Team 6: Books Data Integration

Web Data Intergration Project

presented by
Ganna Bodnya, Yawen Liu, Minh Duc Bui, Keti Korini

submitted to the
Data and Web Science Group
Prof. Dr. Christian Bizer
Anna Primpeli
Alexander Brinkmann
University of Mannheim

December 2020

Contents

List of Figures

List of Tables

1 Phase I: Data Collection and Data Translation

Books are one of the most important information and entertainment sources available. This makes it more so important to have a consistent and large dataset about books. Our project aims to aggregate and manage data of books from different datasets into a single, homogeneous workflow. The process includes collecting data from the web, schema mapping, data translation, identify resolution, data quality assessment and data fusion.

1.1 Profiling of the Data

In our project we used three different datasets. Table 1 summarizes the profile data of each dataset. The first dataset is "Goodreads - 31 Features" dataset from Kaggle.com [5]. It contains the general information such as title, authors, number of pages and so on about the books in the "Best books ever" list on Goodreads.com. The second dataset available on kaggle.com [6] was created using Goodreads API and consists of 11123 books listed with their ID, two versions of international standard book number (ISBN), language, publisher and publication date as well as ratings and review data. The third dataset is the "BX-Books" dataset [7] - one out of three datasets included in the Book-Crossing Dataset published at the webpage of the Informatik Institute of University of Freiburg. The books data was crawled from the from the Book-crossing community bookcrossing.com. and provides information about 271,380 books.

Dataset	**Source**	**Format**	**Class**	**# of entities**	**# of attributes**	**List of attributes**
Goodreads Books - 31 Features (BestBook)	[5]	.csv	Book	52199	31	ID, title, link, series, cover_link, author, authot_link, rating_count, review_count, average_rating, 5-, 4-,3-,2-,1-star ratings, # of pages, date_published, publisher, original title, genre and votes, ISBN, ISBN13, asin, settings, characters, awards,amazon_link, worldcat_link,recommended_books, books_in_series,description.
Goodreads-books (Goodread)	[6]	.csv	Book	11127	12	ID, title, authors, average rating, ISBN, ISBN13, language code, number of pages, number of user ratings, number of reviews, publication date, publisher.
Book-Crossing Dataset (Book-Crossing)	[7]	.csv	Book	271380	8	ISBN, Book title, author, year of publication, publisher, Image-URL-S, -M and -L.

Table 1: Datasets profile data

1.2 Generation of the Integrated Schema

In order to build the integrated schema, we looked closer at the attributes of each dataset and the relationship they had with the Book entity and with each other. The total number of attributes of all datasets together was 51.

The first step taken was to distinguish the attributes whose naming was different accross the datasets but who described the same book attribute and then combine them under the same name. In the end of this step, a total of 36 distinct attributes were observed.

The second step was dimensionality reduction. Firstly, the attributes which were contained in only one dataset and the attributes which had a high percentage of missing values were dropped. Then, the attributes which did not give any useful information were dropped like the original ids and review counts. Finally, the ISBN10 and ISBN13 attributes were also dropped and were reserved only for the creation of the gold standards which were used in the second phase of the project. In the end of the dimensionality reduction step, the total and final number of the integrated schema was 8 attributes. The schema contains one class, the "Book" class and there are 5 attributes which are contained in at least two datasets, and one of these attributes is a list attribute.

Class name	**Attribute name**	**Attribute type**	**Datasets**
Book	title	string	Dataset 1, 2 and 3
Book	authors	string/list	Dataset 1, 2, and 3
Book	genres	string/list	Dataset 1
Book	rating	floating number	Dataset 1 and 2
Book	rating_count	integer	Dataset1 and 2
Book	published_date	date	Dataset 1, 2 and 3
Book	publisher	string	Dataset 1, 2 and 3
Book	pages	integer	Dataset 1 and 2

Table 2: Attribute Intersection with Integrated Schema.

1.3 Translation of the Data

After having built the integrated target schema (shown in Table 2), the translation of the three datasets followed. Mainly the correspondences observed between the datasets, were one-to-one correspondences, like the Title, Pages or Publisher attributes. There was also the case of one-to-many correspondence with the attribute Author, whose values needed to be splited so as to obtain the Author name for each of the authors of a book entity. The same with the the attribute Genre which needed to be splitted and mapped to the genre.

The first step taken includes the creation of a unique identifier for each book entity. The prefix of the id is specific to each dataset and added to that is a number which starts from 0 and increases as the rows increase.

Next step was formatting the dates. In the datasets Book-Crossing and Goodread Books this step was easier as there was only one format of the date that could be observed when studying their CSV files. There were only two cases in the Goodread Books where the day of the month exceeded the normal limit date of that month and this was fixed by substracting the date by one. The step was more challenging in the BestBook dataset as they were more than one formats to the date (e.g. "January 1st 2020"). So for this dataset, firstly the suffixes added to the days ("st", "nd", "rd"

and ”th”) were removed. Then every date was checked using regular expressions what kind of format it belonged to before parsing this date with the suitable format. A problem observed in this dataset, was the presence of negative dates that could be interpreted that those books were written BC. The formatting of these dates resulted in transforming them into the year 1 and not into a negative year.

Furthermore, the attribute Author was split so that all authors were separated entities. This step was easy in the datasets Goodread Books and BestBooks as the delimiters separating the authors were very clear. The problem was with the Book Crossing dataset where there wasn't any delimiter separating the authors. Another observation from this attribute is that the author values in this dataset are not very reliable as when they had several authors, the names and surnames of the authors were mixed up. The new attribute number of authors was calculated from the tokens produced by the splitting of the values by the delimiters.

Genre is one of the attributes which is also a list attribute, but can be found in only one database, the BestBook database. This step of splitting the genres is similar to the one employed to split the authors. The only difference is that before tokenizing the value, we remove the numbers from the string as the way that the genre values are written in the CSV file is genre and then a voting number for each genre. So after clearing the numbers, normalizing the space and tokenizing based on the comma delimeter, genres were able to be obtained.

Lastly for all the attributes of the type String, the normalize space function was applied to remove any excess whitespace.

Additionally, a problem was observed when trying to translate the dataset Goodread Books. There were some rows in the original CSV file, where because of an error that had split the authors into two columns instead of only one column, all other columns after the second one (the author one) were shifted to the right. So as not to discard those rows and lose data, a function was implemented to be able to detect these error rows and after detecting them we could concatinate the splitted authors together, and for each other attribute their values were found one attribute down. The only attribute that could not be saved from these rows was the last attribute (the publisher attribute), as, because of the right shift, the value of publisher had gone outside the normal boundaries of the table and could not be imported into MapForce.

2 Phase II: Identity Resolution

In this phase we will experiment with different identity resolutions, which identifies records in different data sets that describe the same real world entity. We also will evaluate different approaches and choose the best resolution, and with that, produce

the correspondences between records in different data sets that describe the same entity.

2.1 Gold Standard Set

To evaluate a matching rule, we need a gold standard set, which is a dataset with annotated pairs as matches or non-matches. For the creation of such a dataset, we will use the **ISBN number** to match pairs. The ISBN number is obviously a unique identifier for each book. This part was done in Jupyter Notebook, see [4]. We chose a sample size of 500 and our gold standard set will be split into three parts:

1. **Matching record pairs** (20% of GS $= 100$ samples)
2. **Corner cases** (30% of GS $= 150$ samples)
3. **Non-matching record pairs** (50% of GS $= 250$ samples)

In the following sections, we will discuss further how those different parts look like.

For matching record pairs, we will create a list of all matching record pairs based on the ISBN number for each dataset pair. After this, we will randomly choose pairs from this list and include it in our gold standard set. This makes 20% of our gold standard set.

The second part of our gold standard set is corner cases. We wanted to include three different corner cases: (1) **Books from different editions**, (2) **Books from different authors with the same title** and (3) **Books that describe the same entity but have for some reason different titles and authors**. For each corner case, we will randomly choose and fill 10% of the gold standard set with those cases each. This means for each, we have 50 different samples, and in total 150 for corner cases.

Case	Title	Authors	Published date	Publisher	Datasets
(1)	When Santa Fell to Earth	Cornelia Funke, Paul Howard, Oliver G. Latsch	**2006-10-01**	Chicken House / Scholastic	Goodread
	When Santa Fell to Earth	Cornelia Funke, Paul Howard, Oliver G. Latsch	**1994-01-01**	Chicken House / Scholastic	Bestbook
(2)	Hannibal	**Theodore Ayrault Dodge, Ian M. Cuthbertson**	2005-07-16	Barnes Noble	Goodread
	Hannibal	**Thomas Harris**	1999	Del Sol Press	Book Crossing
(3)	**Wrinkles in Time**	**George Smoot, Keay Davidson**	1994-01-10	Harper Perennial	Goodread
	Wrinkles in Time Co	**G Smoot**	1994	Perennial Currents	Book Crossing

Table 3: Examples of different corner cases with chosen attributes. Illustrates the different scenarios that we wanted to capture.

For (1) we will look for books that have the same title and same author but have different ISBN numbers. Here we used a simple normalization (remove space + lowercase) and a levenshtein distance smaller then 3 for the matching of authors.

To search for (2) we will just match the titles with different authors and different ISBN numbers. Again the name of the authors should be normalized and then the levenshtein distance should be equal or bigger than 3. For the last case (3), we will have matching ISBN number with different titles and authors. We used the same tactic as in (2) for finding different authors.

Now we will fill the rest of the gold standard set with non-matching pairs. Again, we will choose non-matching pairs that are not already in the gold standard set.

Since we have the luxuary of having the ISBN as a unique identifier, we decided to have a validation AND a test set (which is then our gold standard set), reasoning in Section 2.6. The validation and test set are being constructed like described above. Both are 500 samples big.

2.2 Comparators

We will first try to create manually matching rules and evaluate it. For this purpose we will create different comperators for different selected attributes. These attributes were selected based on how useful they would be in comparing the book entities so, in the end, our comparing attributes were the five attributes which we can find in at least two datasets: **Title**, **Date**, **Pages**, **Publisher** and **Author**. All available comparators are listed in the table 4 with their respective calculation of similarity.

Name of comparator (with abbreviation)	Calculation of similarity
BookDateComparator10Years (DC10)	$1 - (\text{diff\_between\_dates\_in\_years})$ or if diff_between_dates_in_years > 10 we set sim = 0.
BookDateCamparator2Years (DC2)	$1 - (\text{diff\_between\_dates\_in\_years})$ or if diff_between_dates_in_years > 2 we set sim = 0.
BookDateComparatorWeightedDateSimilarity (DCWDS)	Calculates the similarity between years, month and day separately and calculates the overall similarity with the given weights.
BookPagesComparatorAbsoluteDifferenceSimilarity (PCADS)	$1 - \frac{abs(\text{first\_value- second\_value})}{\text{given\_max\_diff}}$ \ and if $abs(\text{first\_value} - \text{second\_value}) > \text{given\_max\_diff}$ we set sim = 0.
BookPagesComparatorDeviationSimilarity (PCDS)	Returns 1 if values are equal, otherwise $0.5 * \frac{\text{smaller value}}{\text{bigger value}}$
BookPagesComparatorUnadjustedDeviationSimilarity (PCUDS)	$\frac{\text{smaller value}}{\text{bigger value}}$
BookPublisherComparatorJaccard (PCJ) BookTitleComparatorJaccard (TCJ)	Calculates the Jaccard similarity between tokenized strings.
BookPublisherComparatorJaccardOnNGram (PCJNG) BookTitleComparatorJaccardOnNGram (TCJNG)	Calculates the Jaccard similarity on N-Grams.
BookPublisherComparatorLevenshtein (PCL) BookTitleComparatorLevenshtein (TCL)	Calculates the Levenshtein similarity between strings
BookTitleComparatorEqual (TCE)	If titles match return 1 otherwise 0.
BookAuthorComparatorMaximumOfContainment (ACMOC)	Gives the maximum of containment.
BookAuthorComparatorMaxSimilarity (ACMS)	Return the maximum similarity between any two elements of both sets.
BookAuthorComparatorOverlapSimilarity (ACOS)	Calculates the overlap similarity between two lists of strings.

Table 4: Description of all comparators that we used.

The attributes which were of type String like Title, Publisher and Authors, were first pre-processed by removing the punctuation and the non-ASCII characters. Then

they were all lower cased and finally the attribute values were compared together. Before we are ready to manually craft matching rules, we will also define blockers, since our datasets are huge and therefore have potentially high running time without any blockers.

2.3 Blockers

Given the large number of entities in each of the three datasets, the usage of blocking is essential. Though we have tried to execute the identity resolution with no blocking to compare runtime and performance, we ran into the memory problems. We have defined 3 blocking key generators: by year of publication, by decade of publication and by title. For the datasets Goodread and BestBooks we additionally defined blocking key by number of pages. We have tried the standard blocker and sorted neighborhood blocker functions for each of the blocking key. In general, the usage of publication year and publication decade as the blocking key did not work on all of the dataset pairs most likely because the published date attribute has higher variance and is not reliable enough for identification of book entities. Here, the performance metrics for all three identity resolution runs and the number of found correspondences was extremely low. The table 5 summarizes the results for trying out different blocking keys and blocking methods for the matching rule 4 for the datasets Goodread and BestBooks.

Blocker	**Blocking type**	**Runtime**	**#matches**	**Reduction ratio**	**P**	**R**	**F1**
BookBlockingKeyByDecadeGenerator (DG)	SNB	18 sec	21	0.9994	0,00	0,00	0,00
BookBlockingKeyByTitleGenerator (TG)	SB	5 sec	4 656	0.9998	0,7718	0,7667	0,7692
	SNB	16 sec	4 780	0.9994	0,7692	0,7333	0,7509
BookBlockingKeyByYearGenerator (YG)	SNB	16 sec	55	0.9994	1,0000	0,0267	0,0519
BookBlockingKeyByPagesGenerator (PG)	**SB**	**1 min 7 sec**	**4 504**	**0.9947**	**0,9430**	**0,9933**	**0,9675**
	SNB	17 sec	1 517	0.9994	0,9672	0,3933	0,5592

SNB - sorted neighborhood blocker; SB - standard blocker

Table 5: Evaluation of blockers

The blocking key by title performed better on all three datasets pairs. Blocking books by number of pages for Goodread and Bestbook increased the recall significantly and helped to achieve the best performance metrics as you can see from the table. We also compared the sorted neighborhood blocker of different window sizes we noticed that increasing the windows size can increase the recall but it leads to the much lower execution time of the resolution. So, for logistic regression model on the datasets Book-crossing and Goodread decreasing the windows size from 50 to 30 reduces the number of correspondences from 27219 to 24397 without any change in the performance metrics and runs twice as fast as the one with windows size 50. The reduction ratio is also higher (0.997 in comparison to 0.995). The pairs quality is low, about 0.003, for different identity resolution runs.

2.4 Creating Matching Rules

To create matching rules, we will use a linear combination to combine different comparators, that we defined in Table 4. Since creating rules manually takes a lot of time, we decided to do an "exhaustive search" for combinator pairs and for combinations more than 2, we will use our intuition to built rules.

c.

Furthermore, we created different matching rules by hand based on our intuition how books can be matched and give them a name to refer to it later in the evaluation, where we will use the validation set to evaluate each rule. We built those rules on the knowledge that titles and book authors are one of the most important indicators, and with pages or published year we can differentiate between different editions.

At the beginning, we always assigned the same weight to title and author who both get most of the weight and the rest of the weight is distributed to the third comparator that is added. By experimenting with the weights, in the end we arrived in the conclusion that giving more weight to the author attribute would give the best F1-score. Finally, we experimented with adding more than three comparators. Unfortunately, by changing their weights manually we couldn't find a right combination to improve the overall best score reached by having only three comparators for every pair, so the idea of adding a fourth comparator was discarded.

Name of Matching Rule	Threshold of Linear Combinator	List of Comparators	List of Weights
Rule1:Exh-Date10Years&PublisherLevenshtein	0.7	(DC10, DC2)	(0.6, 0.4)
Rule2:Exh-AuthComp&AuthMaxSim	0.7	(ACMOC, ACMS)	(0.6, 0.4)
Rule3:Exh-AuthComp	0.7	(ACMOC,)	(1, 0)
Rule4:Pages&Title&Author	0.7	(PCDS, TCJNG, ACMOC)	(0.2, 0.2, 0.6)
Rule5:Publisher&Title&Author	0.7	(PCJNG, TCJNG, ACMS)	(0.2, 0.2, 0.6)
Rule6:Date2Years&Title&Author	0.8	(DC2, TCJNG, ACMS)	(0.3, 0.3, 0.4)
Rule7:Pages&Title&Author	0.7	(PCDS, TCJ, ACMOC)	(0.2, 0.4, 0.4)

Table 6: Creating the matching rules.

2.5 Learning Matching Rules

In this section, we will use another approach - learning matching rules based on supervised learning using the Weka library.

Before we can start training, we need to create training sets for each pair of dataset. The training sets are constructed like the gold standard set, see section 2.1, but have substantially more samples - 3000 samples for each dataset pair. We also changed the ratio of the parts of the gold standard set to have a more balanced training set ($35\%/30\%/35\%$), but unfortunately could not increase the corner cases, since we only had a limited amount of corner cases available. Of course those pairs should be selected so that it is not overlapping with the validation and test set (gold

standard set). Then we add these 16 comparators from Table 4, to the matching rule. Next, we create a rule learner to train the model. In order to speed up the process, we used different blockers. Then we execute the matching process using the match engine. Finally, we load the validation data set and evaluate our result. The final results can be found in Table 7.

Machine Learning Algorithm (Blocker)	Precision	Recall	F1-Score	Dataset
Logistic Regression (SB TG)	1.0000	0.6200	0.7654	Book Crossing+Goodread
Logistic Regression (SNB TG)	**0.9826**	**0.7533**	**0.8529**	**Book Crossing+Goodread**
Decision Tree, J48 Algorithm (SB TG)	0.8900	0.5933	0.7120	Book Crossing+Goodread
Support Vector Machines, SMO Algorithm (SB TG)	1.0000	0.6000	0.7500	Book Crossing+Goodread
Logistic Model Tree (SB TG)	0.9789	0.6200	0.7592	Book Crossing+Goodread
Logistic Regression (SB TG)	0.9036	0.5000	0.6438	Book Crossing+Bestbook
Decision Tree, J48 Algorithm (SB TG)	0.8900	0.5933	0.7120	Book Crossing+Bestbook
Support Vector Machines, SMO Algorithm (SB TG)	**0.9082**	**0.5933**	**0.7177**	**Book Crossing+Bestbook**
Logistic Model Tree (SB TG)	0.9101	0.5400	0.6778	Book Crossing+Bestbook
Logistic Regression (SB TG)	0.9500	0.7600	0.8444	Bestbook+Goodread
Logistic Regression (SB PG)	**0.9551**	**0.9933**	**0.9739**	**Bestbook+Goodread**
Decision Tree, J48 Algorithm (SB TG)	0.9421	0.7600	0.8413	Bestbook+Goodread
Support Vector Machines, SMO Algorithm (SB TG)	0.9421	0.7600	0.8413	Bestbook+Goodread
Logistic Model Tree (SB TG)	0.9500	0.7600	0.8444	Bestbook+Goodread

SNB - sorted neighborhood blocker; SB - standard blocker

Table 7: Evaluation of Machine Learning Algorithms on our validation set.

2.6 Evaluation of Matching Rules

In this section, we will finally evaluate our matching rules. For the evaluation process, we will evaluate all our matching rules and machine learning algorithms on the validation set, choose the best approaches for each dataset pair and then evaluate our best approaches on the test set (gold standard set) to get an better approximation of an unbiased metrics, e.g. precision, recall and F1-score. We will define the F1-score as our final metric. In the following Table 9, you can see a detailed description of the evaluation of each manual matching rule and in the Table 7 for the Machine Learning algorithms.

#	Matching Rule	Blocker	P	R	F1	# Corr	Time	Datasets
1	**Rule1:Exh-Date10Years&PublisherLevenshtein**	TG	0.9605	0.4867	0.6460	6396	5 sec	Book Crossings+Goodread
2	**Rule6: Date2Years&Title&Author**	**SN(30) + TG**	**0.9135**	**0.6333**	**0.7480**	**6,047**	**6 min 15 sec**	**Book Crossings+Goodread**
3	Rule3:Exh-AuthComp	TG	0.5739	0.4400	0.4981	28806	13 sec	Book Crossings+Bestbook
4	**Rule5:Publisher&Title&Author**	**TG**	**0.6667**	**0.6800**	**0.6733**	**34.504**	**1 min**	**Book Crossing+Bestbook**
5	Rule2:Exh-AuthComp&AuthMaxSim	TG	0.7055	0.7667	0.7348	5389	4 sec	Bestbook+Goodread
6	Rule4:Pages&Title&Author	TG	0.8582	0.7667	0.8099	4,792	4 sec	Bestbook+Goodread
7	**Rule4:Pages&Title&Author**	**PG**	**0.9430**	**0.9933**	**0.9675**	**4,504**	**1 min 11 sec**	**Bestbook+Goodread**

Table 8: Evaluation of each matching rule on our validation set.

We started off with a decent F1-Score computed by the exhaustive Search (Rule1, Rule2), except for the Book Crossing + BestBook dataset pair (Rule3), which

showed bad results. We then tried to improve on those results with manually crafting rules by using more than 2 comparators and using our intuition to carefully craft those. For the Book Crossing + Bestbook and Bestbook + Goodread dataset pairs, this yields a high improvement on the F1-score (Rule5, Rule4), but unfortunately we could not replicate this success to the Book Crossing + Goodread dataset pair, where we only improved the F1 slightly. The results of the Machine Learning approach were much more promising. For the Book Crossing + Goodread dataset pair, the Logistic Regression with SNB TG Blocker improved the F1-score substantially and that is why we chose this as our matching rule for further fusion. After the error analysis, see reasoning in Section 3.5, we chose a Logistic Regression without pages comparators. Moving on to the next pair Book Crossing + Bestbook, we chose the best performing one, the SVM. For the Bestbook + Goodread dataset pair, the best performing algorithm is the Logistic Regression (SB PG) but we decided to keep the manually crafted rule (Rule4 + PG), again see Section 3.5 for reason.

Now that we chose our best approaches, we can evaluate them on our test set, to get an approximation of an unbiased F1-score, see Table 9.

Matching Rule	Blocker	P	R	F1	Time	#Correspondences	Datasets
Logistic Regression	SNB TG	0.9907	0.7067	0.8249	42 min 50 sec	20494	Book Crossing+Goodread
Support Vector Machines, SMO Algorithm	SB TG	0.9000	0.5122	0.6528	50 min	88 637	Book Crossing+Bestbook
Rule4:Pages&Title&Author	PG	0.9245	0.9800	0.9515	1 min 7 sec	4504	Bestbook+Goodread

Table 9: Evaluation of best matching rule on our test set.

We also took a closer look at the group size distribution, which we struggled a lot with. For further details, see Section 3.5. For our final approaches, after merging the correspondences of BestBook + Goodread and Book-Crossing + Goodread pair, we plot the group distribution in Figure 1. As you can see the group size distribution falls exponentially with respect to the group sizes.

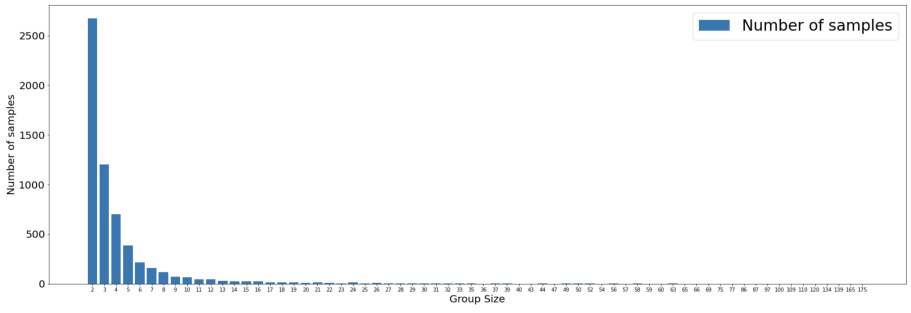

Figure 1: Group Size Distribution [4].

3 Phase III: Data Fusion and Quality Assessment

3.1 Fusion strategy and general information

For the data fusion we have selected two out of three correspondences generated from the identity resolution phase. The best performance on the gold standard was given for BestBook and Goodread identity resolution and Book-Crossing and Goodread pair.

We started with equal provenance scores but during the error analysis we set the provenance scores 3, 2 and 1 for Book-Crossing, Goodread and Bestbook respectively. The density of the input dataset as well as the merged dataset is presented in the table 10. The attribute consistencies were as follows: pages (0.93), rating_count (0.38), rating (0.75), title and publisher (0.53), authors (0.65), published_date (0.45).

	Book-Crossing	**Goodread**	**Bestbook**	**Merged dataset**
Overall density	0.50	0.87	0.98	0.95
Title	1.00	1.00	1.00	1.00
Authors	1.00	1.00	1.00	1.00
Genres	0.00	0.00	0.95	0.58
Publisher	1.00	1.00	1.00	1.00
Published_date	1.00	1.00	1.00	1.00
Pages	0.00	0.99	1.00	1.00
Rating	0.00	1.00	1.00	1.00
Rating_count	0.00	0.99	1.00	1.00

Table 10: Datasets and attributes densities

.

3.2 Gold standard

For the creation of the gold standard set, see [4], we wanted to have samples in the fused dataset that were merged from all three datasets and that the attribute values in the attribute ”Title, Authors, Pages, Publisher, Date” should be different in at least 2 datasets, since we will evaluate our fusion rules on those. The reason is that those attributes are the only ones that were reasonable to compare to some ground truth, e.g. rating is very subjective attribute which vary from source to source. Furthermore we wanted to have entities that have some kind of data conflict in those attributes, so that we can evaluate our conflict resolution functions on samples that are not trivial. We also made sure that those samples of the fused dataset are actually correct matches, again for this, we used the ISBN number. The reason is that we did not want to propagate errors from the identity resolution to our evaluation of the fusion. After picking out samples from the fused dataset, which meet the described requirements, we used the external sources ”Amazon books” [1], ”Bookshare” [2] and Wikipedia [3], to look up authors or book titles. Then, we used those sources to manually look up the attributes that we wanted to merge and fill it out in an xml file, which uses the target schema. After this our gold standard set included 15 samples.

3.3 Conflict resolution functions

The table below lists all the attributes of the fused dataset and the used conflict resolution functions as well as their description and attribute accuracy score.

Book Attributes(Fusers)	Description	Attribute Accuracy
Title(TitleFuserShortestString)	Returns the shortest string value of titles	**0.67**
Title(TitleFuserLongestString)	Returns the longest string value of titles	0.36
Author(AuthorsFuserUnion)	Returns the union of all lists of values	**0.87**
Author(AuthorsFuserIntersectionKSource(2))	Returns a set of all values that are included in at least 2 input values	0.79
Author(AuthorsFuserIntersection)	Returns a set of all values that are included all input values	0.21
Publisher(PublisherFuserLongestString)	Returns the longest string value	0.29
Publisher(PublisherFuserShortestString)	Returns the shortest string value	**0.6**
Published_date(DateFuserFavourSource)	Returns the value from the dataset with the highest provenance score	**0.8**
Published_date(DateFuserVoting)	Returns the value resulting from majority vote	0.57
Page(PagesFuserMedian)	Returns the median of all values	**0.47**

Table 11: Description of all fusers that we used

3.4 Evaluation of data fusion

To evaluate the final fused dataset, five evaluation rules, based on each of the five attributes that are present in at least two of our datasets, were created.

The rules used for evaluating attributes of type string like ”Title” and ”Publisher” were made to be more relaxed and not compare for perfect equality. For these classes we put a threshold of 0.7 that needs to be passed for the values to be called similar; ”TokenizingJaccardSimilarity” was used for non-list attributes and ”OverlapSimilarity” for the author list attribute. On the other hand, for all other remaining attributes like ”Pages” and the ”Published date”, the rules made were strict and they were called similar if their values were equal.

3.5 Error Analysis

Once the first data fusion has been run and evaluated we have seen that despite the high performance metrics in the identity resolution part, the accuracy for data fusion was as low as 0.45. Moreover, there was a problem with group size distribution so that there were present groups of up to 5000 entries. These two factors pointed on the selection bias, meaning that though our matching rules worked good for gold standard pairs, they also generated a huge number of wrong correspondences. Then, we focused on handling this issue and came back to the identity resolution phase. For the identity resolution Book-crossing and Goodread we removed the comparators for pages from the set of possible predictors for the logistic regression because we noticed that due to the high weights, the similar publisher and number of pages generated a lot of false correspondences. After the retraining the model,

the maximum group size dropped from 5905 to 2426. Then, we also made the rule stricter by increasing the threshold value from 0.7 to 0.8.

Even though Logistic Regression (SB PG) was the best performing rule for the datasets Goodread and BestBook, we decided to keep the manually crafted rule (Rule4 + PG), since they have similar F1 performance, but our crafted matching rule is much more simpler and following the "Occam's Razor" principle, we chose Rule4 + PG which achieved the best performance among the manually generated rules. It also produced about 4504 correspondences instead of 17446 correspondences produced by logistic regression. After having done this change, the group size distribution had at most 175 values (see Figure 1).

We started with the initial combination of fusers and evaluation rules which result in the accuracy of 0.48. After inspecting the fused dataset and the gold standard we could identify the following errors. The AuthorsFuserIntersection cut second or third author even though in two out of three datasets the multiple authors were named correctly. So we introduced the AuthorsIntersectionKSource(2) fuser which improved attribute accuracy from 0.21 to 0.43. Then, for title and authors we eased the evaluation rule, since there are some differences in punctuation and small spelling mistakes. This improved attribute accuracy by 0.11. Then we added AuthorsFuserUnion which increased author accuracy up to 0.87. Overall we could achieve the accuracy of 0.68. but there are still errors made by the data fusion method. For example, the pages attribute has accuracy of 0.47 and from the fused dataset we can see that actually in cases where the pages deviate from the gold standard, none of the datasets has correct value. It means that actually changing the fuser cannot help because the input data is not complete.

4 Summary

In summary, our final fused dataset contains 5962 books, which is far less than the largest dataset (Book-Crossing). But if you would add entities, that are in the fused datset but not in our largest one, you can add 1553 additional books to it, see [4]. Furthermore, we increased the density of the data from 0.5 (the score for the largest Book-crossing dataset) to 0.95 in the fused dataset, which is a significant improvement. The overall accuracy of the fused dataset is 0.68, which is being skewed by the low accuracy of the page attribute.

References

[1] Amazon books. https://www.amazon.de/b?node=186606. Accessed: 2020-12-05.

[2] Bookshare - a bentech initiative. https://www.bookshare.org/. Accessed: 2020-12-05.

[3] Wikipedia. https://www.wikipedia.de/. Accessed: 2020-12-05.

[4] G. Bodnya, Y. Liu, M. D. Bui, and K. Korini. Web data integration. https://github.com/yawliu123/Web_Data_Integration.

[5] Austin Reese. Goodreads books- 31 features. https://www.kaggle.com/austinreese/goodreads-books, July 2020. Accessed: 2020-10-21.

[6] Soumik. Goodreads-books. https://www.kaggle.com/jealousleopard/goodreadsbooks, March 2020. Accessed: 2020-10-21.

[7] Cai-Nicolas Ziegler. Book-crossing dataset. http://www2.informatik.uni-freiburg.de/~cziegler/BX/, September 2004. Accessed: 2020-10-21.