ADS508 Data Science Design Document

Part 1

Authors: Sarah Alqaysi, Hanmaro Song and Sean Torres
Company Name: Alta Inc.
Company Industry: Movies Recommender
Company Size: Mid-size company with 500+ employees and $100 million in annual revenue

Abstract:
Alta, is a movie recommender company. It operates by acquiring new content to maintain viewership, by making regional recommendations based on genre and rating score, and suggesting what titles to pick up to maintain viewing time.

Problem Statement:
Alta is a growing company that aims to connect broadcasting/streaming networks to attractive content that can escalate viewership growth. Using sites like IMDB will give some powerful insights on performance of films so that we can acquire the rights to franchises before our clients competitors.

Predictions of potential film quality will help Alta to figure out what genres, actors, and keywords in order to sift through the **2,577 films** produced each year around the world. In addition, machine learning algorithms will help improve suggestions towards certain client preferences in certain regional network hosts when our client introduces us to their user data.

Goals:

1. Analyze IMDB and Netflix dataset and its contents
2. Utilize pretrained model(s) from libraries such as Tensorflow, Pytorch, and/or HuggingFace to analyze semantics of reviews and movie information such as synopsis
3. Feature selection and elimination. Remove any non-alpha-numeric characters such as emoticons in features that are in string data type. We can still keep comma, dot, and space

Non-Goals:

1. Build a model (Bert or Transformer) from scratch as they are already available to us, which will take too much time for training and validating
2. Preprocess (**Lemmatization**, **Stemming**, etc) words and characters for features like synopsis, **if** we are to use Transformer since there are many articles and reviews that they do not much improve the final performance of semantic analysis. Pretrained transformer models can distinguish a word “bank” that has two different meanings, based on the whole sentence it’s in
3. Using user data in order to find a film related to their preference

Data Sources:

IMDB Movies Analysis
https://www.kaggle.com/datasets/samruddhim/imdb-movies-analysis

IMDb Dataset - From 1888 to 2023
https://www.kaggle.com/datasets/komalkhetlani/imdb-dataset

Netflix
https://data.world/alphatango90/netflix-analysis

There are two different sets of data sources for **IMDB** and one for **Netflix**. The two sources have overlapping data but both have more features that the other doesn’t have such as plots, episodes information, and *AKA* titles. To merge all the tables, we will be using IMDB ID which both data sources have (but in different format). The first dataset has around 40k rows while the second has more than a million but once they are joined, the final result will have about 40k records.

There are quite a few different ways to get it delivered but at the moment, we will be loading them directly from S3. If we find that we need to set up a database, one will be chosen appropriately after discussion. Because we can access the data from S3, connection to AWS Sagemaker can easily be configured using boto3 and other necessary libraries.

One potential risk is there is no distinct mapping between these three datasets that can tell us which movies from IMDB are ones in Netflix (or vice versa). In this case, we may have to do fuzzy matching on titles (in addition to any important feature) to generate a mapping table.

Part 2

Data Exploration:

Detail what you will look for in the data during the exploration phase. Consider data quality, potential data bias, key fields, data types, etc.

- Where will you be storing your data, how will you get ingest it there?
    - Data will be stored in S3 bucket
- What tools will you use to ingest and explore the data?
    - Since the data is in S3, there is no extra configuration needed to be done. Pandas library can directly load any data from S3 to Sagemaker with read_csv(or any other format) using '**s3://{bucket_name}/{filename}**'. One may have to boto3, if needed
- Detail what you will look for in the data during the exploration phase. Consider data quality, potential data bias, key fields, data types, etc.
    - With pandas.info(), find if there are (or how many) null or missing values
    - Check data types of each feature
    - Figure out distribution of features using histogram, bar plot, and any other necessary plots (matplotlib)
    - Figure out which column(s) can be used as a linking field (unique identifier) to map between each table. If there is none, figure out how we can create such
- Implement your data ingestion (if done via code) and data exploration using SageMaker studio notebook. Store your code in a github repository and provide a link to the code used for ingestion and exploration.
    - https://github.com/seantorres/Movie-Recommender-System/blob/main/Data%20Prep.ipynb

Measuring Impact:

List at least two specific metrics you expect to change with this project. These should tie back to the goals above.

- F1 score and AUC are the two metrics that we're anticipating to use as we measure how well our predictions/recommendations are ranked.
- Cosine similarity score will be used to compute the distance between two movies and check how similar they are based on synopsis (can also be used on Genre)

Security Checklist, Privacy and Other Risks:

- Will this store or process PHI data?
    - No, this dataset does not consist of personal health information so it would neither be stored or processed.

- Will this store or process PII data?
    - We have no PII data being accessed that can identify a full name or an id number system. So neither is being stored or processed.

- Will user behavior be tracked and stored?
    - User behavior will not be taken into consideration during the analysis of content ratings. However we could implement a system that tracks user like ratings if PII data does become available from Netflix.

- Will this store or process credit card data?
    - Credit card data is currently not considered within the analysis since we are not using user data to make recommendations.

- What S3 buckets will this application read or write to?
    - We are currently reading from a standalone standard S3 data bucket since we are using a structured format. No data will be written to the S3 bucket since none of the values will be rewritten.

- What data bias should be considered?
    - Survivorship bias is one our company will constantly look for while making a recommendation to their viewers, especially when the customer has based his review from a very personal point of view that is not directly related to the movie overall.
    - Availability bias could be another one to look for which could be overcome if we increase the data resources that we feed our recommendation system to, for example, Rotten Tomatoes or Amazon Prime.

- Are there any ethical concerns with the data or business problem that should be addressed?
    - Being a movie recommender company, ethical filters should be considered. Tatenda addresses a great ethical concern with a recommendation on Meduim.com by suggesting, “Provide user-adjustable “ethical filters” that screen the items that can be recommended based on the user’s specified ethical preferences. This gives users control over the sensitive aspects of recommendations”.
    - Privacy is another ethical consideration to look for. The input we get and feed our recommendation system is based on personal preferences which in turn might lead to 3rd party companies obtaining this personal information and misusing it.

Part 3

Data Preparation:

How will you transform the raw data so that it is ready for training?

- Data Scrubbing: What data cleansing techniques will you apply?

- Any columns in string data type that have nan values will be replaced to empty string
- Filter out titles that do not have plot/description (or synopsis) value as this is one of key features we use to recommend titles
- Remove any non-alphanumeric characters in columns with str dtype
- Because Netflix has a bit different format for genre column from IMDB, manually inspect distinct genres and match them to IMDB genres so that we can compare if two movies have same or similar genre value(s)

- Feature Selection: Which fields from your data will you use/exclude?

There are more than 30 columns in the combined dataset for IMDB. However, instead of using all of them, we've chosen a subset that are crucial for recommendation which are

1. Title
2. Genre
3. Synopsis (named plot and description in data)
4. Awards
5. Runtime
6. Country
7. IMDB Rating
8. IMDB Votes
9. Release Year
10. IsAdult : Integer dtype with 0 or 1

- Feature Creation: Which fields will be combined, or bucketed?

- Using a pre-trained **Bert** model from **HuggingFace**, synopsis will be converted into 768 different numeric columns to represent semantics for each movie title

- Use KMeans Clustering to group different genre combinations into 10 specific bins. This helps the recommendation because assuming there are 5 distinct genres, it can have 5! (or 5*4*3*2*1) number of different combinations at max. But by bucketing them into 10, it's much easier to get similar titles. How it's done is answered below

- Feature Transformation: What other transformations (such as one hot encoding, etc) will you apply to your data?
- Because genres column is also in string format, we will apply **One-Hot-Encoding** to it to keep track of the count of genres appearing for each distinct title. After that, KMeans Clustering is used to group different combinations of genres into 10 specific groups. This is so that when we compare two titles having action/adventure and action/adventure/thriller, we can consider them to have similar genres. Also **TfidfVectorizer** will be used on this genres as well. In addition to having genre group value, we create this to see how much this can help boost the performance later

- How will you balance your data set?
- No balancing is needed as we are creating a recommendation system. Since we are mainly using semantics of synopsis of each title and already filtered out those that are missing this value, we don't have to worry about this

- How will you split your dataset?
- Again, because this is a recommendation and no actual training is done, how to split the dataset is not our main concern. Instead, by creating a list of titles that appear both in IMDB and Netflix (by fuzzy match algorithm), we make sure that our recommended titles do not overlap with what Netflix already has in their system

Part 4

Model Training:

How will you train your model, what tools will you use?

- Will you be using SageMaker Jumpstart, built-in-algorithms, bring-your-own-script or bring-your-own container?
    - For the model we created, we used bring-you-own-script.

- Which algorithm will you be using?
    - To analyze the semantics of synopsis for each title, we used a pre-trained model called Bert from HuggingFace. By using this, plot(or synopsis) is now converted into 768 numeric features that a model can use to recommend. KMeans Clustering is also used to group different genre values into 10 specific bins by their similarity (so that action and adventure is considered quite similar)

- Which parameters will you be passing?
    - For K-Means Clustering, only number of clusters is provided as it is a unsupervised machine learning model. Because this is a recommendation system and not classification problem nor a regression problem, there are only a limited number of parameters we could use and in this project only that param was used.
    - Also for semantic analysis, since we are using a pre-trained Bert (base-uncased) model, the training has already been done and verified by community. So there are no parameters we could try here.

- Which instance size/count will you be using?
    - For Sagemaker instance, the standard instance version **ml.t3.medium** with 4GiB memory and 2 vCPU is enough for our project.

- How will you evaluate your model?
    - To get the final results and its performance evaluation, cosine similarity is used to compute the similarity between input and other titles in the dataset. The list of

recommended titles are for Netflix to decide if they are truly what they are interested in. So the only way we could evaluate the model is by checking the relevancy between an input title (with its metadata) and the recommended titles to it.

Part 5

Future Enhancements:

1. Scaling up the model.

Currently the Bert model (pre-trained, verified, and top-voted by HuggingFace community) we are using is only the base model and uncased. Given an input title and its synopsis (or any other features of string data type), this outputs to a vector of 768 numeric features. This also has the limit on the number of characters in a given sentence (or a paragraph). At the moment, there are some cases where the synopsis of a movie is longer than 512 characters, even with some pre-processing, so it had to be cut down and lose some information about it. Increasing the size of the model can produce bigger and more detailed semantics on the synopsis of a title. By doing so, a title can be represented more in depth which leads to a better recommendation. Additionally, since we don't have to suffer from losing data due to the limitation, this could boost the performance of the final model.

Transitioning to a bigger model we would most likely use a MABS to determine which model to keep on using to get the job done. Using a MABS system our model can be scaled in proportion to the accuracy rating of customer reviews. However, as a company, scaling out is considerable in order to introduce new models for certain regions and parallelize the load of new information that we bring to the model. New information would be brought in using real-time prediction on a delay. While the output of our models would be using a batch transformation to generate a list amongst the average daily user traffic.

2. Continuously updating dataset (for cleaner, better, and more populated features)

Some titles suffer from features, such as genres or synopsis, being missing. In addition to that, columns like IMDB Rating is something that constantly changes which can affect the recommendation, if only high ranking titles are being served to users. To address this problem, we could set up an automated script that keeps scraping the most recent data of a title and updating them regularly to the database. Also by adding more features such as Rotten Tomatoes' critics and audience score could provide a wider range of information and give them more options to filter/improve the results.

Not only this helps and provides users with the most recent information of movies but also offers more accurate results as the list of recommended titles could change based on new data. New data can be included from SageMaker Ground Truth to validate using sample size votes from sample polls or assign a point system of people votes that have higher weight of being correct. Using these tools could reduce or detect bias from IMDB or any other movie score acquisition we acquire resulting in better cleaned data. Lastly, data-quality baseline could be included to to automate the flow of input data to make sure we have the correct amount of features to make sure it is ready for analysis.

3. Dimensionality Reduction

Although **PCA** might be the first technique to consider here since it is famous for being fast to use and retains the overall variance of the dataset, however one of its challenges is that it does not retain non-linear variance.

One alternative method could be applied here to reduce dimensionality reduction is using the t-SNE technique. t-SNE is a **nonlinear dimensionality reduction** technique which is a great fit for our model as it embeds high dimensional data into lower dimensional data. The way it would help our model is by generating slightly different results each time on the same data set, focusing on retaining the structure of neighboring points. Another technique would be **UMAP** which is a non-linear dimensional reduction method. We could apply all three techniques

and compare the results of which one best increased the model performance. At the end we would apply the **Canary Traffic Shifting** technique to test a portion of our end points or the model performance, and once we decide which one performed the best through traffic fleets, our model would be ready for deployment after we have set up **Amazon CloudWatch alarms** for our endpoint.

Resources:

Tatenda. (2019, May 2). *Recommender Systems: Ethics and Confidentiality*. Medium.com. https://medium.com/@tatendambz/recommender-systems-ethics-and-confidentiality-717234379e54