# DMC2021 Project Report

The Dataworms

presented by
Cheng Chen (1662473)
Chihyen Ou (1725126)
Keyvan Amiri Elyasi (1745168)
Miriam Repp (1722309)
Nazanin Rashedi (1719453)
Sunita Pateer (1715906)
Wafa AbuObid (1619985)
Zhuoyan Chen (1725228)

submitted to the
Data and Web Science Group
Prof. Dr. Heiko Paulheim
University of Mannheim

June 2021

# 1 Introduction

It has become common for enterprises to collect large volumes of transactional data that allows for deeper analysis of how a customer base interacts with the space of product offerings ([Melville and Sindhwani, 2010]). In this real-world case scenario, the owner of a book store is transferring all his business to an online-shop because of the pandemic situation. Both item-specific profile attributes and transactional details are collected and we need to build a reliable book recommendation system which can provide a targeted recommendation to every product page.

# 2 Data Preprocessing

## 2.1 Data Introduction

### Data Understanding

Provided are two data sets, one containing book metadata (*items.csv*) and the other describing transaction details (*trainsaction.csv*). Each of their attributes is listed below (see Table 1)

| **Items** | | | | | | |
|---|---|---|---|---|---|---|
| **Attribute** | **itemID** | **title** | **author** | **publisher** | **main topic** | **subtopics** |
| Unique Values | 78030 | 72128 | 35970 | 7073 | - | - |
| **Transactions** | | | | | | |
| **Attribute** | **itemID** | **sessionID** | **click** | **basket** | **order** | **-** |
| Unique Values | 24,909 | 271983 | - | - | - | - |

Table 1: Attributes of Original Data

We have in total 365,143 sessions in our transactions data set, among which 271983 IDs are unique. 65% of sessions only have one record. In contrast with the other 35%, we have 129,501 records. No user information are provided and the distinctness of different sessions could not be recognised, which implies that our recommender cannot work as usual systems who mainly focus on user-personalization, but make recommendations based on items' information. Although transactions history only covers 32% of books, we could still use it to find item-wise associations within like which of them have been bought or at least clicked and viewed together. When it comes to book data, for 5.8% of all records there is at least one other record with the same title, author and publisher but different ID. 3.7% of records are duplicated with regards to same title and author, but come from different publisher. Moreover, 3.5% of records have same title but different author.

Deduction can be that it's just a coincidence but it's also likely to be the typos of initial data extractions.

#### External Datasets

We used two external datasets[1] for feature generation: 1) Google books dataset[2] resulted from querying title, and author 2) Thalia dataset [3]. Further, we come up with various tweaks for resolving discrepancies between multiple sources, and generating quality features (see: section 2.3).

### 2.2 Data Cleaning

As it is discussed in the data preprocessing section, item dataset includes considerable amount of duplicates. Apart from the issues mentioned in 1, There are 3,390 items, 0.04% of the whole item dataset which are duplicated in terms of having same title, author, publisher, main topic and subtopics but different itemIDs. Since the final goal is to have recommendations per itemID, it is obviously not desired to get the exact copy of the query item as a recommended new book. Therefore, we cleaned these by prefering the itemID in validation and otherwise the minimum of itemIDs representing that single book. We performed this cleaning on transaction dataset too, so that all transaction records will still be considered but with the cleaned itemIDs. We are aware that this may result in lower precision and recall for our recommender but we keep this approach since we believe it to be beneficial in the online evaluation method. After adding language feature to our dataset, we find that 141 of duplicates refer to same books published in different languages. Other duplicates might be caused by different publish dates. Since we put a duplicate title removal in our recommendation pipeline, we do not remove duplicates. Nonetheless, we remove all duplicated items and aggregate all of their transactions data for any further mining activities related to the transaction dataset (e.g. creation of transaction network). Furthermore, using duplicated items, and external datasets 1034 NaN values of author feature are replaced properly.

### 2.3 Feature Generation and Processing

#### Language Detection

It seems to be rational that users who are searching a book on our website, to be more interested in books that are written in the same language. Further exploration

[1] Credit of datasets belongs to other teams that generously shared their results with other groups.
[2] `https://www.googleapis.com/books/`
[3] `https://www.thalia.de/`

of transaction dataset w.r.t items that are clicked, put in basket or ordered in the same sessions, supports this hypothesis. Therefore, we decided to use a language filter right at the beginning of our pipeline. However, this decision requires a highly reliable language feature generation. To do so, we used external datasets (Google books, and Thalia datasets) as well as langdetect library [4]. Nevertheless, we still should deal with the limited performance of the library as well as the problem of multiple objects for each item ID found in external datasets. Since lengthier inputs will increase performance of language detection, both title and description are used for language detection. Further, and even more importantly though, we decided to improve the quality of language feature in multiple iterations. In each iteration, we compared language features derived from different sources, and resolved discrepancies.

### Interest Age

In recommendation system, books are often recommended with the other books with similar interest age. We used topic category [5], starting with 5A and Y, to generate age feature. The 5A category, interest age, contains 17 age categories which we grouped them into four groups, infant (0-6 years old), children (7-12 years old), teenager (13-17 years old), and adult. Since we need a numerical value to calculate the age similarity between items. We assigned a value of 0, 1, 2, and 3 for infant, children, teenager, and adult, respectively. If an item is without 5A, then the interest age of the item is based on topic starting with Y. Y is a category for children and teenager which does not have an obviously age boundary for each subcategory. The interest age of the item starting with Y is assigned a value of 1.5, between the value of children and teenager. Then we used the other dataset, Googlebooks, to fill the missing value of interest age. Googlebooks also contains age information, which a value of 1 represents books for adult, and a value of 0 is not for adult. We combine this age information to our data, which has missing value in interest age feature, by assigning a value of 3 for adult and 1 for non-adult. Since most of the items contain more-than-one categories starting with 5A or Y, then we average all the values in interest age feature. After matched by all age information, the item with missing value in interest age is assigned a value of 20.

### Item Description

To prepare the book descriptions for our model we used a standard preprocessing procedure. We first removed all special characters and punctuation. Subsequently

[4] `https://pypi.org/project/langdetect/`
[5] `https://ns.editeur.org/thema/en`

all uppercase letters were replaced by their lowercase counterparts. Furthermore, we removed all stop words according to the language of the description. Note that this was limited to the five most common languages which were English, German, Spanish, Italian and French. Following this, we used lemmatization to map inflected word forms to their base form. After lemmatizaion we were still left with some plural forms so we used stemming as a final step to remove them.

**Publisher Clusters**

The concatenated preprocessed descriptions for all books that are written by same publisher is used to cluster publishers. This feature generation activity is only applied for the five most common languages in our dataset (publisher clustering is done for each language separately). Since the number of books written in other languages is very limited, clusters are much helpful for recommending books written in that languages. The task at hand was very similar to document clustering problem. We used Singular Value Decomposition (SVD) for dimensionality reduction, then applied k-means clustering method with k=5 for all languages. To come with a suitable valued for k, we run k-means with different values of k, and choose k where Sum of Squared Error (SSE) improvement decreases (knee value of k). We also consider the cosine similarity of centroids, and the number of books in each clusters in our decision making w.r.t optimal value for k. Finally, we check the predictive power of our clustering approach using the Amazon validation dataset.

**Author Clusters**

Similar to publisher clustering, k-means method is also used for authors. We create documents consisting of preprocessed concatenation of books' category heading that are written by same author (i.e. the equivalent categories to cryptic topics, and sub-topics[6]). Using SSE improvement, cosine similarity of centroids, and number of books in each cluster as decision makeing criteria, k=20 turned out to be the optimal value for number of clusters. The predictive power of author cluster feature is also successfully assessed using the Amazon validation dataset.

**Number of Pages**

Using external datasets, we add number of pages as a new feature to our item dataset. Despite the fact that we find out that this feature can play an important role in book recommendation, there were still three main challenges: multiple number of pages found for each item ID, outliers (i.e. books with less than 24, or more than

[6] `https://ns.editeur.org/thema/en`

640 pages), and finally the limited availability of number of pages for all books in items dataset. Since multiple entries are likely to refer to different editions, we simply use the mean of all found page numbers. Further, we mapped all values to [24,640] interval to to avoid deteriorating role of outliers. Finally, we used two separated content-based recommender systems as the number of pages feature includes 11,667 NaN values.

# 3 Methodology

## 3.1 Content-based Model

### Transaction Network

In the absence of user rating data on book items, we use the transaction data to simulate the user behavior. This idea comes from a movie recommendation system which recommends movies without user preference data [Debnath et al., 2008].

There are 365143 transaction data which creates a transaction network with 14279 nodes and 303666 weighted edges. Each transaction has an action such as click, basket, or order. In our transaction network, all actions are considered as the same. Each node represents a unique item. If two unique items appear simultaneously at the same session ID, regardless of their action, these two nodes will be linked by a weighted edge with a weight of 1. The weight of each edge is the frequency of two items in the same session.

### Feature Weight

Each item is represented by a sequence of features which are author, main topic, interest age, title, publisher cluster, number of pages, and author cluster. Based on the assumption that the weight of each edge represents the similarity of each node pair. The similarity S between node $n_i$ and node $n_j$ is the sum of the difference f($A_{ni}$, $A_{nj}$) in attribute An, given $w_n$ as feature weight, between each node pair. Those weights are calculated as follow:

$$S(n_i, n_j) = w_1 f(A_{1i}, A_{1j}) + w_2 f(A_{2i}, A_{2j}) + ... + w_n f(A_{ni}, A_{nj}) \quad (1)$$

| Feature | Type | Distance |
|---|---|---|
| Author ($x_1$) | String | $A_1$=$A_2$? 1 : 0 |
| Main Topic ($x_2$) | String | Lin_similarity($M_1$,$M_2$) |
| Interest Age ($x_3$) | Float | $\|\|A_1$-$A_2\|\|/20$ |
| Title ($x_4$) | String | BM25($T_1$, $T_2$) |
| Publisher Cluster ($x_5$) | Float | Cosine($P_1$,$P_2$) |
| Author Cluster ($x_6$) | Float | Cosine($A_1$,$A_2$) |
| Number of Pages ($x_7$) | Integer | 1 - ($\|\|N_1$-$N_2\|\|/(640 - 24)$) |

Table 2: Feature Weight

Table 2 shows the calculation of the difference of each feature f($A_{ni}$, $A_{nj}$) between node i and node j. If $n_i$ and $n_j$ is written by the same author, f($A_{ni}$, $A_{nj}$) will be 1. Otherwise, a binary value 0 will be given. The difference of main topic is computed by lin similarity which proposed in 2018 [Shahzad et al., 2018]. Since the range of interest age is from 0 to 20, the age difference is normalized to the range of 0 and 1. BM25[7] is used for the similarity of title. Both publisher cluster and author cluster have similar calculation process. First, language information for each node is necessary. If $n_i$ and $n_j$ are in the same language, f($A_{ni}$, $A_{nj}$) will be assigned as the value of the cosine similarity matrix (cosine similarity of centroids) at the position i and position j. Otherwise, the similarity will be 0. The last feature is the number of pages. Based on the distribution of the number of pages, the number of pages under 24, and over 640 are considered as outliers. These outliers are removed and reassigned either 24 or 640. If the number of pages of $n_i$ and $n_j$ are not zero (NaN), f($A_{ni}$, $A_{nj}$) will be the page difference between both nodes. If there is any node without any pages, it will be assigned a random number. This way, we can ensure that NaN values will not destructively influence feature weighting.

Given each attribute $A_n$ is an independent feature and the weight of the edge is the predicted variable in the regression model. Before training the feature weight, some outliers are removed. For example, network data which has title score larger than 50, and weight score larger than 200 are removed. To have a clear understanding on age difference, the interest age is minus by 1, denoted age closeness in the following. Then the score of title is normalized to the range 0 and 1. After data pre-processing, all feature weights $w_n$ are trained by Random Forest which has a good fit, a R-square of 0.76. The optimal feature weights are showed in Fig. 1.

[7] https://pypi.org/project/rank-bm25/

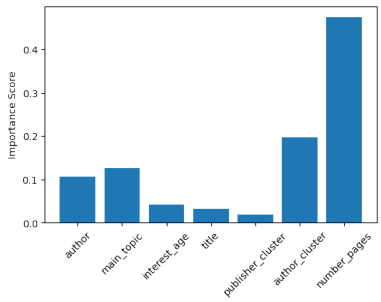


Figure 1: Feature Weighting

**Content-based Model with Feature Weight**

Our content-based model is a sequence of pipeline. First, we keep the candidate items written by same language as the query item. Second, the candidates with same author as query item will be assigned a value 1; otherwise, will be assigned 0. Third, the similarity of title and description between candidate and query item are calculated by BM25. From the distribution of transaction network, over 75 % of the data has a value of age closeness and publisher cluster over 0.95 and 0.76, respectively. Those candidates with lower value in these two features are removed. Based on the description similarity, computed by BM25, the final candidates have the top 1000 largest description score. The similarity of author cluster is from author cosine similarity matrix. The similarity of number of pages and main topic are the absolute page difference, and computed by Lin similarity, respectively. Last, each feature weight which is trained by Random Forest is assigned to each feature in content-based model. Then the items in top 5 largest similarity score are our recommendations.

## 3.2 Hybrid Model

In this step we tried to predict for each item a list of items that will potentially co-occur with it in one session. For this purpose we used *transactions.csv* dataset.Since this is an implicit approach towards the interactions, we needed to take care of the unique characteristics of such implicit feedbacks in the domain of a book store. For instance, unlike the user interactions in the environment of Music Recommender System that the number of plays indicates the level of preference, in a purchase transactional dataset we should ignore the number of times each interaction occurs but instead take care of the weights for different interaction types. More precisely, the data point indicating 118 clicks and the one with only 1 click should be considered both with click-level attraction of item for user. Therefore, we computed

'attraction' feature by adding 1 attraction score for having clicks and 5 for having either basket or order. This way any outlier detection and removal of some data points for having much higher number of clicks for example, becomes irrelevant.
In the next step we created the item-item co-occurrence matrix and later used Collaborative Filtering to predict this attraction scores for missing pairs considering the unique characteristics of implicit feedback, which prevent the direct use of algorithms that were designed with explicit feedback in mind[Hu et al., 2008].

**ALS Collaborative Filtering**

We iused the implementation of Alternate Least Squares in implicit package. In this procedure we assumed sessions acting as users baring in mind the difference of those. Since we are using the model to predict the pairwise transactional similarity of items and not getting recommendations for users, we assume this simple assumption would not hurt the results.
This model distinguishes between Preference and Confidence levels to keep track of never seen. If we assume $r\_ui$ equal to the attraction of item i for user u, then the preference level $p\_ui$ would be 0 for $r\_ui = 0$, and 1 for $r\_ui > 0$. Then the confidence level will be considered as $c_{ui} = 1 + \alpha r_{ui}$ where $\alpha$ is the hyperparameter to the model. Since we are already putting weights on our interactions in the computation of attraction feature, we set $\alpha = 1$. Fitting the model to our dataset, we got a matrix of pairwise transactional similarities for all items participating in the *transactions.csv*. We later used this as another similarity feature, namely *transactional_similarity*, to our model in combination with the average similarity derived from the Content-Based model 3.1 and after evaluations on a separate validation set, we decided to give this new feature a weight of 0.2. Although this hybridization made only a tiny improvement in the current evaluation metrics, it clearly brings an approach of putting higher priority to top sellers and even in the cases of similar recommendations with the previous version, the rank of recommended books differ resulting the top sellers appearing on top of the list.

# 4 Results and Evaluation

To select the recommender system with the best performance, we used Amazon recommendations for validation and introduced various evaluation metrics like recall, precision, and f1 score to compare different models. Given that our recommender systems derive five recommendations for each query, the evaluation metrics are modified and are calculated as follows:

$$recall@5 = \frac{\#\ of\ true\ labels\ captured}{\#\ of\ true\ labels} \tag{2}$$

$$precision@5 = \frac{\#\ of\ true\ labels\ captured}{\#\ of\ predictions\ made} = \frac{\#\ of\ true\ labels\ captured}{5} \tag{3}$$

$$f1@5 = \frac{2}{\frac{1}{recall@5} + \frac{1}{precision@5}} \tag{4}$$

Since our candidate pool contains only a small portion of items in the Amazon database, the size of the amazon recommendation set varies greatly. Therefore, we mainly rely on precesion@5 for model evaluation. The first version of our content-based model has achieved a precision value of 0.1595. After higher weights are assigned to the attributes "interest age" and "publisher clustering", the precision score has increased to 0.1672. The hybrid model that which has incorporated the transactions similarity has demonstrated the best performance (precision@5: 0.1707). We also have experimented with different values of the weight of the transaction similarity in hybrid models. Once the weight has increased from 0.2 to 0.5, we did not observe any variation in the model predictions. This result indicates that although transaction co-occurrence can help yield better recommendations, the content information of items still plays a dominant role in making predictions.

| **Models** | **Average Recall@5** | **Average Precision@5** | **Average f1@5** |
|---|---|---|---|
| **Content-based Model 1** | 0.2344 | 0.1595 | 0.1730 |
| **Content-based Model 2** (after weight adjustment) | 0.2495 | 0.1672 | 0.1824 |
| **Hybrid Model 1** (content similarity:transactions similarity = 0.8:0.2) | 0.2545 | 0.1707 | 0.1863 |
| **Hybrid Model 2** (content similarity:transactions similarity = 0.5:0.5) | 0.2545 | 0.1707 | 0.1863 |

Table 3: Model Results

We noticed that approximately 36% records in the validation dataset only have at most 2 Amazon recommendations, and realized such records can possibly bias the value of evaluation metrics downwards. Thus, we additionally created several subsets of the validation items by implementing different restrictions on the number of an item's Amazon recommendations and then calculated the evaluation metrics for all models on each subset, respectively. Results are reported in Table 2. In all cases, the hybrid models have better performance than the content-based model, showing that the incremental prediction accuracy of hybrid models upon the content-based models is pretty robust. Besides, if we just look at a single model, either a pure content-based one or a hybrid one, we can see that the prediction score monotonically increases as the minimum number of Amazon recommendations increases. We interpret this trend as a positive signal of the prediction power of our models: if given a same candidate pool for recommendation as Amazon, our recommender system is very likely to output a list of recommended books that can better align with the Amazon recommendations.

| Minimum number of Amazon Recommendation | Content-based Model | | | Hybrid Model 1 | | | Hybrid Model 2 | | |
|---|---|---|---|---|---|---|---|---|---|
| | **Average Recall** | **Average Precision** | **Average f1** | **Average Recall** | **Average Precision** | **Average f1** | **Average Recall** | **Average Precision** | **Average f1** |
| **1** | 0.2505 | 0.1680 | 0.1832 | 0.2556 | 0.1714 | 0.1871 | 0.2556 | 0.1714 | 0.1871 |
| **2** | 0.2318 | 0.1935 | 0.2017 | 0.2381 | 0.1978 | 0.2066 | 0.2381 | 0.1978 | 0.2066 |
| **3** | 0.2202 | 0.2150 | 0.2130 | 0.2248 | 0.2190 | 0.2172 | 0.2248 | 0.2190 | 0.2172 |
| **4** | 0.2236 | 0.2429 | 0.2305 | 0.2266 | 0.2464 | 0.2338 | 0.2266 | 0.2464 | 0.2338 |
| **5** | 0.2382 | 0.2790 | 0.2556 | 0.2423 | 0.2840 | 0.2601 | 0.2423 | 0.2840 | 0.2601 |

Table 4: Model Results

# 5 Conclusion

In this project, we develop two recommender systems: the content-based model, and the hybrid model. The content-based model is a chain of filtering steps followed by a final step for calculating the similarity of all remaining items. In the filtering step we follow the basic strategy of keeping only books that are written in the same language. Further, we used description, age interest, and publisher cluster for a more fine-grained filtering. After filtering, the content-based model calculates the similarity of all remaining items using most important features, namely author, main topic, interest age, title, publisher cluster, number of pages, and author cluster. To find the optimal weights for these features, we used the transaction data to simulate user behavior. Each node in the transaction network represents an item, and edges between nodes show the co-occurrence of items. The basic assumption here is that the weight of each edge can effectively represent the similarity of pair of nodes connected by that edge. Finally, we use Random Forest to find weights of features in a way that the error of predicting weights of edges in transaction network would be minimized. Since weights of all features in our content-based model derived from the transaction network, we have already incorporated the user-base history profile. Nevertheless, our hybrid model explicitly captures the co-occurrence of items in transaction dataset. This model deploys the Alternate Least Squares (ALS) method to generate the transactional similarity feature. The hybrid model is developed on top of the content-based model and is able to bring an approach of putting higher priority to top sellers.

To evaluate performance of our recommender system, the Amazon validation dataset, and various performance metrics are used. Number of entries in the validation dataset were limited, nonetheless our evaluation approach is robust enough to distinguish models that outperform their counterparts significantly. The final evaluation in the Data Mining Cup 2021 will be based on the user votes. Yet, we believe that the promising result obtained by our model in the similar setting can be considered as an indication of decent performance of our model.

# Bibliography

[Debnath et al., 2008] Debnath, S., Ganguly, N., and Mitra, P. (2008). Feature weighting in content based recommendation system using social network analysis. *Proceeding of the 17th International Conference on World Wide Web 2008, WWW'08*, (December 2015):1041–1042.

[Hu et al., 2008] Hu, Y., Koren, Y., and Volinsky, C. (2008). Collaborative Filtering for Implicit Feedback Datasets. In *2008 Eighth IEEE International Conference on Data Mining*, pages 263–272, Pisa, Italy. IEEE.

[Melville and Sindhwani, 2010] Melville, P. and Sindhwani, V. (2010). *Recommender Systems*, pages 829–838. Springer US, Boston, MA.

[Shahzad et al., 2018] Shahzad, K., Pervaz, I., and Nawab, R. M. A. (2018). WordNet-based semantic similarity measures for process model matching. *CEUR Workshop Proceedings*, 2218:33–44.

## 1 Supplemental Material

You may find our project code and used data mentioned in the paper on GitHub:
`https://github.com/chihchih-ouoh/DataMiningCup2021-TheDataworms`