Project Stage 3: Blocking

Project Description
This part of the project focus on developing a blocker on two datasets before we start to do matching on them. Two movie databases are chosen for this part of the project. The data are crawled from the websites and stored in CSV format. Seven attributes/columns are selected and cleaned for blocking.

Summary
The Blocking is split into 3 stages:
Stage1: Develop potential candidate blocker using base attributes
Stage2: Build blockers on top of selected potential blockers from Stage1
Stage3: Develop final blocker from Stage2

Final Blocker Design
Stage1:
Candidate 1 is obtained by using Magellan's Overlap Blocker on the **"Title"** attribute.
Candidate 1x is then obtained by using Edit Distance Similarity with a Threshold of 0.4 on Candidate 1

Stage2:
Candidate A is obtained by using **"Director"** with Magellan's Overlap Blocker on Candidate 1x
Candidate B is obtained by using **"Category"** with Magellan's Overlap Blocker on Candidate 1x
Candidate C is obtained by using **"Duration"** with Magellan's Attribute Equivalence Blocker on Candidate 1x

Stage3:
The final blocker is obtained using the union of Candidate A, Candidate B and Candidate C.

Data Introduction
Dataset A: IMDB Data
Dataset B: Rotten Tomatoes Data
Blocking is needed to reduce the size of Dataset A x Dataset B
Size of Dataset A = 21701 rows
Size of Dataset B = 6921 rows
Size of Dataset A x Dataset B = 150,192,621 rows

Selected Potential Attributes for Blocking:
ID: Intger, Unique ID for each row
Title: String, name of the movie
Category: Sets of strings, genre for the movie
Duration: Integer, in minutes
Rating: Integer, x/100, average rating of the movie out of 100
Rating Count: Integer, number of people that contribute to the rating
Director: String, name of the director of the movie

Development Process

Cleaning and Information Extraction Stage

Dataset A: IMDB

1) Combine the two columns "director" and "creator" into one single "director" column since they are the same thing
2) Remove white spaces, extra commas at the end of string, remove utf-8 encoding, convert everything to ascii
3) Convert duration to "x minutes" instead of "x hour y min" format
4) Multiply the rating by 10 to convert it to an integer
5) Remove the comma in rating count to convert it into an integer
6) Replace N/A and empty values to Null

Dataset B: Rotten Tomatoes

1) Multiply the rating by 20 because rotten tomatoes rating is x/5 and we want it to be x/100
2) Remove the comma in rating count
3) Remove the comma in rating count to convert it into an intger
4) Replace N/A and empty values to Null
5) Some categories have an '&' in the string such as 'Anime & Manga', split it into two different category to match IMDB's style
6) Remove white spaces, extra commas at the end of string, remove utf-8 encoding, convert everything to ascii
7) Remove the string "minutes" from duration

Categories Blocking

Categories and their counts. Have to clean this and match them because IMDB and Rotten Tomatoes named their category differently.

IMDB Original:

['Musical', 'Horror', 'War', 'Western', 'null', 'Comedy', 'Animation', 'History', 'Fantasy', 'Music', 'News', 'Short', 'Adult', 'Biography', 'Reality-TV', 'Talk-Show', 'Adventure', 'Action', 'Drama', 'Game-Show', 'Romance', 'Film-Noir', 'Sci-Fi', 'Thriller', 'Crime', 'Sport', 'Mystery', 'Documentary', 'Family']

ROTTEN Original:

['null', 'Comedy', 'Kids', 'Manga', 'Sports', 'Lesbian', 'Family', 'Documentary', 'Science Fiction', 'Adventure', 'Action', 'Fantasy', 'Drama', 'Anime', 'Spirituality', 'Animation', 'Special Interest', 'Western', 'Television', 'Cult Movies', 'Musical', 'International', 'Art House', 'Mystery', 'Horror', 'Fitness', 'Classics', 'Performing Arts', 'Gay', 'Faith', 'Suspense', 'Romance']

IMDB	Rotten Tomatoes
Music, Musical	Musical
Horror	Horror
Western	Western
Comedy	Comedy
Animation	Animation, Anime, Manga
History	History
Fantasy	Fantasy
Adult	Lesbian, Gay
Game Show, Reality TV, Talk Show	Television
Adventure	Adventure
Action	Action
Drama	Drama
Romance	Romance
Sci Fi	Science Fiction
Mystery, Thriller	Mystery, Suspense
Sports	Sports
Documentary	Documentary
Family	Family

Category that does not match:
IMDB: War, History, News, Short, Biography, Film Noir, Crime
Rotten Tomatoes: Kids, Spiritually, Special Interest, Cult Movies, International, Art House, Fitness, Classics, Performing Arts, Fatih, Suspense

Blocking First Stage

Started off with a blocker for each attribute.

Candidate 1: Block on Title using Magellan's Overlap Blocker

Results:

Reduced to 1,794,800 rows, should be able to reduce to a smaller number because there are a lot of irrelevant matches

Debugger shows that there are little to none potential matches that are dropped, so we can assume that this candidate is safe

After Revised:

Candidate 1x: Block on Candidate 1 using Edit Distance Similarity with a Threshold of 0.4

This blocker is tested with other threshold and a Threshold of 0.45 gives the best result while, 0.5 and above dropped a lot of potential matches

Results:

Reduced to 100,789 rows

Debugger shows similar to result of Candidate 1's debugger with a few additional potential matches dropped (around 4 or 5)

Result is promising, will try to build blocker on top of this Candidate blocker

Candidate 2: Block on Duration using Magellan's Attribute Equivalence Blocker

Results:

Reduced to 1,421,519 rows, a lot of irrelevant matches that exist

Debugger shows that a lot of matches are dropped, most of them has a similarity measure of 0.8 and above

We decided to not use Candidate 2 for blocking

Candidate 3: Block on Director using Magellan's Overlap Blocker

Results:

Reduced to 690,292 rows, however this candidate is not ideal because there is a lot of Null values

Debugger shows that a lot potential matches are dropped because some directors have Null values

Candidate 4: Block on Category using Magellan's Overlap Blocker

Results:

Reduced to 3,347,432 rows, a lot of irrelevant matches

Debugger shows that a lot of potential matches dropped because both Datasets uses different types of categories

Will try to match the categories by hand and cleaned the data based on the categories before using this to block

Revised (After cleaning the data):

Reduce to 3,391,344 rows, still a lot of irrelevant matches, only reduced to a third of the total matches

Debugger shows slightly better but similar results

Blocking Second Stage

Decided to use Candidate 1x from the First Stage:
Candidate 1x uses Magellan's Overlap Blocker then apply Edit Distance Similarity with a Threshold of 0.4
The size of the table is reduced to 100,789 rows, which is only 0.1% out of the total size
From here, we will call Candidate 1x = Candidate X

We applied more blocking on the Candidate X obtained from the First Stage

Candidate A: Block using Rating with Relative Distance on Candidate X
Result:

With relative distance < x, where x is 0.1,0.2,0.3

We managed to reduce the size but a lot of potential candidates are dropped when tested with the debugger.

Hence, this is not a good method.

Candidate B: Block using Director with Overlap Blocker
Result:

Reduce to 15,417 rows, Debugger shows only a few potential matches dropped

Candidate C: Block using Category with Overlap Blocker
Result:

Reduce to 5,085 rows, Debugger shows a lot of potential matches dropped

Candidate D: Block using Duration with Equivalence Blocker
Result:

Reduce to 6,966 rows, Debugger shows a lot of potential matches dropped

Although Candidate C and D managed to reduce the size significantly, a lot of potential matches are dropped. Hence, we figured that the best solution would be a union of Candidate B, Candidate C and Candidate D.

Blocking Final Stage

After developing some secondary blockers on top of Candidate X, we plan to union the results of potential Candidates (B,C and D), (B and C), (B and D), (C and D)

The best result that we obtained is union_A which is the union of Candidate (B,C and D)
Union_A gives the best reduction and the least potential matches lost
The final size is 23,331 which is 0.016% of the total size

Questions and Answers

How did you develop the final blocker? What blocker did you start with? What problems did you see? Then how did you revise it to come up with the next blocker? In short, explain the *development process*, from the first blocker all the way to the final blocker (that you submit in the Jupyter file).

Answer can be found in the development process.

If you use Magellan, then did you use the debugger? If so, where in the process? And what did you find? Was it useful, in what way? If you do not use Magellan, you can skip this question.

Yes, I used Magellan and the debugger. Throughout the whole process of doing blocking and developing potential candidates. Yes, it is very useful to decide if your blocker is working properly or not. The debugger is used to narrow the potential blockers.

How much time did it take for you to do the whole blocking process?

To generate Table C, it takes around 7minutes. To come up with the blocking process, it took me around 6 hours.

Report the size of table A, the size of table B, the total number of tuple pairs in the Catersian product of A and B, and the total number of tuple pairs in the table C.

Size of Table A = 21701 rows
Size of Table B = 6921 rows
Size of Table A x Table B = 150,192,621 rows
Size of Table C = 23,332 rows
Size of Table C / (Table A x Table B) = 0.01553471791% ~ 0.016%

Did you have to do any cleaning or additional information extraction on tables A and B?

Yes, we have to do some cleaning to standardize Table A and Table B. More information on cleaning can be found in the development process.

Did you run into any issues using Magellan (such as scalability?). Provide feedback on Magellan. Is there anything you want to see in Magellan (and is not there)? If you do not use Magellan, you can skip this question.

One big issue that I ran into is that whenever I try to run Magellan (more complex blocking) on a table size that is too large, my laptop will crash. It's probably my laptop's fault for not having enough memory. That is the reason why I reduce the table size before doing more intensive blocking on it. I am not sure how Magellan works but it will be great if Magellan can somehow store part of the data on disk instead of the memory to reduce lag or crash while its doing something with the data if the data is too large. (Just a suggestion)