**Frontiers of Computational Journalism – Assignment 2: Filter Design**Name – Shreya Vaidyanathan

UNI – sv2525

* *Designing an Information Filtering Algorithm*

1. ***Users*** – For General consumers
2. ***Filtering what –*** Designing a recommendation filter for the “discover/explore” tab on a primarily image-based social media application like Instagram/Pinterest.
3. ***Available Input to algorithm –*** This hypothetical social media application will allow users to post images (with text captions) and curate feeds of posts within the application. Users can also follow other users and feeds on the app.

*This recommendation filter will mainly be used for the discover tab that has data inputs from the following sources:*

* + Set of all users
  + All User posts
  + User data (preferences, clicks, likes, followers, follows etc.)
  + User Location data
  + All curated feeds on application
  + Staff Top Picks (Human curated/edited feeds)

1. ***Design Factors*** –

This information filter for the explore tab will take a hybrid approach by combining certain aspects of the collaborative-filtering algorithm and the content-based approaches that exist.

Traditional filtering algorithms fail to account for the value of the content when adopting a purely mathematical approach with Collaborative filtering, and the strictly content-based approaches don’t have an efficient way to recommend similar items to users. Both these approaches are essential in the tab as we have data from users on their preferences and data on things which will be explore-worthy from their interactions with others in the application.

The filter here tries to take a more nuanced approach to answer the primary question of ‘Who should see what when?’

The design approach taken here considers the concept of ‘choice architecture’ by trying to align the Key Performance Indicators (KPIs) of the app with the user’s inputs/preferences (allowing more control and flexibility) and the social aspects of content consumption (by redirecting users viewing hateful or undesirable content).

The value of the tab is not dependent on clicks as there is no advertisement opportunity in the tab, so I make a fair assumption that it is rather driven by user-satisfaction. I adopt the factor of giving people more control over what their algorithmic feed serves up in the tab. This also helps overcome the problem of taking click-stream as the only qualitative indicator to push content that has cosine similarity.

Therefore, user preferences are given higher importance by providing more transparency and control in this filter of application (the application itself would have other tabs and pages with algorithms that may have different KPIs).

Users have long term interests in certain topics as well as short term interests in the topics that are relevant globally. The application caters to both long term interests as well as short-term interests based on user input.

Most applications do not allow users to browse freely without impacting your future predictions. This filter understands that user preferences should be accommodated on a spectrum rather than a default fashion. So we provide three modes of interaction on the application through which users can browse through items (images/posts/feeds): (1) “I Like + Want More”, (2) “I Like + But not often” and (3) “I Like + Don’t want more”.

The algorithm will operate in the mode 1 (by default) and the other modes can be switched on the application. Mode (2) will serve as the basis to get inputs on things the user didn’t know that they wanted. The algorithm should sample the space of possible items in a way that reduces uncertainty fastest without drastically altering their feed. Mode (3) is sort of like an ‘incognito mode’ that allows users to explore items in the application which they don’t want the algorithm to learn from and impact their short-term or long-term likes/dislikes. The actions committed in mode (3) will also serve to reduce the noisy data.

The weightage for the mode (2) and (3) will differ and users can also report and give feedback on items that may not fit the specifications that they expect so that we tailor it to their needs.

The algorithm will also provide a relevant score of the final “top stories/top posts” which are displayed in the tab so that it is transparent, and they can make sense of the tab and how their actions impact the algorithm.

It is hard to design in a manner that accommodates all views in a pragmatic and unbiased fashion, so we will also consider the “curated feeds” from editors/staff which will serve the purpose of providing some alternative feeds (not manipulated by their choices) that could lead them to a completely different discoverable set of posts on the application.

Lastly, the filter also takes an approach inspired by the “redirect method” project (https://redirectmethod.org) that senses when users continuously engage with content that is socially unacceptable (i.e. ISIS beheading videos, racist or inflammatory posts).

1. ***Pseudocode for a function that produces “top stories”***

In the case of this hypothetical application, we will consider the objective of the filtering function to be finding the ‘top posts’ to be displayed on a user’s “discover/explore” tab.

One of the major drawbacks of the collaborative filtering is not being able to recommend items that have had no user interaction - known as the cold start problem. This filter uses a random algorithm to solve the cold-start problem and keep the explore tab diverse on the onset when (or until) there is no user preference set.

The matrix factorization approach of calculating the user-item matrix to obtain similarity inputs is computed using the kNN algorithm to get similar users (through the cosine value) and a visual recommendation algorithm to get similar content.

Here is the pseudocode for finding the ‘top\_posts’:

|  |
| --- |
| Input: *(as specified above)*  Output: recommended items (Er)  Method: top\_posts   1. Iu ← items browsed by user (u) #default mode (1) 2. Iu2 and Iu3 ← items browsed by user in modes (2) and (3) respectively 3. Dm ← browser matrix of all users 4. if (Dm = null or Iu = null) then 5. Er = recommended N items by random algorithm 6. + push ‘Top Staff Picks’ to the top when this happens 7. Return Er 8. U = {u1, u2, u3,…..Un} #Defining set of all users 9. Count = 0 10. for each (user in U) do 11. Nb = number of browsed items by user 12. If (Nb >= TR- threshold on recommendations) then 13. Count = Count + 1 14. if (Count <= 0) then 15. Er = recommended N items by random algorithm 16. + push ‘Top Staff Picks’ to the top when this happens 17. Return Tr 18. Nk = N \* (1 – random recommendations) 19. Tk = recommended Nk items by cosine similarity matrix (Taking items in Iu for 100% weightage; Iu2 with 50% weightage and Iu3 with 0%) 20. Trd = recommended Nrd items by random algorithm 21. #Keeping a list of ‘flagged items’ to redirect users when viewing items in list for more than a specified time frame – then induce randomness and top picks into the Er too 22. Er = Tk U Trd 23. Return Er |

* ***Additional References:***
* <http://www.niemanlab.org/2012/07/who-should-see-what-when-three-principles-for-personalized-news/>
* <http://www.niemanlab.org/2012/07/are-we-stuck-in-filter-bubbles-here-are-five-potential-paths-out/>
* <https://www.wired.com/story/creating-ethical-recommendation-engines/>
* <https://tech.wayfair.com/2017/12/recommending-visually-similar-products-using-content-based-features/>
* <https://medium.com/@bagofwords/algorithmic-bias-in-choice-architecture-3917047ed923>
* <https://www.hindawi.com/journals/mpe/2015/145636/alg1/>
* <https://medium.com/@Pinterest_Engineering/evolving-search-recommendations-on-pinterest-136e26e0468a>