# Individual Report

## Recommender System
## *for Shopify App Store*

WANG SHUMIN
A0195334B

# DATA PREPARATION

The success of text analysis and building a high performing recommender system depends on a significant and meaningful set of data on which to operate. To derive business values from text analytics and ratings data for the recommender system, it is important to prepare the data in such a way as to draw out its inherent business context prior to analytics usage and modeling.

## CHALLENGES

### DATA SELECTION

The datasets obtained from the online open source are comprised of several separated files. Although the big amount of data ensures the completeness of our dataset, not all datasets are relevant, and decisions must be made on which datasets to utilize.

### DATA CLEANING

To build the content-based recommender system, the description text data for each APP is used. Cleaning the text data becomes a tedious work because the regular process of removing punctation and stop word, converting all words to lower case and stemming results in some words not understandable. On the other hand, to make the final words meaningful on the contextual basis, it is required to find out more words to remove.

### DATA IMPUTATION

In the ‘reviews’ dataset, the ‘username’ variable is a string and there’s some blank values in this variable. This makes recommendations difficult and inaccurate to some extent. Methods or data imputation must be applied.

## RESOLUTIONS

### DATA SELECTION

Upon some data exploration, two files are selected: apps.csv and reviews.csv among the total six files.

- apps.csv: contains the APP related info such as APP ID, APP name, APP descriptions, which can be used in the content-based model
- reviews.csv: contains the user and rating info of each APP by each user, which can be used in the collaborative filtering model

Other datasets like pricing plans, key benefits and APP categories are either duplicated info or hard to join with the main datasets. Therefore, they are dropped out from the analytics.

### DATA CLEANING

- Certain words must appear together to make the vectorized data meaningful in measuring how important a phrase is in a text. I find in the sklearn package in python there’s a

convenient function TfidfVectorizer with an argument ngram_range to set how many combinations of words need to be considered at the same time. By increasing the number in this argument, the results could be improved, although later the similarity score decreased a bit due to the occurrence of two or more words together might be low. Therefore, it's really up to business context and I need to try and adjust accordingly.

- I draw the word cloud on all APP descriptions to find out the common words appearing on all data and further remove them in the text data. After several rounds of this process, most meaningless words can be dropped efficiently.

### DATA IMPUATATION

User IDs are assigned to each unique user to utilize in the subsequent models so that functions can be created to easily call the User ID for specific user to make recommendations. It does not make sense to assume that all those with missing usernames is the same person and assign the same User ID to them. Therefore, those missing usernames rows are dropped.

## SOLUTION BUILDING

Content-Based, Collaborative Filtering, and Hybrid models are built for the recommender system.

## CHALLENGES

### RATINGS NORMALIZATION

Most users rated the APPs with a higher rate 4 or 5 and some users even rated same for all the APPs they used. This makes normalization difficult to allow ratings among different users become comparable because it would return a lot of NaN given the normalization formula below.

$$Normalized\ Rating = \frac{Rating - \min(Rating\ for\ each\ user)}{\max(Rating\ for\ each\ user) - \min(Rating\ for\ each\ user)}$$

### USER PROFILING

The datasets don't have any info about users, which makes it hard to develop a complete content-based system. Ways to derive the user profiles must be discovered so that recommendations can be made upon interactions of user and item features.

### MODEL EVALUATIONS

- The evaluations for content-based model is difficult because no accuracy metrics can be generated as the fundamental is based on APP descriptions and their similarities.
- Interpretation of results from different approaches in collaborative filtering method can be tricky by looking at different accuracy metrics. Decision is required on what's the main metrics to compare with.

- In the process of building a hybrid recommender system by combining the content-based and collaborative filtering models, a proper weighting must be assigned to both models to ensure the high performance of the hybrid system.

### CODING APPROACH SELECTION

As I used Python to do the coding part while other team members used R, this can be a safety net for ensuring both our codes are correct by comparing our results. However, python and R have different packages with slightly different algorithms to build the models. The recommendations are generally the same but could have discrepancy in terms of order.

## RESOLUTIONS

### RATINGS NORMALIZATION

I've found a function called minmax_scale in python to successfully conduct normalization as well as scaling to avoid the NaN issue. However, the distribution of the normalized rating is left-skewed with a lot of zeros. Therefore, normalization is not applied finally as it's not suitable for this dataset.

### USER PROFILING

In the content-based model, a self-defined algorithm of profile learner is developed as using rating as weightings on the similarities between APPs.

The algorithm is:

$$App\ Recommendation\ Scores\ for\ Each\ User\ =\ (App\_1\ Similarity\ Matrix\ *\ App\_1\ Normalized\ Rating)\ +\ (App\_2\ Similarity\ Matrix\ *\ App\_2\ Normalized\ Rating)\ \ldots\ +\ (App\_nSimilarity\ Matrix\ *\ App\_n\ Normalized\ Rating)$$

$$where\ App(s) are\ those\ that\ the\ user\ have\ rated.$$

### MODEL EVALUATIONS

- Assume if the model is put into production, real conditions evaluation (like A/B testing or sample testing) could be the only way to evaluate content-based model.
- There's always a trade-off between Precision and Recall based on the preference of Type I or Type II given the business context. In our case, it is always less tolerable when the recommender system recommends some APPs to a user who might not like than failing to recommend something that the user might like. Therefore, FP must be eliminated to make the Precision higher and the model with highest Precision should be the optimal one.
- The weighting finally assigned is 50% for each approach because this is safe mechanism at first. When in the real world the hybrid system is put into use, weightings could be adjusted upon feedback from users when they send further reviews and ratings on recommended APPs.

### CODING APPROACH SELECTION

I personally prefer using Python because its packages generally give more flexibility and convenience in terms of data processing and modelling and the printed results usually look prettier to fit into the report. To make the results consistent between Python and R, we need to make sure we use the same datasets upon random modules and train/test split. Further study of the algorithms between the packages/functions of the same purpose is also required.

## CONCLUSIONS

Recommender systems have taken more and more place in our daily lives from e-commerce to online advertisement and they benefit consumers in making a better life and suppliers as well in generating huge amount of income in various industries.

## LESSIONS LEARNED

### CONTENT VERSUS COLLABORATIVE

Through carrying out building the two systems and comparing against each other, I've summarized some advantages and disadvantages of the two approaches as below:

- Collaborative Filtering approach:
- Pros:
  - Requires no info about the user and item
  - Recommendations become more and more accurate and effective with more users' interactions with items overtime
  - Memory based collaborative approach has low bias while model based collaborative approach has low variance
- Cons:
  - Only consider past user/item interactions (cold start problem) and impossible to recommend to new users or recommend a new or any item with too few interactions. The common resolutions of this issue include random strategy, maximum expectation strategy and exploratory strategy
  - Memory based collaborative approach has high variance while model based collaborative approach has high bias
- Content-based approach:
- Pros:
  - Suffer less from cold start problem than collaborative filtering approach since new users or items can be described by their characteristics so relevant recommendations can be done for these new entities.
  - Lowest variance
- Cons:
  - Users profile might change overtime and some item features could be unseen initially, which can deteriorate the effectiveness
  - Highest bias