

## **High-level Design Logic**

- App clustering
  - NLP topic modeling
    - The app descriptions contain important information about the app. I chose to use Non-negative Matrix Factorization (NMF) over Latent Dirichlet Allocation (LDA) since NMF works with a Tf-idf vectorizer that works well in reducing influence of common words. I preprocessed the descriptions by removing stop words, tokenizing, stemming, and Tf-idf vectorizing. Then, I fitted and transformed the descriptions with NMF. I used NMF on apps with and without descriptions. (use on app descriptions if there is any; otherwise use on app names). I then labeled each topic identified by NMF by looking at top words and apps in that topic.
  - Similarity distance matrix
    - The NLP topic modeling was completed purely on app information dataset. I'd like to leverage the much larger usage dataset and user behavior. Thus, I implemented a simplified collaborative filtering by calculating similarity distance matrix for app-user matrix (with average usage time as values). Then, I extracted the top 10 similar apps for every app. If the app did not have a label, I used the most common label among the top 10 similar apps as the label for the app. Finally, I merged the result with the NLP topic modeling result to complete app clustering.
  - Unsupervised learning
    - I tried to cluster apps using unsupervised learning and considered adding app label on top of app usage time. I tried to use K-prototype clustering, but it takes a while to cluster. I then decided to simply filter the apps based on existing labels and then use a pipeline of normalization, principle component analysis, and KMeans to cluster.
- User clustering
  - Unsupervised learning
    - The same pipeline of normalization, PCA, and KMeans is applicable with user clustering. I just had to transpose the features. However, the cluster results were not ideal and not self-explanatory. Thus, I chose to use data manipulation to group users.
  - Data manipulation

- Leverage pandas dataframe and the .groupby() / pivot\_table() method to extract user groups of interest. Since this process is manual and repetitive, I decided to write a simple plotly dash-based dashboard for easier control and visualization

## - Result interpretation

### ○ Dashboard

- As mentioned before, I wrote a plotly-dash-based dashboard for result visualization. The dashboard looks like the image below. User can select app label and sublabel to select one or more groups of app. Then, user can select the parameters interested (number of apps or daily time spent) and select interested group (passionate, above average, ordinary, or sleeper). The first four tabs correspond to this portion.
  - 'Raw' is raw data
  - 'Agg (overall)' is the overall aggregation regardless of parameter and group
  - 'Agg (by label)' is the overall aggregation with an additional labels field
  - 'Group' is the filtered user group based on parameter and group
- The dashboard is also capable of showing the top 10 similar apps for any given app\_id to help with extracting useful insights.

Please Select Label, Sublabel, Parameter, and Group

Filters

Label:  All

Sublabel:  All

Parameter:  App\_count

Group:  Passionate

app\_id:  133

App Name: Origin

Filtered Results

Raw	Agg (overall)	Agg (by label)	Group	Top 10 Similar Apps (app specific)					
EXPORT									
user_id	app_id	app_name	daily_mins	app_name_full	description	Unnamed: 3	Topic	Label	Sublabel
filter data...									
001e66d76b692614144832e80b51ce5258f62ce676147b3660d5f0cc0b18ee	14	firefox	311.5	firefox browser	Mozilla Firefox is a free and open-source web browser developed by The Mozilla Foundation and its subsidiary	20	7	Software	Opensource/Media
00a52ee6092cc45f9b267539ed757b017b19cc492f07e46ca34d7570e4920e96	13	googlechrome	180.3	Chrome browser	Google Chrome is a freeware web browser developed by Google LLC. It was first released on September 2	20	5	Browser	Browser
008dc62a1008f654f47cb1e076e32aa867dc84334a1101241babd25c46725d4	73	powerpoint	21.7	Microsoft Office PowerPoint	Microsoft PowerPoint is a presentation program	20	3	Productivity	Office/Anti-virus
0086974c24851b09370115a80fba2a8d9978b3d0069e842cca34ae66fab621	73	powerpoint	15.9	Microsoft Office PowerPoint	Microsoft PowerPoint is a presentation program	20	3	Productivity	Office/Anti-virus
00116c89a25f7che589a8e34806c33a514e0f8c339ca68f143396d08415e66c	60	notepad	1.1	Notepad	Notepad is a simple text editor for Microsoft Windows and a basic text-editing program which enables computer users to create documents. It was first released as a mouse-based MS-DOS program in 1983	20	4	Multimedia	Video
00b80d6171215836ecc9675508997579b3018590c089cbe8b8197eb37e55	287	flashhelpservice	1.2	Flash Helper Service		20	6	Productivity	Adobe

## **Code Design**

- Modularity
  - o The code was split into main.py and utils.py. In main.py, the code calls functions and user can specify parameters and execute the analysis. All functions are written in utils.py and can be modified without impacting the main execution much.
- Readability
  - o Comments and docstrings for most functions.
  - o Easy to understand function and variable names.
  - o Fairly easy to understand code flow
- Scalability
  - o Input files are pickle files for faster read for larger input data.

## **How to Use**

- Package used
  - o Package used in this project are pandas, numpy, matplotlib, sklearn, genism, nltk, and dash (for dashboard). The packages are specified in the requirements.txt file
- Main code
  - o Main execution file is main.py. Parameters to specify are file names (app information, user app usage, and category look up table), topk (k most similar apps), NMF n\_components
  - o Functions are in utils.py
- Dashboard
  - o Run group\_dashboard.py. A local address will display (such as <http://127.0.0.1:8050/>). Click the link to get to the dashboards

## **Future Improvements**

- K-prototype
  - o Could potentially use K-prototype or other unsupervised clustering algorithm to further assist app or user clustering
- Deep learning
  - o Could potentially use Keras and implement autoencoder or other clustering algorithm to further assist app or user clustering
- Data cleaning

- The current method did not perform too much data cleaning. I made the assumption that each record in the user\_app\_usage data is unique, even if the daily\_mins and app\_id are the same.
- NLP topic modeling and similarity distance matrix integration
  - The current method simply appends results together. Could potentially try to replace based on common topic percentages or other metrics
  - Current similarity distance matrix method is a simple implementation of collaborative filtering. Could add more complexity to it.
- Try-catch / unit test
  - Could potentially add try-catch-finally block and utilize unit test modules like pytest to make sure the code is robust enough
- OOD
  - Could further utilize OOD concepts such as defining customized classes for more modularity and easier collaboration