FACULTY OF INFORMATION TECHNOLOGY
OF TECHNICAL UNIVERSITY OF MUNICH

Seminar paper

Master Seminar Data Mining

Author:	Maurice Schweitzer
	Maximilian Lange
	Quazi Fahim Faisal Dhruba
	Leon Schulz
	Tawkir Ahmed Fakir
	Haowen Guan
	Natalie Dach
Supervisor:	Prof. Martin Bichler
Assistant:	Stefan Heidekrüger, M.Sc.
	Nils Kohring, M.Sc.
Submission date:	23.08.2021

Table of Contents

List of Figures

List of Tables

List of Abbreviations

API	Application Programming Interface
ISBN	International Standard Book Number
MCA	Multiple Correspondence Analysis
URL	Uniform Resource Locator
VAE	Variational Autoencoder

1 Introduction

As part of this seminar the team took part in the Prudys' Data Mining Cup 2021, a yearly data mining competition among student teams from all over the world. The organisers provide the task and data and each student team works independently on solving the task.

This year the challenge was to build a recommender system for a book store. The competing teams were to find the best five recommendations for 1,000 randomly chosen books. Besides the task itself the organizers provided the teams with a first set of data. One dataset contained item specific information like title, author and topics for several thousand books and another dataset contained transaction specific information on whenever a book was clicked, ordered or put into a basket.

A special challenge in this year's task was posed by the evaluation criterion: After the end of the challenge the organisers published the recommendations on a website and it was up to the website's visitors to decide on the best recommendations in a voting mechanism. For each item, the website visitor was presented with three of the submitted recommendations from three different teams. The best voted recommendations were rewarded with points. The final ratio was normalized based on each team's individual frequency of recommendations (prudsys, 2021).

We tackled this challenge using a two level scoring model based on data provided by the organizers and external data we collected from third party sources. The first level of our recommender system consists of a scoring model combining four underlying models that leverage item specific and transactional data. The outcome of those four models are up to five recommendations per model that are combined into an ordered list of up to 20 recommendations for any given book out of the test set. The second stage model optimizes our recommendations on the evaluation method and the way the books are presented on the evaluation website by using additional features like cover images to boost those recommendations from the ordered list that e.g. have a similar cover image. Out of that reordered list we take the top five recommendations and return them as the final output of our recommender system. See Figure 1 for an overview of our approach.

In the end we secured an excellent 10^{th} position out of 115 competing student teams from all over the world.

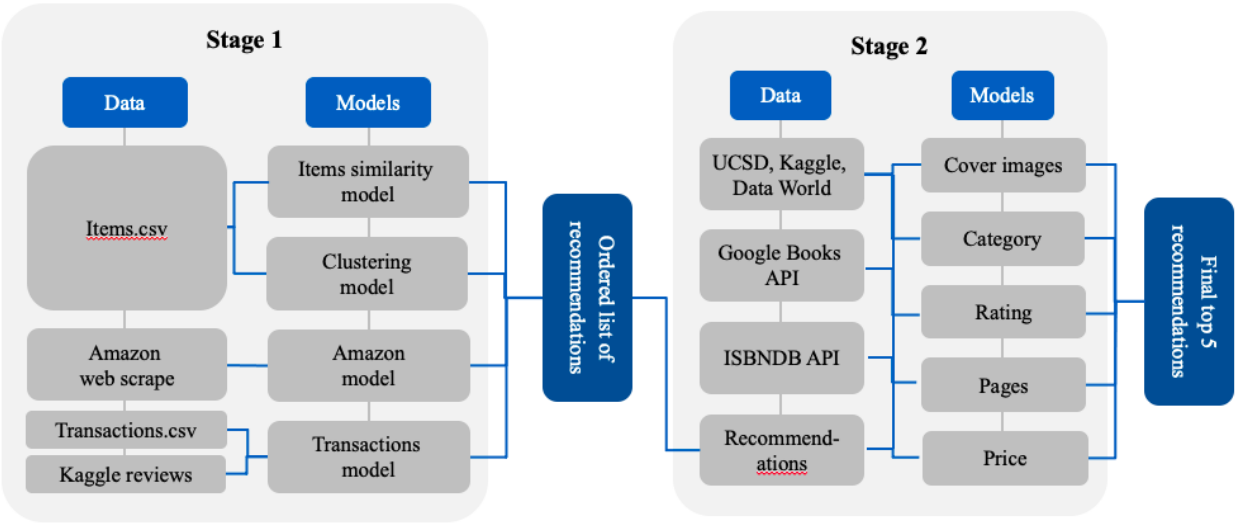

Figure 1 – Overview of two model approach

2 Data

In this section we briefly explain the primary datasets used to build the recommender system. We will also present initial analysis and observation on the provided data and introduce various external data sources that we used to supplement our models.

2.1 Data Analysis

There were two datasets provided for the challenge. The item dataset contains descriptive features of the books (78,334 in total) that are available for the system to recommend. An overview of the features is presented in Table 1. We observed that some items had the same *Title* with only 72,404 unique values from a total of 78,334. Further we identified 28 different languages in the *Titles* with about 93% in English or German. Missing values were an issue for most features including the *Sub topics* (47%), *Author* (4%) and *Publisher* (0,01%). Where possible we imputed the missing values through the collected external data described in the chapter 2.2.

The second dataset is the transaction dataset which contains information about clicks, baskets and orders over a period of three months. An overview of the features is presented in Table 2. Each entry in the dataset represents one transaction for one single item. In total 365,143 transactions are available to use and no missing values in the features. A core observation via the *ItemID* was that only 32% of the books present in the item dataset were available in the transaction dataset. Further

we realized that if a customer buys multiple books in a single session, then the *SessionID* will be the same for all those entries. Overall, we identified 271,983 unique *SessionIDs* in total.

Table 1 - Items data

Feature	Description	Example
ItemID	Unique identifier for every books	8960
Title	Name of the book	Harry Potter 2 und die Kammer des Schreckens
Author	Name of the author	Joanne K. Rowling
Publisher	Name of the publishing company	Carlsen Verlag GmbH
Main topic	Single alphanumeric code for primary book genre	YFH
Sub topics	One or more alphanumeric codes for sub-genres	5AL,5LD,1DDU,6FG,FM,YFB,YFC,YFH

Table 2 - Transactions data

Feature	Description	Example
ItemID	Unique identifier for the book in the transaction	21310
SessionID	Unique identifier for a transaction session	0
Click	Number of clicks for a book in a session	1
Basket	Quantity of the book that is added into the basket during a session	0
Order	Number of orders for a book in a session	0

2.2 Data Collection

Because the provided data sets were limited, we conducted extensive research for additional data. In a first step, we found four existing datasets. The first dataset was scraped from Goodreads.com, a book cataloguing website (UCSD, 2019) and contained 2.36 million books. Additionally, two datasets could be found on Kaggle (Kaggle, 2021b). The last dataset was sourced from Data World and contained another 270,000 books (Data World, 2017). All datasets were pre-processed and transformed. This for example included the merging of files, e.g. with ratings or author information

that was stored in separate files. Additionally, numerical data such as the ratings had to be adapted to the same scale in order to be comparable. Afterwards, the datasets were concatenated while obvious duplicates were eliminated. Eventually, the external data consisted of roughly 2.5 million rows. Besides the standard information regarding title, author and publisher the dataset for example contained the ISBN, country, language, average rating, number of pages or even links to cover images of a book, if available. Next, this data had to be combined with the items in the items.csv dataset via fuzzy string matching. "Fuzzy string matching […] is the process of finding strings that are similar, but not necessarily exactly alike." (Morton, Ingersoll, & Farris, 2012). For all 78,000 items in the items.csv dataset, the fuzzy string matching method was applied to search for the best match in the external dataset. Because existing libraries like Fuzzywuzzy (Seatgeek, 2020) were too computationally expensive, we applied an alternative algorithm that was able to drastically decrease the processing time (Taylor, 2019). Besides existing datasets we found two promising APIs: Google Books API (Google, 2021a) and ISBN Data Base (ISBNdb, 2021). By querying the ISBNDB API for each item in the items dataset using the title, we found a match for 98% of the books. Afterwards we again used the fuzzy string matching algorithm to calculate the confidence for the title, author and publisher. After adapting the API script to the Google API we applied the same methodology as for the ISBNDB API. Both API datasets included information such as the ISBN, average rating, links to cover images, prices and languages.

Eventually, we used all three datasets from the external sources to find the best match for each item in the items.csv dataset. We combined all datasets and calculated the confidence for each match for the title, the author and the publisher using Fuzzywuzzy. For authors and publishers we used the function "partial_ratio". The function looks for partial matches between the two strings, which for example covered cases where the items.csv dataset included only one author, but the external dataset had a list of authors. In a next step the dataset was grouped by the ISBN. This prevented loss of information and e.g. allowed to store links to cover images from different sources as it can be seen in the example in Appendix A. Afterwards the dataset was prefiltered. Thereby, each entry had to match at least one of the following rules: (1) The confidence of the title was greater or equal to 95% and if the author was known the confidence had to be greater than or equal to 30%. (2) The confidence of both the title and the author was greater than or equal to 45%. These thresholds were concluded after analysing several sample cases. After sorting the data we only kept the first respectively best match for each item. The final dataset contained one match each for roughly 87%

of the items (68,678 items). 59% of the matches came from Google, 29% from ISBNDB and 11% from the fuzzy matching. Additionally, 8.9% of the items in the evaluation set did not find a match. Lastly, we were also able to fill missing authors information in the items.csv dataset. Relevant features of the final external merged data is depicted in Appendix A respectively Table 5. Besides the data described here we scraped Amazon.com recommendations for items in the test set, which is described in chapter 3.4.

3 Models

Our approach to the tasks consists of two distinct stages. In the first stage, we created four different models, which produced a ranked list of up to 20 recommendations for each book. These models include the

Transaction Model (based on the transaction data), Item Similarity Model and Clustering Model (based on the items data) and the Amazon Model (based on external data). In the second stage, the recommendations generated by the first stage models were merged and re-ordered by utilizing external data, such as covers, ratings and price. Finally, the top five recommendations for each book were extracted which formed our final submission. In addition to the following explanations, we have published our code in full (haowggit, 2021).

Given the subjective nature of the evaluation criterion, we decided to iteratively evaluate the performance of our models by manually examining the recommendations. Therefore, we divided the recommendations among ourselves and examined them for discrepancies, noting the presumed reason. After aggregating our notes, we decided in which respects our models needed improvement.

3.1 Transaction Model

The only source of user data is the transaction dataset provided by the organisers. Nevertheless this data consists of non-personalized sessions and there is no way to connect those session to their underlying user accounts. The transaction dataset contains data that can be used to discover patterns of books that are more likely to be bought together than others and thus would also make for good recommendations. Due to the importance of unbiased transaction data, we extended the existing transaction dataset with ratings data, that can be found online (Kaggle, 2021a) and merged them to the existing dataset via ISBN numbers.

The underlying assumption of how to use transaction and external ratings data is that a book that was clicked together, put into the basket or ordered together with another book in one session would make for a good recommendation for that book. Same counts for two books that were rated positively by a given user.

We leverage the transaction data using a scoring model combining a models based on (1) Apriori Algorithm and (2) Itempairs.

The Apriori Algorithm is frequently used in market basket analysis. It runs on transactional datasets and discovers association rules. We use the Apriori Algorithm to generate association rules on the levels of click, basket and ordered that were given in the transaction dataset. For this we used the apriori python package (pypi, 2021) corresponding apriori the dataset for the specific level (click, basket, order) and values for minimum support and minimum confidence as an input. The algorithm returns association rules between a given item and at least another item and a confidence value for that rule between zero and one. An association rule with a confidence of one for two items based on the ordered data series for example would mean that those two books are always ordered together. We transformed all association rules between one and n items into n pairs and combined the rule confidences from order, basket and click into one score using factors that add up to one.

$$weighappri_i = ruleorder_i * 0.55 + rulebasket_i * 0.35 + ruleclick_i * 0.1 \qquad (1)$$

The Itempairs Model transforms the underlying transactional data into pairs of items and for each pair counts how often both books were ordered, put into a basket, clicked together in one session or were rated positively (more than 3 out of 5 stars) together by one user. Some pairs have a very high occurrence of any of those four events and others a very low one, the data therefore is skewed. We logarithmize and normalize each series of occurrences using a min max scaler. For every pair of books we end up with a score between zero and one for being ordered together (*occorder*), being rated together positively (*occrating*), being put into a basket together (*occbasket*) and being clicked together (*occclick*). We combined those four scores into one score using factors that sum up to one.

$$weightpairs = occorder * 0.4 + occrating * 0.3 + occbasket * 0.2 + occclick * 0.1 \qquad (2)$$

Both submodels output a score between zero and one for any given pair of books, zero being the lowest score and one the highest. We end up with more scores from the weightpairs model, due to the fact that the Apriori Algorithm only considers connections above a certain threshold (=rule

confidence). We take the *weightpairs* score as the basis of the final score and boost those pairs for which the Appriori Model also generates a positive score using following formula.

$$weight_i = norm_{minmax}(weightpairs_i + (1 + weight\_apriori_i)) \quad (3)$$

For every book out of the test set we take the five recommendations with the best score and return them as the recommendations of the transaction model. In a last step we extend the model by introducing recursion into the algorithm by running it again to look for recommendations of recommendations until five recommendations have been discovered or the algorithm terminates because no more recommendations can be discovered. We assume that a third book, that might be a good recommendation for the recommendation of a book of the test set, would also be a good recommendation for the book out of the test set itself.

In order to prevent books with a very weak connection from being recommended, we define a cut-off value of 0.1 for the weight of each pair. All recommendations with a lower score are not returned as recommendations.

3.2 Item Similarity Model

The Item Similarity Model is based on a simple Longest Contiguous Subsequence String Matching approach; where for each of the 1,000 books in the evaluation set, a similarity score was generated against all the books in the items.csv. The features considered for generating the score were: *Title*, *Author*, *Main topic*, *Publisher* and *Language*.

We stored the matrix of similarity scores for each of the features after calculating the scores as it took around eight hours in our systems. These scores were then used to generate a single similarity score for each book in the evaluation set against all the books in the item.csv. Initially we took a manual approach to selecting optimal weights for each of the features. This was done iteratively by observing the recommendations for a set of weights. Since this method introduced subjectivity, we strived to create an automated optimizer for selecting optimal weight.

The transactions data provided a source of unbiased recommendations. Since our transaction model is based on this data, we concluded that its recommendations can be used to automate the optimization of weights. The optimizer created and selected a random set of weights and compared the recommendations produced against the ones produced by the transaction model. Effectively, our optimizer treated the weights as hyperparameters and carried out a grid search. The set of

weights that produced recommendations with the greatest overlap with the transaction model, were then selected to create recommendations for all 1,000 books. This resulted in weights that gave importance to the features in the following order, with the most important first, *Title*, *Publisher*, *Author*, *Language* and *Main topic*.

3.3 Clustering Model

Based on our observation that the provided data did not contain any personalized data we decided to apply content-based filtering methods. Thereby, in contrast to the Items Similarity Model, we aimed to utilize all categorical features contained in the items data set and compute the most similar to a specific article. This approach is based on the assumption that similar items are close in their feature space. While this assumption holds for numerical features where computing the similarity can be done by calculating the distances between elements in the vector space, we do not have any numeric features in the items data set. The features that we have are categorical, of high-cardinality, and of varying lengths. Thus, we needed to find a suitable numerical representation for the features.

We decided to transform the *Publisher*, *Author*, *Main topic* and *Sub topics* into numerical representation by assigning binary dummy variables to each unique category. For the *Title* feature, we tokenized the *Title* and removed the stop words using the word_tokenize and stopwords function of the Python nltk library (Bird, 2009). One-hot encoding the intermediate result yielded the final representation of the *Title* as a binary vector of dimensions equal to the count of all unique tokens. The new one-hot encoded dataframe was a sparse matrix with a shape of around 77,000 x 45,000. Since we did not want to feed the entire data with mostly zeros in float32 datatype to further steps, we transformed the values of the dataframe into a scipy sparse matrix for more efficient calculation and less memory footprint.

As we already had data in numerical form, we were able to continue to calculate the distance between each entry. However, we decided to oppose it. The reason is the high dimensionality of the encoded data. Besides the relatively high computational expenses, the items in very high dimensional binary vectors are almost equidistant if measured by their Euclidean distance. Cosine distance could have solved the problem to some extent, but we found it to be sub-optimal.

In the preprocessing phase, we transformed the *Publisher* and *Author* names into the same format in order to eliminate obvious differences which arise, for example, from a different order of

surname and first name or lower- or upper-case letters. However, we were not able to eliminate all differences. In addition, we observed that many *Main topics* and *Sub topics* overlap with each other. Therefore, we concluded that this model should have the ability to abstract data to a certain degree and master the versatility of data.

In order to meet the additional requirement, we tested various methods. First, we started with Multiple Correspondence Analysis (MCA), which applies correspondence analysis to the one-hot-coded data. However, we were only able to do MCA with a small training set and the result was not satisfactory. We also implemented an auto-encoder that can process the sparse matrix as input. However, the networks had problems dealing with the scarcity of data. Training ends to always predict a null vector. Finally, after testing the auto-encoder with a few different cost functions, we decided to move on to the next approach: singular value decomposition, which showed promising results. We were able to capture around 80% of the variance with 3,000 components, which is only about 7% of the previous 4,500 dummy variables. In the end, we used the sklearn.neighbors.NearestNeighbors (scikit-learn, 2021) method to find the nearest five elements to a particular element in sub-dimensional space after performing singular value decomposition and use them as our recommendation.

3.4 Amazon Model

There are a large number of websites that provide reading recommendations based on individual books. Given its popularity, large database, and longevity, we assumed that the recommendations provided by Amazon (2021b) in particular would be of high quality. Accordingly, we decided to scrape the site for the recommendations it provided for the books in our test set.

The scraping itself was done in two stages and was executed through the Java script package Puppeteer (Google, 2021b), which allows us to control and interact with a Chrome browser via an API. In the first phase, we searched for matching Amazon articles in its dedicated book search (Amazon, 2021a)based on author, title, and publisher, and saved the URL of the best recommendation. All retrieved product description pages were then manually checked and replaced where necessary.

In the second stage, we extracted the ISBNs of the recommendations, if available, directly from the saved product description pages. Since Amazon sometimes offers multiple recommendation lists, we chose the one that appears furthest up on the given website. We then reduced all

recommendations to those books that appeared in our items dataset via direct matching with the ISBN and fuzzy string matching by title, author, and publisher. Since Amazon already sorts its recommendations by relevance, we decided to keep the top five recommendations.

3.5 Level-2 Model

The four base models were developed separately. Thus, the recommendations' weights were derived on different metrics and are therefore not directly comparable. From each model we took up the best performing five recommendations, if available. We merged the best recommendations of each model based on an ordinal scale. Therefore, we assigned the highest ranked recommendation with weight one, while each additional recommendation was weighted in descending 0.2 increments. Each model received the same weight. As we were not able to objectively judge which recommendations are better than others, we did not artificially adjust the importance of a single model. If, for example, three models have ranked a certain recommendation first, but a fourth model ranked it second, the weight after merging is 0.95, i.e. $\frac{1+1+1+0.8}{4}$. Thus, we reward when multiple models give the same recommendation. We refer to these recommendations as level-1 output.

This approach potentially leads to more than five results. However, only five were needed. Furthermore, we could not objectively judge whether a recommendation from one model is better or worse than another recommendation from a different model. To come up with a solution for this problem, we developed a level-2 model that takes the total recommendations from level-1 models as input and impacts their ranking. For this purpose, we calculated an overall level-2 score per recommendation, which is the weighted average of five features. These in turn partly consist of sub features. The respective weights are always determined as a score between zero and one, with one being the best. The final weights of the corresponding features are depicted in Table 3.

Each level-1 output weight is boosted with the level-2 feature to derive the final overall weight for each recommendation. The corresponding formula is given below.

$$weight_{final} = weight_{level-1}(1 + feature_{level-2}) \quad (4)$$

We sort all recommendations per item in descending order and take the top five as our final recommendations per item.

Table 3 - Level-2 features with weights and sources

Main Feature	Main Feature Weight	Sub Features	Source	Sub Feature Weight
Topic	0.25	Main Topic Cluster Match	DMC	0.5
		External Category Match	External	0.3
		Subtopic Match	DMC	0.2
Cover	0.25	Cover Similarity Score	External	1
Rating	0.225	Weighted Average Rating	External	0.8
		Total Number of Ratings (log)	External	0.2
Pages	0.225	Pages	External	1
Price	0.05	Price	External	1

For the level-2 approach we rely mostly on external data sources. The external sources contain a lot of missing data. If a main feature cannot be computed due to missing values, the feature will be left out from the overall level-2 feature calculation, which gives the remaining features more weight. The individual features are explained in the following.

In level-1 models, we realized the importance of thematic similarity that is well described by the internal topic features. As all other features entail a lot of missing values and could not be calculated for large parts of the recommendations, we rely on internal categorical features as basis feature, additionally extended with external data. By applying K-means algorithm from the library scikit-learn (2021), we reduced the 726 different main topics to 29 clusters. For both *Main Topic* and *External Category* feature, we checked for matches between item under consideration and corresponding recommendations and assign one in case of match and zero in case of no match. We further counted *Subtopic Matches* between item and recommendation. *Main Topic Cluster* feature is assigned with the largest weight as this feature represents basic similarity while the other two specify the thematic context even further. The *External Category Feature* is assigned a higher weight compared to the subtopics feature to give external data more weight whenever possible as we did not rely on it in the level-1 models.

Since the book cover was presented to voters during voting period, we decided to include book cover similarity as feature. Therefore, we applied variational autoencoder (VAE) which is a neural network consisting of encoder and decoder. The encoder transforms the high dimensional data into the parameters of a low dimensional latent distribution. The decoder aims to reconstruct the input with the sampled representation of the simple parameterized distribution (Ardi, 2020). Applying VAE, we can capture the latent features of our cover images using only a 10-dimensional distribution. We further normalized the distances in the latent space between item and recommendation to create a score that represents cover and pattern similarity.

We further assumed a book's *Rating* to be an indicator for general popularity and therefore to be a good recommendation. For this reason, we used the *Weighted Average Rating* as feature and normalized the score to range from zero to one. Furthermore, we also added the *Total Number of Ratings* as a high value is an indicator that the book was read often and thus was sold successfully. For the latter, we assigned a weight of 0.2 as it further stresses the book's popularity but is less important than the actual rating.

For the *Pages* and *Price* feature we applied the same metric. We calculated the relative Euclidean distance between item and recommendation. After normalization, a value close to one means that item and recommendation have a very similar number of pages or price. We assume a similar amount of pages to be an indicator that books have a similar genre. While for example children's books have a small number of pages, fantasy novels are typically very long. In general, we assume a reader to prefer books of similar length over others, independent of the book's content. Similarly, the idea of the price feature is based on the assumption that voters would prefer books of similar price range over others.

For deriving the main weights, we manually reviewed the overall level-2 feature as well as the five main features in conjunction with the respective recommendations for any given item in an iterative process. Additionally, we compared extreme feature values (close to zero or one) with the underlying data. For example, if the value of the cover feature was close to one, we checked the cover of the item and the respective recommendation more closely. Correspondingly, we adjusted the weights in an iterative process.

The *Topic* and *Cover* feature have turned out to be the most important ones in this step as the link between item under consideration and recommendation is stressed most by these two features. We further see a positive but less significant impact of *Pages* and *Rating* features. As the price is not

given in the voting process and difficult for voters to estimate, we expect the *Price* feature to have a high variation, which is why we reduced its weight to 0.05. As *Rating* and *Price* have a lot of missing data, *Cover* and *Pages* features are the most important drivers for most recommendations, along with *Topic*.

4 Results

Due to the scarcity of the transaction data the transaction model was only able to generate recommendations for 308 out of the 1000 items of the test set. In comparison, the item similarity model was used to generate a ranked list of 15 recommendations for each of the 1,000 books. Thereby, it acted as a "fail-safe" in the absence of recommendations from other models. The clustering model in comparison found items from the same book series (if available) or books from the same *Author*, *Publisher*, or topic label. The model also suggested books from the same *Publisher* contained in the data with different spellings. However, since the vector multiplication of sparse vectors dilutes the information, it did not provide recommendations for very rare categories. The Amazon model provided at least one recommendation for 440 books and five recommendations for 290 books. In total, the combination of the four base models provided 12 recommendations per item on average. Table 4 depicts the different model combinations and the respective number of recommendations they produce. It turns out that the models complement each other as they deliver rather distinct recommendations. Only around 10% of the recommendations are from more than one source.

After application of level-2 approach on level-1 output only around 50% of the recommendations generated by level-1 approach remain on their initial rank. Furthermore, around 10% of the top five recommendations initially created by level-1 approach are replaced by new recommendations that were ranked up as cause of level-2 approach. While the category feature could be computed for 100% of the data due to partial usage of data provided by the organizers, *Cover* was computed for 66%, *Pages* for 52%, *Price* for 8% and *Rating* for 2%. As level-2 approach focused only on a small number of features with a lot of missing values, some good recommendations were ranked down or even replaced by others that performed well regarding the features. However, as we only need five good recommendations, level-2 approach served as additional mechanism to ensure high quality of the top five recommendations. Overall, we secured the 10^{th} place by achieving a ratio of 0.5993. The best performing team achieved a score of 0.6746.

Table 4 - Number of recommendations that result from different model combinations

Item Similarity	Clustering	Transaction	Amazon	Total Number of Recommendations	Relative Share of Recommendations
Yes	–	–	–	4,002	33.6%
–	Yes	–	–	4,084	34.3%
–	–	Yes	–	1,166	9.8%
–	–	–	Yes	1,500	12.6%
Yes	Yes	–	–	642	5.4%
Yes	–	Yes	–	89	0.7%
Yes	–	–	Yes	71	0.6%
–	Yes	Yes	–	47	0.4%
–	Yes	–	Yes	42	0.4%
–	–	Yes	Yes	35	0.3%
–	Yes	Yes	Yes	7	0.1%
Yes	–	Yes	Yes	28	0.2%
Yes	Yes	–	Yes	57	0.5%
Yes	Yes	Yes	–	96	0.8%
Yes	Yes	Yes	Yes	29	0.2%

5 Discussion

In this section, we reflect on our approach in light of the final results and the limitations we encountered. We consider our approach of developing four distinct base models to be fundamental to our success in taking one of the top ten places. By focusing two models on the item dataset, which we believed to be of most value, and creating a separate model for the transactional data, we were able to overcome the limited intersection of books in the two datasets. By scraping Amazon recommendations, we were further able to implicitly benefit from Amazon's vast repository of user-specific data that exceeds the insights from the session-based transactional data provided to us. Merging and reordering the results based on external data, in our level-2 model, finally enabled us to compensate for the potential weaknesses of the individual models.

Providing deeper insights into the performance of the individual models or our overall model is hardly possible for us. This is due to two specific limitations. First, we were not provided with any data from the final evaluation beyond our ratio. Second, the method of evaluation via an open vote on a website was highly subjective and therefore could only have been replicated by surveying a

larger group of people. Because of the associated effort and time constraints of the competition, we, like most of the top ten teams, chose not to collect such data. The fact that the three teams from Bonn-Rhein-Sieg University of Applied Sciences and Frankfurt University of Applied Sciences which did collect such user data, fully collaborated in their effort, further confirms our assessment. Nevertheless, this decision compelled us to optimize the level-1 models, their weighting in the level-2 models and the weighting of the ranking features in the level-2 model on the basis of our subjective judgement.

Due to the limitations described above, we can therefore only make assumptions about which approaches could have improved our model. Looking at the other teams, we were particularly struck by the fact that only four of the eight teams that presented their solution used external data, with the top-ranked team not among them. In addition, we noticed that some of the models used were much less complex. For example, the core of the best team's solution consisted of calculating feature similarity from five features of the item data and computing a sparse matching matrix through a formula that required “no training and about 20 seconds to run on a laptop using Python” (Appendix B). This suggests that the greatest potential lies in the similarity of core features. Ultimately, however, this can only be concluded by collecting extensive user data.

6 Conclusion

The task of the Data Mining Cup 2021 was to develop a recommendation system for a bookstore. Based on provided and gathered data, we had to make five recommendations for 1,000 books. For evaluation, the recommendations then competed against each other on a public voting website. Four separately developed models formed the basis for our approach. Two models focused on the provided item dataset using a clustering and a feature similarity approach. One model made use of the provided transaction dataset with a scoring model. A fourth model was based solely on web scraped book recommendations from Amazon. We then took the ranked recommendations of the four models and combined them into In a level-2 approach, we optimized the order of the combined recommendations based on externally collected features. As a result of our approach, we achieved a quotient of 0.5993 on the evaluation website, which placed us tenth out of 115 teams.

List of References

Amazon. (2021a). *Advanced Search: Books*. Retrieved from https://www.amazon.com/advanced-search/books

Amazon. (2021b). *Books at Amazon*. Retrieved from https://www.amazon.com/b?node=283155

Ardi, M. (2020). *Using Variational Autoencoder (VAE) to Generate New Images*. Retrieved from Medium: https://becominghuman.ai/using-variational-autoencoder-vae-to-generate-new-images-14328877e88d

Bird, S. E. (2009). *Natural Language Processing with Python.* O'Reilly Media Inc.

Data World. (2017). *Users-Books-Dataset*. Retrieved from https://data.world/divyanshj/users-books-dataset

Google. (2021a). *Google Books API*. Retrieved from https://developers.google.com/books

Google. (2021b). *Puppeteer (v10.2.0)*. Retrieved from https://github.com/puppeteer/puppeteer#readme

haowggit, L. L. (2021). *haowggit/DataMiningCup21_Team_TUMuenchen1: final solution (1.0.1)*. Retrieved from Zenodo: https://doi.org/10.5281/zenodo.5231406

ISBNdb. (2021). *ISBNdb: The World's largest book database™*. Retrieved from https://isbndb.com/

Kaggle. (2021a). *Kaggle User Data*. Retrieved from Kaggle User Data: https://www.kaggle.com/bahramjannesarr/goodreads-book-datasets-10m

Kaggle. (2021b). *Goodreads-books*. Retrieved from https://www.kaggle.com/jealousleopard/goodreadsbooks

Morton, T. S., Ingersoll, G. S., & Farris, D. (2012). *Taming Text: How to Find, Organize, and Manipulate It.* Manning Publications. Retrieved from https://livebook.manning.com/book/taming-text/chapter-4/8

prudsys. (2021). *Data Mining Cup 2021*. Retrieved from https://www.data-mining-cup.com/reviews/dmc-2021/

pypi. (2021, 08 21). *pypi apriori*. Retrieved from pypi apriori: https://pypi.org/project/apriori-python/

scikit-learn. (2021). *sklearn.cluster.KMeans*. Retrieved from https://scikit-learn.org/stable/modules/generated/sklearn.cluster.KMeans.html?highlight=kmeans#sklearn.cluster.KMeans

Seatgeek. (2020). *Fuzzywuzzy (0.18.0)*. Retrieved from https://github.com/seatgeek/fuzzywuzzy

Taylor, J. (2019). *Fuzzy matching at scale*. Retrieved from TowardsDataScience: https://towardsdatascience.com/fuzzy-matching-at-scale-84f2bfd0c536

UCSD. (2019). *Meta-Data of Books*. Retrieved from https://sites.google.com/eng.ucsd.edu/ucsdbookgraph/books

Appendix

Appendix A

Table 5 - Overview of relevant columns from merged external data

Feature	Description	Example
Title	The title of the book in the external dataset match	Mommy, Am I A …?
Publisher	The publisher of the book in the external dataset match	Avid Readers Publishing Group
Author(s)	The author(s) of the book in the external dataset match	Anila Ali, Karen Gottlieb
Category	The category of the book in the external dataset match	Juvenile Fiction
ISBN	Uniquely identifying number of a book, valid worldwide	1935105450
Description	A short description of the book	Aisha liked going to school. She liked her third grade teacher. She enjoyed working on class projects. One day, all that changed. Aisha couldn't believe what she was hearing. How could this be? What had she done? How could this be happening to her? Follow in Aisha's footsteps and discover how one incident changed the lives of many. You may even see yourself in this young girl's simple tale.....
Maturity rating	Stating whether the book is mature or not mature	NOT_MATURE
Saleability	Stating whether the book is currently for sale or not	NOT_FOR_SALE
isEbook	Stating whether the book is an Ebook or not	False

Average rating	Average rating of the book	3.5
Ratings count	The number of ratings that are available for the book	11
Binding	The kind of binding of the book	Paperback
Cover images	Link(s) to the cover image	['http://books.google.com/books/content?id=ptTlRgAACAAJ&printsec=frontcover&img=1&zoom=5&source=gbs_api', 'https://images.isbndb.com/covers/54/59 /9781935105459.jpg']
Pages	Number of pages	28
Price	The price that is known for the book at time of scraping	2.49 EUR
Country	The origin of the book	US
Language	The language of the book	eng
Date	The publication date of the book	2010

Appendix B

Screenshot of code explanation provided by team Uni_Geneva_2.

And a little explanation of our code.

For each element of a set of 1000 books, we had to come up with five recommendations taken from an inventory of about 80.000 items. Descriptive information on each books as long as historical transaction data were provided.

Pre-processing :
A first phase consisted on cleaning authors name formatting, correcting typographic errors, and merging author name string that could be considered to represent the same person.
Then, items with the same author name, title and publisher where merged, in the items, transactions and evaluation databases.
This allowed to reduce the database size, and avoid recommendation of identical books for one targeted item.

Attributes selection, features engineering and model :
Language detection was performed on the title of each book.
In order to have meaningful recommendations, we designed a set of tools to capture the context of every books. This toolbox relies heavily on the theory of N-Gram cosine similarity and Term Frequency Inverse Document Frequency matrices.
The following similarity matrices were computed :
- simTitle : 4-gram cosine similarities on title
- simAuthor : 3-gram similarities on authors
- simPub : 3-gram similarities on publisher
- simLang : similarities on language

Furthermore, this feature was also introduced :
- simTopics : a cosine similarity matrix on the topics and subtopics combined
And we included historical data using the following weighting :
- transWeight : a weight taking into account the importance of each items in the transaction database, defined as follow :
    log(e + 0.01*(0.5*(num_click) + num_basket + num_order^2))

Finally, a (sparse) matching matrix was computed, using the following formula (element wise operations):
    matches = (simAuthors*(simTitle + 1) + 5*(simTopics*(simPub + 1)) + simLang)*transWeight

Then, for each evaluated items, the fives recommendation were chosen using the five maximum matches values. This model requires no training and about 20 sec. to run on a laptop computer, using Python.