Movie Streaming Information Integration with IMDb Attributes

Web Data Integration Project

presented by Team 9
Lukas M. Loos
Matriculation Number 1560056
Ashish Rana
Matriculation Number 1822317
Subash Poudel
Matriculation Number 1754578
Honglin Li
Matriculation Number 1717048

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

December 2021

Contents

0.1 Introduction

Movie streaming services market has witnessed immense growth over the past decade with the onset of multiple streaming platforms like Netflix, PrimeVideo, Hulu, Disney+ etc. Also, streaming services platforms can gain business edge over the others by utilizing the past detailed movie data to find streaming success patterns. This would assist in predicting streaming success of new movies, old gems or hidden gems and niche/cult-favorites etc. added to the streaming catalogue. For this a consolidated informative datasource containing multiple detailed information attributes like IMDb votes, IMDb ratings, availability on other streaming, country availability, directors etc. would be required. But, there can be many challenges while preparing such a consolidated datasource from simpler ones like typographical or omission mistakes to semantically more complex one like the same movie re-released with different audio translation registered as a separate entity.

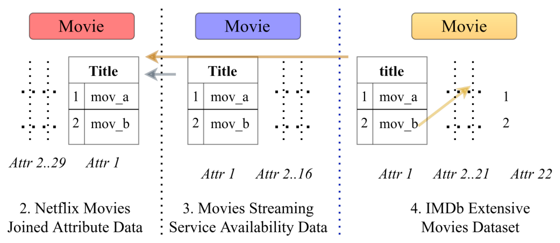
Movie
Movie
Movie
Title
mov_a
mov_b
Title
mov_a
mov_b
title
mov_a
mov_b
Attr 2..29
Attr 1
Attr 1
Attr 2..16
Attr 1
Attr 2..21
Attr 22
2. Netflix Movies Joined Attribute Data
3. Movies Streaming Service Availability Data
4. IMDb Extensive Movies Dataset

Figure 1: The functional data flow description for integrating *imdb, streaming, & imdb* movie entity data sources.

With Figure 1 as reference, in this report over the next three sections we describe: *a.* Integrated schema mapping transformations. *b.* preparing data matchers and workaround over working huge *imdb* datasource. *c.* fusing data with conflict resolving strategies. Our code and all other related inventory files are available at *url: https://github.com/Humorloos/IE683* on GitHub.

0.2 Schema Translation

For the schema translation task we used the *Pandas* library and implemented our transformations in *Python3*. We obtained all the data sources *.csv* formatted files from *Kaggle*. The detailed dataset profile information for each data source is provided in Table 1. Also, the datasets contain unique movie items with respect to all the attributes under analysis for this project use-case. Hence, we define the granularity level of the data at *movie* level and the corresponding class having a well defined integrated schema with all its attributes is listed in Table 2.

Dataset	**Source Class**	**# Entities**	**# Attributes**	**Attribute List**
IMDb.*csv*	Kaggle Movie	85854	22	IMDb_title_id, title, original_title, year, date_published, genre, duration, country, language, director, writer, actors, production_company, avg_vote, votes, budget(MV), usa_gross_income(MV), worldwide_gross_income(MV), metascore(MV), reviews_from_users, reviews_from_critics
Streaming.*csv*	Kaggle Movie	16744	16	Title, Year, Age, IMDb, Rotten Tomatoes(MV), Netflix, Hulu, Prime Video, Disney+, Type, Directors,id, Genres, Country, Language, Runtime
Netflix.csv	Kaggle Movie	15071	29	Title, Genre, Tags, Languages, Series or Movie, Hidden Gem Score, Country Availability, Runtime, Director(MV), Writer, Actors, IMDb Score, Rotten Tomatoes Score(MV), Metacritic Score(MV), Awards Received(MV), Awards Nominated For(MV), Boxoffice(MV), Release Date, Netflix Release Date, Production House(MV), Netflix Link, IMDb Link, Summary, IMDb Votes, Image, Poster, TMDb Trailer(MV), Trailer Site(MV)

Table 1: Dataset profiles for *imdb, streaming & netflix* data sources.

For transformations, since our data is available at movie granularity we directly mapped almost all of the fields with integrated schema as reference, in Table 2 this operation is denoted by *"1-to-1 direct map"*. Only for the duration field in *netflix* data source we actually transformed the string values of *('$<$ 30 minutes', '30-60 mins', '1-2 hour', '$>$ 2 hrs')* to corresponding numeric values of *(15, 45, 90, 180)*, this operation is denoted by *"if-else rounding-off"*. With list attributes we did a split operation based on the comma separator operation and added child elements to each of the parent list nodes in our final *XML* file, this operation is denoted by *"string split(',')"*. For adding the *id* field of each of the data sources we appended an incrementing sequence value to the data source name i.e. *data-source-name-inc-count*, this operation denoted by *"iterative sequence"*.

0.2.1 OOM Issues with Large XML Files

Finally, we generated corresponding *XML* files for each of our datasource which were compliant with the integrated schema. We faced *Out for Memory (OOM) errors* while working with large data sources like *IMDb*. Because of which we further improved our *XML* representations by removing empty *NaN* that were written into *XML* schema post that *Pandas* mapping stage. This reduced our *XML* file sizes for each data source as shown in Table 3. Although, minorly this improved our processing time but still *OOM errors* persisted for *IMDb* datasource. Our workaround for resolving this problem is explained in section considering its compatibility with that stage.

0.3 Identity Resolution

We use the WInte.r framework in this identity resolution phase for implementing matching algorithms to identify unique identity correspondences pairs across the data sources [2, 4]. In this section we first prepare our gold standards, resolve *OOM error* for *IMDb*, implement & compare matcher performances, plot group distribution sizes and finally, conduct error analysis from debug reports.

0.3.1 Gold Standard Preparation

For evaluation of our identity resolution algorithms, we constructed two separate gold standard comma separated *.csv* files with 500 entity pairs for performance calculation of correspondences derived from *netflix-imdb* and *netflix-streaming* matchers. Since our datasets were huge we opted for creating gold standards with 500 pairs each. First, we created our gold standard with assistance from simple matchers and blockers. Specifically, from title matcher's *Levenshtein similarity* metric

Class	Attribute Details	Parent Sources	Type & Transformation Logic
movie	genre: *list[string]*	*imdb, ntflx & strm*	1-to-1 map, string split(',')
movie	title: *string*	*imdb, ntflx & strm*	1-to-1 direct mapping
movie	actor names: *list[string]*	*imdb, ntflx & strm*	1-to-1 map, string split(',')
movie	release year: *date*	*imdb, ntflx & strm*	1-to-1 map, extract year info.
movie	directors: *list[string]*	*imdb, ntflx & strm*	1-to-1 map, string split(',')
movie	duration: *real*	*imdb, ntflx & strm*	1-to-1 map *imdb & strm*, if-else rounding-off in ntflx
movie	language: *list[string]*	*imdb, ntflx & strm*	1-to-1 map, string split(',')
movie	production company: *list[string]*	*imdb & ntflx*	1-to-1 map, string split(',')
movie	writer: *list[string]*	*imdb & ntflx*	1-to-1 map, string split(',')
movie	IMDb Votes: *integer*	*imdb & ntflx*	1-to-1 direct mapping
movie	score: *real*	*imdb & ntflx*	1-to-1 direct mapping
movie	avg vote: *real*	*imdb & strm*	1-to-1 direct mapping
movie	country: *list[string]*	*imdb & strm*	1-to-1 map, string split(',')
movie	budget: *integer*	*imdb & ntflx*	1-to-1 direct mapping
movie	original title: *string*	*imdb*	1-to-1 direct mapping
	reviews from critics: *integer*		1-to-1 direct mapping
	reviews from users: *integer*		1-to-1 direct mapping
	title id: *string*		1-to-1 direct mapping
	id: *string*		iterative sequence
movie	country availability: *list[string]*	*netflix*	1-to-1 map, string split(',')
	tags: *list[string]*		1-to-1 map, string split(',')
	hidden gem score: *real*		1-to-1 direct mapping
	image: *string*, IMDb link: *string*		1-to-1 direct mapping
	netflix link: *string*, poster: *string*		1-to-1 direct mapping
	netflix release date: *date*		1-to-1 direct mapping
	series or movie: *string*		1-to-1 direct mapping
	id: *string*		iterative sequence
movie	disney+ flag: *bool*	*streaming*	1-to-1 direct map
	prime video flag: *bool*		1-to-1 direct map
	netflix flag: *bool*		1-to-1 direct map
	hulu flag: *bool*		1-to-1 direct map
	id: *string*		iterative sequence

Table 2: Integrated schema description alongside schema translation transform operations analysis for data sources *imdb, streaming(strm*) & netflix(ntflx*).*

Datasource Name	**Before Size**	**After Size**	**Compression Ratio**
netflix	47.7 MB	39.8 MB	*(1.2)*
imdb	199.2 MB	133.2 MB	*(1.5)*
streaming	25.5 MB	12.6 MB	*(2.0)*

Table 3: Reduction in *XML* file sizes post removal of attributes containing *nan* values.

with title and release year based blocking key. We randomly added 50 *(10%)* direct complete matched entries from each of these methods amounting to 20% *(100 pairs)* for each of our gold standards. Post this, we manually added 75 *(15%)* really tough corner cases each from these methods amounting to 30% *(150 pairs)* for each of our gold standards. At last, we randomly appended 250 *50%* non-matching entity-pair correspondences to each of these gold standards.

Our initially built matchers performed really well on both these gold standards and we decided to refine our gold standards with more complex entities. Finally, both the gold standards were further then fine-tuned with additional 10% *(50 pairs)* more corner cases each from relatively more complex matchers and the additional pairs from 50% non-matching entity-pairs were removed to accommodate this fine-tuning. The matching scheme used for additions can be functionally stated as, *a.* same title and release year with at least one common director or actor and *b.* equally weighted linear matcher with title, release year, directors, genres and languages cumulative similarity greater than 0.5. We manually analyzed and added 25 confounding hard pairs *(5%)* from each matching strategy for each gold standard *(10%)* to the best of our knowledge.

0.3.2 Reduced Dataset Preparation for *imdb*

Since, *imdb* corpus is very extensive and contains a large number of movie records from multiple languages across many decades, we filter out additional movies which are not relevant for *netflix* datasource. Before, using this approach we also attempted reducing the *XML* file sizes by carrying out matcher analysis with limited overlapping column XMLs only but the *OOM error* persisted. Instead of opting for batch processing based design we eliminated the *imdb* datasource movies that are not matching with any of *netflix* movies. Specifically, we utilized Google colab's computational resources to implement a Python based filtering linear matcher and eliminated most of the movies with threshold values less than 0.3. This matcher was implemented between *netflix-imdb* data sources with an linear combination of attributes title, release year, writers, actors, and directors. Finally, we carried out our matchers for identity resolution on limited 15855 movie entities from *imdb* data source.

0.3.3 Attribute Section for Matching

There are eleven & six overlapping movie attributes between *netflix-imdb* and *netflix-streaming* datasource pairs respectively. With our empirical matching analysis we found that imdb_votes, production_campanies, duration are not reliable overlapping fields in comparison to title, imdb_score and release_year attributes. Additionally, for our datasets other list attributes like genres, actors, languages, directors and writers are highly reliable. Based on this information for the final matcher implemented for *netflix-imdb* datasets utilized title, release_year, actors, directors, writers, genres, languages, and imdb_score. And the *netflix-streaming* matcher used genres, title, release_year, directors & language attributes for identity resolution.

0.3.4 Matching & Blocking Experimentation

We systematically built linear matcher rules for the *netflix-streaming* & *netflix-imdb* data sources, starting from simple title & release year based matchers to complex matchers including list and other numeric attributes. Specifically, we found that combining list attributes with overlap coefficient similarity method, numeric attributes with permissible absolute difference limit and title with lower case Levenshtein similarity gave great metric boost as shown in Tables 4 and 5. Our final best matching rules for *netflix-imdb* & *netflix-streaming* are stated by $sim(x,y) = 0.4 \times sim_{title}(x,y) + 0.15 \times sim_{release\_year}(x,y) + 0.15 \times sim_{directors}(x,y) + 0.15 \times sim_{actors}(x,y) + 0.15 \times sim_{IMDb\_score}(x,y), sim(x,y) > 0.7$ and $sim(x,y) = 0.4 \times sim_{title}(x,y) + 0.15 \times sim_{release\_year}(x,y) + 0.15 \times sim_{directors}(x,y) + 0.10 \times sim_{languages}(x,y) + 0.20 \times sim_{genres}(x,y), sim(x,y) > 0.685$ respectively.

We utilized two title based blockers for our matching algorithms, namely: a. TitleBlocker: It uses the first two characters of titles as blocking key, achieving an average reduction ratio 96.54% . TitleSNMBlocker: It uses the first two characters of the first three words, achieving an average reduction ratio 99.72%. With our matching analysis we found that the standard blocker was more accurate but SNM blocker was really fast. Hence, we used the TitleSNMBlocker for prototyping matchers and TitleBlocker for reporting the final matcher's result, as demonstrated in Tables 4 and 5. We did not run the identity resolution without blocking because we faced the OOM issue.

#	Matching Rule	Blocker	P	R	F1	Corr	Time
1	Rule1:Title&Year	Title SNM	0.97	0.66	0.79	3904	7 sec
2	Rule2:Title&Year&Genres	Title SNM	0.98	0.71	0.82	4250	9 sec
3	Rule3:Title&Year&Directors	Title SNM	0.98	0.71	0.83	3971	16 sec
4	Rule4:Title&Year&Language	Title SNM	0.96	0.71	0.82	4587	5 sec
5	Rule5:Title&Year&Writers	Title SNM	0.99	0.71	0.83	3827	9 sec
6	Rule6:Title&Year&Actors	Title SNM	1	0.7	0.82	3429	7 sec
7	Rule7:Title&Year&IMDb_score	Title SNM	1	0.69	0.82	3481	9 sec
8	Rule8:Title&Year&Actors&Directors	Title SNM	1	0.72	0.83	3340	13 sec
9	Rule9:Title&Year&Actors&Writers	Title SNM	1	0.72	0.83	3333	15 sec
10	Rule10:Title&Year&Directors&Writers	Title SNM	0.99	0.73	0.84	3450	16 sec
11	Rule11:Title&Year&Genres&Languages	Title SNM	0.98	0.72	0.83	4278	8 sec
12	Rule12:Title&Year&Directors&IMDb_score	Title SNM	1	0.72	0.84	3525	10 sec
13	Rule13:Title&Year&Writers&IMDb_score	Title SNM	1	0.72	0.84	3525	6 sec
14	Rule14:Title&Year&Actors&IMDb_score	Title SNM	1	0.72	0.84	3372	13 sec
15	Rule15:Title&Year&Actor&IMDb_score&Directors	Title SNM	1	0.72	0.84	3316	7 sec
16	Rule16:Title&Year&Actors&Writers&Directors	Title SNM	1	0.72	0.84	3313	12 sec
17	Rule17:Title&Year&IMDb_score&Writers&Directors	Title SNM	0.99	0.74	0.84	3387	8 sec
18	Rule18:Title&Year&IMDb_score&Writers &Directors&Actors	Title SNM	0.99	0.74	0.85	3347	7 sec
19	Rule19:Title&Year&IMDb_score&Writers &Directors&Actors	Title SNM	1	0.72	0.84	3308	7 sec
20	Rule20:Title&Year&IMDb_score&Writers &Directors&Actors&Languges&Genres	Title SNM	1	0.72	0.84	3319	9 sec
21	Rule15:Title&Year&Actor&IMDb_score&Director	Title SB	0.996	0.98	0.99	3836	3 min

Table 4: Identity Resolution Performance Table for *netflix* and *imdb* data sources.

0.3.5 Group Size Distribution

The group size distribution for our final correspondences set between *netflix-streaming* & *netflix-imdb* are demonstrated in Figure 2.a plot. Here, we observe some correspondences falling onto group sizes $\{4, 5\}$, these pairs are manually analyzed to comprehend these findings.

0.3.6 Error Analysis

The near perfect F1-score performances can be somewhat attributed to the decent-quality data sources that are used for matching. But, relative performance improvements in latter reported matchers shows that adding more informative list attributes with right blocking strategy are primarily responsible for these performance improvements.

Finding the right balance is important, as with much lesser attributes the predictions get biased and highly dependent on the accuracy of those attributes which in real world data might not be 100% reliable. Whereas, adding more attributes

#	Matching Rule	Blocker	P	R	F1	Corr	Time
1	Rule1:Title&Year	Title SNM	0.74	0.62	0.67	4216	8 sec
2	Rule2:Title&Year&Genres	Title SNM	0.90	0.79	0.82	3750	10 sec
3	Rule3:Title&Year&Directors	Title SNM	0.86	0.65	0.74	4071	9 sec
4	Rule4:Title&Year&Language	Title SNM	0.83	0.63	0.72	4581	8 sec
5	Rule5:Title&Year&Languages&Directors	Title SNM	0.86	0.65	0.74	3956	11 sec
6	Rule6:Title&Year&Genres&Directors	Title SNM	0.95	0.67	0.79	3496	9 sec
7	Rule7:Title&Year&Genres&Languages	Title SNM	0.91	0.66	0.77	3516	12 sec
8	Rule8:Title&Year&Genres&Languages &Directors	Title SNM	0.98	0.70	0.81	3619	14 sec
9	Rule8:Title&Year&Genres&Languages &Directors	Title SB	0.95	0.86	0.90	3959	2.7 min

Table 5: Identity Resolution performance table for *netflix* and *streaming* data sources.

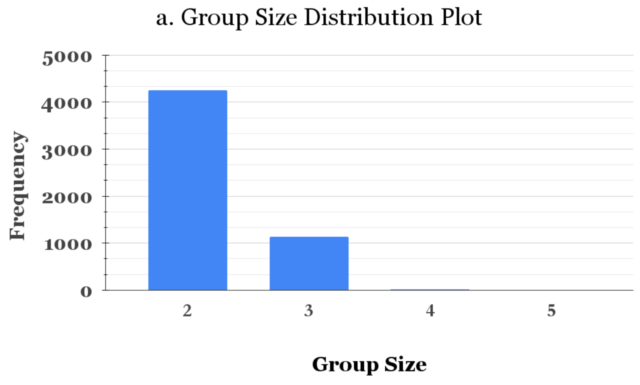

Figure 2: *a.* Final group size distribution plot.

does increase the generalizing capabilities but makes our variance vulnerable as shown by the slightly imperfect score of our final matchers.

Big Studio Movies

Big movie franchises with multiple releases in a single year like Star Wars with multiple tv-series streaming variations having a lot of common attributes like directors, actors, writers and some degree of title similarity which creates problems for our matchers. For example, a big anime brand like Naruto have multiple movies that appear in a single year with only slight variation in title having almost the same writers, actors & directors. This leaves a scope of unavoidable matches to be included in our improved matchers, pairs like { *Naruto Shippuden: Road to Ninja,* *Naruto Shippuden : Blood Prison, 2018* }, { *Naruto the Movie 3: Guardians of the Crescent Moon Kingdom, Naruto the Movie 2: Legend of the Stone of Gelel, 2014* } & { *Naruto Shippuden, Naruto Shippuden The Movie: Bonds, 2009* } were indistinguishable by our matchers.

Here, we can add more weight to the title but then we lose other correctly matched correspondences or we can use a nested matcher but that will bias our correspondences on the 100 % accuracy of smaller numbers of fields on both data sources. Both, the suggested tunings lead to small performance detriment for the correspondences against the gold standard. Hence, especially keeping such cases separately as a post-processing matching step will remove matching imperfections.

Almost *same* Movie Entities

From the group distribution plots we observed the group count of very few results exceeding upto *4 correspondences pairs* and even fraction of those results going upto *5 correspondences pairs* for our implemented final matchers for both *netflix-imdb* & *netflix-streaming* correspondences pairs under analysis. This issue can be attributed to correspondences pairs of movies that represent the same movie which is re-released in different languages, like { *Escape Plan: The Extractors, Escape Plan 3 - L'ultima sfida* }, { *26 Years, 26 nyeon* } and { *Happy End, Haepi-endeu* }. Second, it can also be the case that the same movie is made available on the Netflix platform again after its popularity again like { *Michael, increased country availability* }. But, this has to be treated as a separate new correspondence pair because the same movie is now available with multiple different attributes like changed hidden gem value, new voice actors, more available countries, changed netflix release date and additional language based writers. And this performance decrease can be attributed to our manually created ambiguity in the gold standard. Hence, our manual analysis shows that group sizes of 4 & 5 are still reasonable with different language re-release & streaming platform re-release cases in consideration.

Additionally, we also observe in between our datasources *(netflix-imdb, netflix-streaming)* that few of our movie titles and other related attributes don't match with real-life entity representation for some movies in our data sources. For example, in *imdb* the movie title for *"Crazy Rich Asians"* is wrongly annotated as *"Crazy and Asians"* in the data source. But, at the same time *imdb* provides great reliable information for other attributes. Hence, these matched entities and their corresponding attributes should be correctly resolved with conflict resolution policies which is discussed in the next section.

0.4 Data fusion

In this section we describe the different conflict resolution strategies we employed for removing inconsistencies in our final fused dataset based on the provenance information and metadata provided for our data sources [3, 1].

0.4.1 Dataset Provenance

We selected all our datasets for fusion namely, *netflix, imdb & streaming* and attempted three-way conflict resolution fusing for evaluation against the fusion gold standard. For metadata, we added weights to each dataset based on the last update they received i.e. the more recent dataset got the higher weightage, refer date-weight metadata tuples { (*imdb*: "2020-09-15"), (*netflix*: "2021-04-27"), (*streaming*: "2020-05-23") }. For provenance information, we employed a value-level provenance metadata approach for all the common shared attributes amongst the data sources. Hence, all the common attributes in fused xml datasets is having the information about the unique entity from which they are mapped from depending on the conflict resolution strategy.

0.4.2 Gold standard Preparation

We selected 15 movies that appear in three data sources at the same time. For each movie, 13 attributes are present in at least two data sources. Namely, title, genres, countries, languages, writers, actors, production_companies, release_year, directors, imdb_votes, imdb_score. Because the real IMDb website contains the values of all overlapping movie attributes between the three datasets, we compare all the overlapping attributes with the values from the IMDb website as our external information verification source. If all three data sources contain different values for a given attribute within an entity, we further verify it by Google search. For example, the release years may be different because different data sources if different language re-release cases are considered. Hence, we select release years from different countries with the oldest release information. Also, the titles may be different because of re-release titles being in different languages and in that case we selected the original english title.

0.4.3 Implementation Details

Similar to the comparators in the identity resolution stage, we also implemented 15 different fusers and evaluation rules for the data fusion step. We followed the similar workflow and approach to that of the comparators. We implemented all methods of *AttributeValueFuser* in an abstract base class, *MovieFuser* and all

Attribute	Cnstency	Attribute	Cnstency	Attribute	Cnstency
languages	0.35	imdb_votes	1.00	directors	0.45
release_year	0.89	countries	1.00	writers	0.28
title	0.88	imdb_score	0.83	duration	0.47
actors	0.30	production_companies	0.99	genres	0.65
budget	1.00				

Table 6: Consistency shared column metric *Cnstency** per attribute for all the common attributes shared amongst the three data sources.

methods of *EvaluationRule* in *MovieEvaluationRule*. Then we extended those abstract classes on two levels to first specify attribute type specific details and then attribute specific details.

0.4.4 Conflict Resolution Functions

We utilized the same conflict resolution functions for attributes of the same type. For list type attributes (actors, countries, directors, genres, languages, production companies, writers) we used *Union* to include all possible options from the source datasets in our fused dataset. For numeric attributes (avg vote, budget, duration, imdb score, imdb votes) we used *average* because for these attributes, we found it more reasonable to interpolate the sources' values rather than using the median. An exception to that logic is the attribute year's resolution. For the class *YearFuser* we furthermore created a custom conflict resolution function named *Minimum*. So, that our fused dataset would contain the earliest release year in case movies were re-published later on certain regional platforms. Finally, we chose the conflict resolution function `LongestString` for our only overlapping string type attribute title, to retain as much information from the original sources in our fused dataset as possible. We only tried these specific rule combinations for our attributes because conflict resolution was especially time consuming considering our large datasets containing lots of overlapping list attributes.

0.4.5 Evaluation Metrics

Tables 6, 7 and 8 show the result of column consistency, density and accuracy metric evaluation of our fused dataset for the shared attributes. We have only highlighted the shared column level metrics as they help in evaluating the conflict resolution strategies impact in more detailed manner.

Attribute	Den.	Attribute	Den.	Attribute	Den.
languages	1.0	imdb_votes	1.0	directors	1.0
release_year	0.94	countries	1.0	writers	1.0
title	1.0	imdb_score	0.87	duration	0.91
actors	0.93	production_companies	0.94	genres	0.99
budget	0.95				

Table 7: Density metric *Den.** per attribute for all the common attributes shared amongst the three data sources.

Attribute	Acc.	Attribute	Acc.	Attribute	Acc.
languages	0.81	imdb_votes	1.00	directors	0.85
release_year	0.89	countries	0.75	writers	0.78
title	0.91	imdb_score	0.74	duration	0.64
actors	0.75	production_companies	0.72	genres	0.96
budget	1.00				

Table 8: Accuracy metric *Acc.** per attribute for all the common attributes shared amongst the three data sources.

0.4.6 Error Analysis & Limitations

Our manual error analysis demonstrated the limitations of using these simple resolution strategies. For example, the longest string might not be the most accurate attribute value; it simply be the one with more escape sequences. Second, even though we assign weightage to each dataset's credibility based on their latest updates in actual empirical practise each dataset provides some attributes in a highly accurate manner whereas the other one simply contains relevant enough noise to decrease its weightage.

0.5 Summary

In this project we systematically merge data from three data sources, namely *netflix, streaming imdb* by applying mapping transformations, finding resolution correspondences with matchers and fusing the conflicting entity attributes. Compared to the largest dataset *netflix*, we reduced to only 4976 additional entities, density is increased by 90 % for common attributes on average. The overall accuracy of our final dataset is 79% according to our gold standard.

Bibliography

[1] Oliver Lehmberg and Christian Bizer. Stitching web tables for improving matching quality. *Proceedings of the VLDB Endowment*, 10(11):1502–1513, 2017.

[2] Oliver Lehmberg, Christian Bizer, and Alexander Brinkmann. Winte. r-a web data integration framework. In *International Semantic Web Conference (Posters, Demos & Industry Tracks)*, 2017.

[3] Oliver Lehmberg, Dominique Ritze, Robert Meusel, and Christian Bizer. A large public corpus of web tables containing time and context metadata. In *Proceedings of the 25th International Conference Companion on World Wide Web*, pages 75–76, 2016.

[4] Dominique Ritze, Oliver Lehmberg, and Christian Bizer. Matching html tables to dbpedia. In *Proceedings of the 5th International Conference on Web Intelligence, Mining and Semantics*, pages 1–6, 2015.