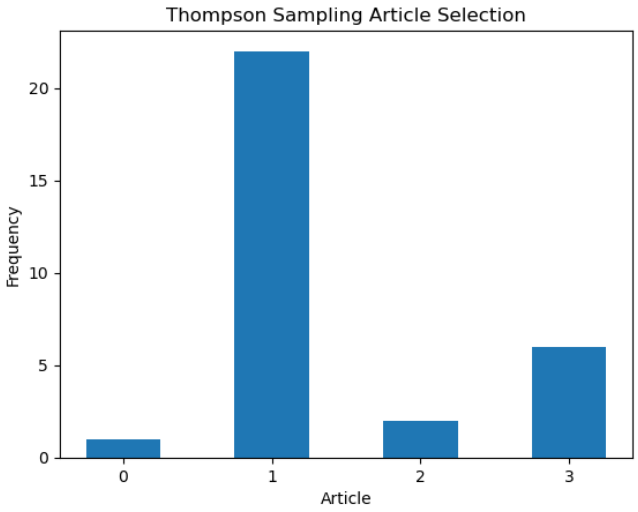
Like probability of user for each arm is [0.2,0.75,0.3,0.1]
As expected in the below graph mean value of beta distributions are converging to the like probability of the user.



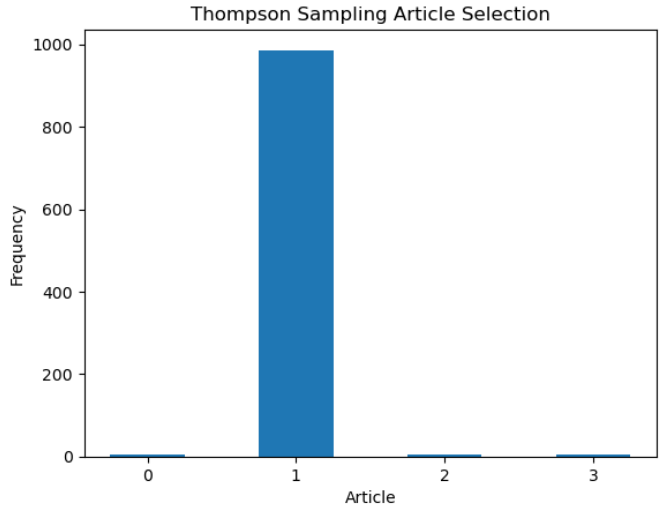


Frequency of news articles of each arm before 30 iterations.

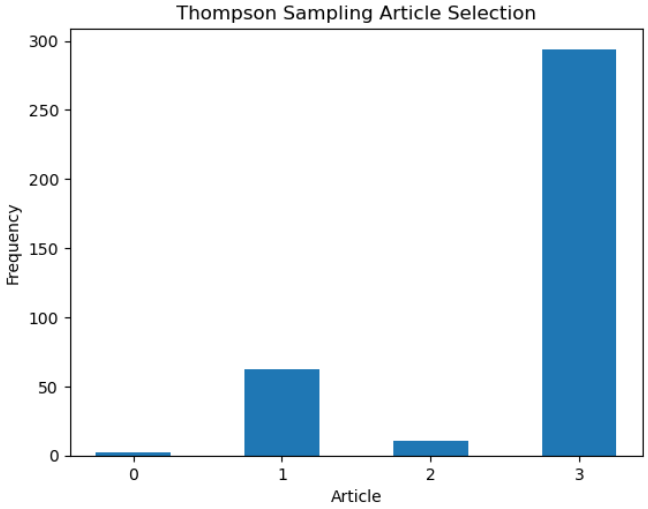
Since the like probability of users towards arm 2 is high, so we expect high frequency for news articles of arm 2. This observation can be seen in the below graph.



Frequency of articles of each arm after 30 to 400 iterations.



Initial like Probability of user is [0.2,0.75,0.3,0.1] after 30 iterations like probability is [0.2,0.4,0.3,0.75]



In the below graph, we can observe that after a couple of iterations, the mean distribution of the second arm is close to 0.75 and after some iterations, it goes close to 0.4. The mean distribution of the fourth arm is updated to 0.75 as expected.

REFERENCES

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