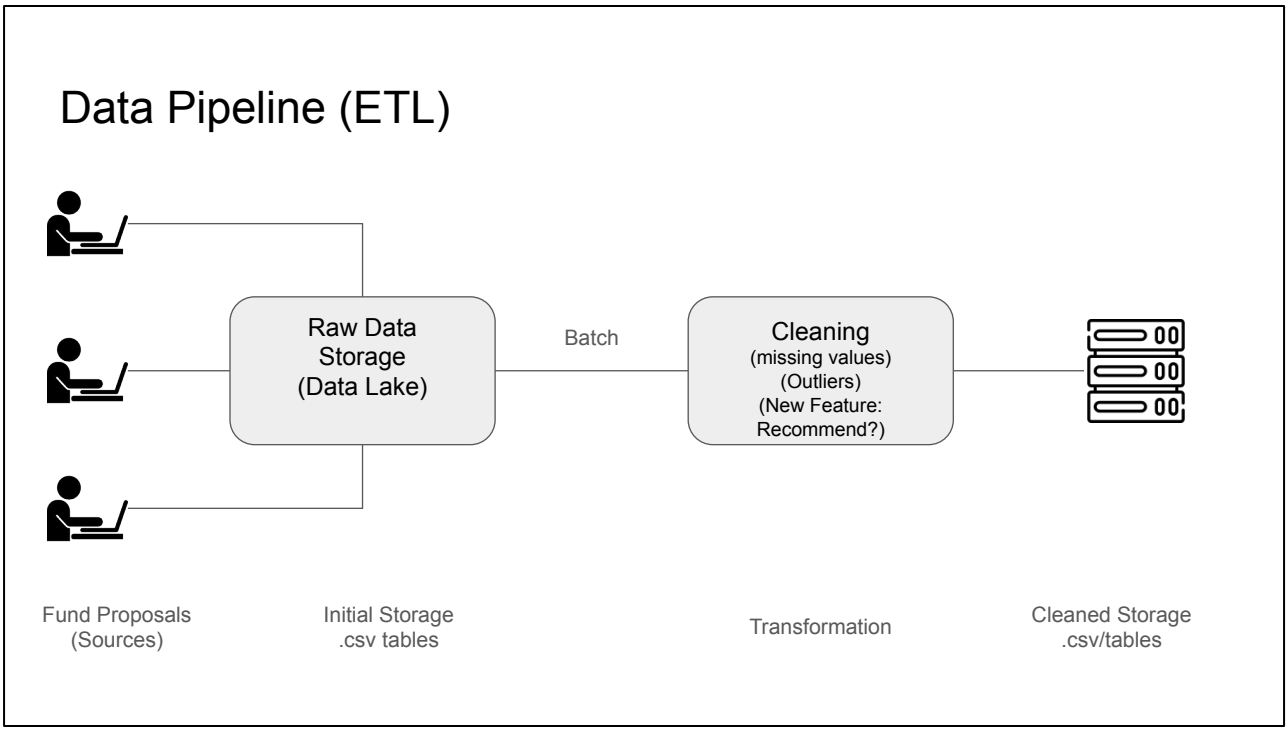
Data Pipeline and Quality Control

Kyle Mellors

MSDS 610 Week 3



1. **Source:** Movie script writers or their representatives fill out a proposal form on the company's website for a request of funds. Each request generates a .csv table that includes the features that align with the online form boxes (I.e.: Funds Requested, Estimated Budget, Movie Title, Tag Line, Summary, Genre(s), Estimated Release Date, etc. The forms will be a mix of text, date, and currency (number) data formats.

2. **Initial Storage:** Once the writer submits the form it is transferred to a data lake where each form would be stored as its own table with the raw data. **Professor FeedBack Request:** I was also considering if it would be possible for each separate entry to populate the same table at this part of the process, since it would likely save resources/storage space and then once the table (or time) has reached its batching rules, than only one table would be submitted, instead of multiple tables stored and then be fused together into one.

3. **Batch:** Once the storage receives an X amount of entries OR an X amount of time has passed since last batch processing, the tables will be collected and transformed into one table -- which would be a master table with the same features each of the entry tables would have. I was considering whether this should be batch or streaming, because of timeliness. Given that writers are likely submitting proposals to multiple companies and shopping around, the sooner an offer should be submitted to the writer. I would make the argument that if the organization is small and fund proposals are inconsistent, streaming would be the way to go, since they would want to process the request as

1. near-real-time as possible. Alternatively, I would recommend a large company that would get 1000s of proposals in a day or a week to use a batch processing scheme, since they would still be processing close to real-time and have plenty of options if another company grabs up a recommended proposal.

2. **Transformation:** This is where the clean up would occur. Depending on the feature, different rules would need to be in place to handle different situations. For instance, if the “Funds Requested” is either 0, missing, or over X amount (determined by how much the company is willing to invest in a film) then the row (proposal) would get dropped (rejected), since this is the overarching reason for the proposal to exist to begin with. The same would be for “Estimated Release Date”, if no release date than the proposal gets rejected. To prevent these situations from occurring, the recommendation would be for the organization to make each field a mandatory field (plus, if the requester was serious about the request, they would be as detailed as possible). Since there is text-based data, the text would need to be cleaned up for standard analysis. A new feature would need to be developed “Cleaned Data” that would include the removal of punctuation, standard lower case, removal of stop words and symbols, etc.

3. **Final Storage:** A data warehouse where the different batches of cleaned data tables (.csv) would be stored for retrieval and analysis. I am choosing CSV table because the proposal recommendation features will need to be compared against the movie data set that it would be compared to. It would have to generate a “similarity percentage” based on the known data of our current movies and our fund request proposals.

Recommender System Vision / Professor Feedback Request:

When I imagine this recommender system, I imagine a cleaned batch table to be retrieved and put through ML models that run each row (proposal) against the TMDB movie data set and make a recommendation on whether the company should invest in the request. What I envision is that the TMDB would include a feature “Was it Successful?” (which would be in terms of profit, and the answer would be yes if “Revenue” was equal to or higher than 2.5 time the budget) that I would have to generate. And that my ML model would scan each proposal (looking at genres, keywords, themes) and then weigh it against each movie in the data set. Each movie would have a “weight” assigned to it for how similar it is the request. The model would then count the number of “similar” movies (based on some “weight” rules), look at if those similar movies were (or weren’t successful) and make a determination - based on the number of similar movies and a successful/unsuccessful ratio (another rule that would be determined), and with those numbers combined the recommender system would either recommend the company fund the film, not fund the film, or doesn’t have enough data: For instance, if a proposal came in that only had 2 similar

movies and they were both successful, the recommendation would be “Not Enough Data”, since only 2 similar movies - even though they were successful - exist it wouldn’t be enough to show a pattern of success and therefore the recommendation would be for someone to manually read the request and make the decision to fund (or not). Alternatively, if a proposal found 15 similar movies and 12 of them were successful, then the recommender system would recommend the company funds the film (the opposite also being true). So this would require the ML model to have a rule that makes the decision on whether it has enough information and, if so, to make a recommendation decision. This means that there would have to be a minimum number of similar movies (such as at least 20% of the text features have keyword similarities) and that a threshold has to be set to distinguish between recommend or not recommend (like 70% of the similar movies were successful). The result would be a spreadsheet with all of the proposal of a batch with a recommendation decision.

To add, I don’t know if it makes more sense to find the similarity based on one feature (how alike are the known movie summaries are to the proposal films) or if it makes more sense to generate “similarity percentages” that lead to an “overall similarity percentage” that goes across multiple features (%similar for Genre,Summary, Keywords, etc. - and then an average %similarity score is generated and use against the threshold to determine if the movie is similar enough to be considered similar -- I will have to think on this, and any feedback is appreciated.

This would have to be a if/else situation: If the movie doesn not have enough similar movies (6) recommend “Not enough data”; Else if the movie has enough similar movies, find the success percentage of the similar movies (aggregated) and make a “Yes” recommendation if the success percentage is over 70%, else make a “No” recommendation”

My feedback request is two fold:

1. Does this recommendation model make sense and is it doable within the context of this class? Am I under thinking or overthinking this model?
2. If I need the proposal table to include 3 additional features “# Of Similar Films”, “Success % of Similar Films”, and “Recommendation”, would I generate these during the transformation step of my pipeline (knowing they will be blank, but filled in at the ML/analysis phase) or is better to generate the new features at the ML Level of the process?

Thank you!

Data Quality Issues - With the Known Dataset

Since my dataset is text-based and number based, there is the potential for data quality issues in either section.

- Duplicate entries: If the same movie is entered more than once, all but one instance would be dropped
- Missing Numeric Data: If the Budget data is missing, I would research and enter the data manually (if few entries), or I would impute the data of those with similar genres and keywords.
- Missing Text Data: If Genre or Summary (or other text boxes) are missing, I will research them on IMDB and copy/paste the relevant information from there.
- Outliers: If outliers exist, I will research their accuracy and fix them. If there is many outliers, I will likely impute them based on other films in the genre.

Data Quality Issues - With the Incoming Data (Proposals)

Since the data inside would be text-based and number based, there is the potential for data quality issues in either section.

- Duplicate requests: A requester is spamming requests in hopes the proposal would be seen and approved: Duplicate requests would be dropped
- Missing values: If features like “Amount of Funds Requested” is missing, those requests would be dropped - its the core of the proposal.
- Wrong Value Type: If the “Budget” feature has letters instead of numbers or it is less than the requested amount of fund it would be dropped
- If text is missing from a text feature, such as genre, tagline, a generic text will be included, such as “Information not provided”**

** Missing Text Note: I am concerned on how the best way to handle this is. If it is in the training data, I would be keen on manually filling out the text, like the genre and summary - since we want our training data to be as accurate as possible. If the data that is missing is coming from a proposal request, I would be more keen to have it autofill with a generic statement, so that it still allows the proposal to be processed -- but it will suffer when it comes to defining similarity scores (which would be crucial in predicting recommendation) and likely either reject the proposal or not recommend. Ideally, the proposal form would have all sections as requirement, and if they put in random strings in the text boxes, that would still hurt their chances of getting funds approved.

Detecting and Monitoring Drift

One of the ways the company would monitor drift would be to keep statistics on each of the recommendation decisions. For instance, if the model starts rejecting 100% of the requests, it would be an indicator that something is wrong with the system, which would require additional testing and possible retraining.

If there is no noticeable drift, the company would still want a way to test the model. My recommendation would be to keep a separate ongoing CSV file of the proposals (with their initial recommendations) that is updated with their actual end results (Was it Successful or Not?) and compare initial recommendations to the actual end result and see if the model is performing well.

Additionally, the separate CSV with the initial and end results of the proposals could be separated with a table for the films where the “initial recommendation” matched the “end result” (i.e. recommended the film and it succeeded, and didn’t recommend the film and it failed) and run that through the model to test if the model maintained it is accuracy

The proposed films, once they are released, will be added to the dataset the proposals are tested against. Along with any other films released. I would recommend the training data be updated with latest releases at least every 6 months.

Loaded Cleaned Data in PGAdmin - Screenshot