Real-time, Personalized Movie Recommendations with Amazon Personalize

Siddhant Badola
dept. of Computer Science
Lakehead University
Thunder Bay, Canada
sbadola@lakeheadu.ca
ID-1154586

Harshil Vipulkumar Prajapati
dept. of Computer Science
Lakehead University
Thunder Bay, Canada
hprajap3@lakeheadu.ca
ID-1148896

Jaswanthmani Madisetty
dept. of Computer Science
Lakehead University
Thunder Bay, Canada
jmadiset@lakeheadu.ca
ID-1155228

Lakshmi Preethi Kamak
dept. of Computer Science
Lakehead University
Thunder Bay, Canada
lkamak@lakeheadu.ca
ID-1160111

***Abstract*—In today's world of the internet era, there is an explosion of information present on the web. Every day, we are bombarded with choices regarding several aspects of our digital lives. Be it choosing what to order to eat, what to shop for clothing, or even which media to consume for entertainment purposes. There are just too many choices to make in such a limited time. It is here that recommendation systems come into play. In this report, we propose a recommendation system, with movies as a specific use case. We try to build a cloud system that can seamlessly recommend movies to a user based on his preferences in real-time. We intend to use AWS for this project and Amazon Personalize will be the primary cloud service used for the project.**

***Index Terms*—AWS, Personalize, CLoud Computing, Recommendation System.**

I. Introduction

Due to the abundance of digital media, choosing the media to consume can be a daunting task, especially for entertainment purposes. There are many companies like Netflix, Amazon Prime, Hulu, etc. that provide digital media like movies and series to be watched by consumers. They have a huge library of movies and series and the user often feels confused as to what to watch next. It is here that such companies provide recommendations to the consumers as to what to watch next. Netflix seems to have the most sophisticated recommendation system to date. They also have prize money [1] for any developer or researcher who can improve their present recommendation system. The research in the area of recommendation systems has been going on for several decades now, but the interest still remains high because of the abundance of practical applications and the problem-rich domain. Recommendation systems are special types of expert systems in the sense that they combine the knowledge of the expert in a given domain (for the product type being recommended) with the user's preferences to filter the available information and provide the user with the most relevant information. Two primary paradigms of filtering are content-based and collaborative filtering. Although, most recommendation systems use a hybrid approach, which is a combination of these two approaches. While a content-based filter uses the user's past history to recommend new items to the user, a collaborative filter uses the preferences of other users with a similar taste to recommend certain movie titles.

A. The proposed method

In our project, we will use Amazon Personalize [2] which is their state-of-the-art system recommendation. Amazon Personalize is a fully managed machine learning service that goes beyond rigid static rule-based recommendation systems and trains, tunes, and deploys custom ML models to deliver highly customized recommendations to customers across industries such as retail and media, and entertainment.

B. Dataset used

MovieLens is a movie recommendation service that provides a machine learning data set of 100,000 ratings and 3,600 tag applications applied to 9,000 movies by 600 users [3]. It was collected around the years 1996 to 2018. The data set was collected from random users from various demography, who had to rate at least 20 movies. Each user is represented by an id only and all the data sets are available in CSV format. Ratings of each movie on a scale of 0 to 5 are stored in the rating data set along with timestamps. Each movie has defined a tag which is a user generator metadata, that contains a single word or short phrases about the movie. movie information. The movie information is also categorized by genre like Action, Adventure, Animation, Children's, Comedy, Crime, etc. Movie title information is also imported from the source data like IMDb and also the links are provided to the movies.

C. Training Analysis

The training model used in this recommendation is the User-Personalization recipe which is optimized for all personalized recommendations using automatic item exploration. With this model, we are capable of balancing the less interacted data items to be recommended more frequently or the more relevant recommendations based on the history of the user.

Parameter Tuning: For the algorithm - arn:aws:personalize:::algorithm/aws-user-personalization the following parameters were set: The performHPO variable was set to true executing CreateSolution and CreateSolutionVersion operations. The EVENT_TYPE data column with values were provided while created to train the solution model as click and watch as interaction data.

1) *Exploration weight*: The recommendations factor to include items with more exploration or more relevance. At zero, no exploration occurs and recommendations are based on current data (relevance) and the closer the value is to 1, the more exploration occurs.
2) *Exploration item age cut off*: The weightage for the scope of item exploration based on the interaction time frame. The number of items considered is proportional to the value. The value corresponds to the days.

II. METHODOLOGY

We will primarily be using the following technologies to create our movie recommendation system in AWS:-

- Amazon Personalize
- Amazon SageMaker
- AWS Identity and Access Management
- Amazon S3
- AWS SDK for Python

We will build, train, and deploy an Amazon Personalize recommendation model (a solution version) with the AWS Management Console using the AWS SDK for python. We will create an Amazon SageMaker notebook instance and attach the appropriate policies required for the SageMaker role. An Amazon SageMaker notebook instance is a fully managed machine learning (ML) instance that runs the Jupyter Notebook App. Our Amazon Personalize model will be trained on the above-mentioned MovieLens dataset which is curated by GroupLens Research. MovieLens will help us find movies we like. We will import and fetch the dataset. Then the dataset is prepared by splitting it based on movie ratings. We will create an Amazon S3 Bucket store for our interaction dataset. Amazon Simple Storage Service (Amazon S3) is an object storage service that offers scalability, data availability, security, and performance. We will use an Amazon S3 bucket to stage the interactions dataset and store the model generated by Amazon Personalize during model training.Let us look into each step in more detail. Amazon personalize is a cloud service in AWS that is used to make machine learning models run

on specific datasets to provide users with personal recommendations, without any specific knowledge in Machine Learning for developers. In other words, it allows organizations to subscribe to machine learning recommendation models without creating their own models.Let us describe the workflow in detail now. After we have created our AWS account, the first thing to do is to create an Amazon SageMaker Notebook. We will use this notebook as a jupyter notebook to write our code. We primarily use Amazon SageMaker to build machine learning models as it is an IDE specifically designed for such tasks. So we can use SageMaker to prepare the dataset, build and train the model and finally deploy it on the cloud. After a few minutes, the SageMaker will create a notebook instance which is a preconfigured Jupyter notebook for making Machine Learning models. We will then attach an IAM role to this notebook. This IAM role will contain certain "policies" that are a set of predefined rules as to what we are allowed to do with the notebook instance. We will give full access to this IAM role for S3 bucket storage access as well as Amazon Personalize. After accessing the notebook our next step is to prepare the dataset. We have chosen an "Interaction" dataset which contains metadata about users and how they interact with items. The dataset we have used has been provided by MovieLens[1]. After importing the required libraries, we fetch the dataset and prepare it for the model we are going to create after this. We have decided to remove any movie with rating below 2 to remove negative feedback. We have also marked movies above 2 as "clicked" and above 3 as both "clicked" and "watched". We then convert this dataframe into a csv to be stored in a S3 bucket. We also need to create the schema for the dataset before sending it to our S3 bucket and making it live. In case the S3 bucket is not already preconfigured and live, we will need to create the bucket. We will also need to create a policy that Amazon Personalize can read the S3 bucket. After properly configuring the bucket, we can import the data into it. Our next step is to configure a solution which is AWS term for a machine learning model. We will specify the solution by configuring its parameters and choosing a "recipe" which is AWS equivalent of an algorithm being used to train the model. We have used the "User Personalize" recipe for training the model. After setting up all the parameters we finally start training the solution. We will then create a "campaign" which means that we create an API for our solution which goes live and we can get recommendations from this API. For our demo, this API will be connected to a User Interface element like a custom website.

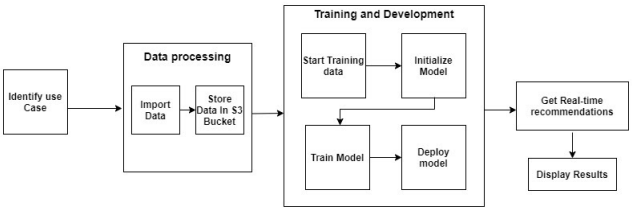

Fig 1 : Overview of the Recommendation Model

A. *Evaluation of the Metrics:*

We evaluated the performance of the model using the following metrics collected offline and allowed us to view the effects of modifying a solution's hyperparameters and compare results of different models. The algorithm splits the dataset into a 90:10 ratio as a training and testing set based on oldest and new interaction data for each user.

- *Training time* : 30 mins
  "coverage": 0.27,
  "mean_reciprocal_rank_at_25": 0.0379,
  "normalized_discounted_cumulative_gain_at_5": 0.0405,
  "normalized_discounted_cumulative_gain_at_10": 0.0513,
  "normalized_discounted_cumulative_gain_at_25": 0.0828,
  "precision_at_5": 0.0136,
  "precision_at_10": 0.0102,
  "precision_at_25": 0.0091,
  "average_rewards_at_k": 121.214

- *Coverage*: fraction of unique recommendations out of the total number of unique items in Interactions and Items datasets. Higher coverage is favourable
- *mean reciprocal rank at 25*: The relevance of the recommendation with the top 25 ranked recommendation
- *normalized discounted cumulative gain at K* : It uses a weighting factor of 1/log(1 + posi-

tion) based on various sample sizes to assess relevance.

- *Average_rewards_at_k , precision at K* : The relevance of the recommendation with various sample sizes and average.

III. EXPERIMENTAL RESULTS

History of User : User 580 was chosen as a case study. Few of the major patterns found in the user's interactions were low preference of Drama movies based on consistent low ratings. The user had high preference for Action & Thriller movies based on high ratings on his recent interaction with both click and watch events.

The Exploration weight was varied from 0.1 to 0.9 . We focused on user 580 and learned about the user's viewing patterns, such as the movies the user has clicked or watched. We built an exploration weight that ranges from 0 to 1 dependent on the user's habit. The user receives movie recommendations based on the exploration weights. If the user's exploration weight is 0.1, the user will receive a recommendation based on the movies that he or she has rated , clicked and watched. Here, the user gets Action movies as top recommendation this means he/she likes and watches Action genre more. If the exploration weight is 0.6-0.7, the user is recommended movies which he/she likes to see along with new genre movies which he/she has not yet explored. Here, along with Action genre, Thriller and Drama genre are recommended which he/she has not yet explored. For exploration weight 0.9, the user is recommended movies which he/she has not yet clicked or watched. This allows the user to explore new movies based on different genres which the user has not yet watched. Here, the user_id 580 is recommended with genres like Comedy, Romantic for exploration weight 0.9.

EXPERIMENTAL RESULTS

UserId - 580

Exploration weight : 0.1 ; Training Time: 10min1s

Recommendation Movie	Genres
28 Days Later (2002)	Action|Horror|Sci-Fi
Django Unchained (2012)	Action|Drama|Western
Lucky Number Slevin (2006)	Crime|Drama|Mystery
40-Year-Old Virgin, The (2005)	Comedy|Romance
Sin City (2005)	Action|Crime|Film-Noir|Mystery|Thriller

Fig 2: results for 0.1 weight

EXPERIMENTAL RESULTS

UserId - 580

Exploration weight : 0.7 ; Training Time: 10min 25s

Recommendation Movie	Genres
Cabin in the Woods, The (2012)	Comedy|Horror|Sci-Fi|Thriller
History of Violence, A (2005)	Action|Crime|Drama|Thriller
Brazil (1985)	Fantasy|Sci-Fi
Syriana (2005)	Drama|Thriller
8 Mile (2002)	Drama

Fig 3: results for 0.7 weight

EXPERIMENTAL RESULTS

UserId - 580

Exploration weight : 0.9 ; Training Time: 09 min 54s

Recommendation Movie	Genres
28 Days Later (2002)	Action|Horror|Sci-Fi
Bridget Jones's Diary (2001)	Comedy|Drama|Romance
Kate & Leopold (2001)	Comedy|Romance
Zombieland (2009)	Action|Comedy|Horror
28 weeks Later (2007)	Horror|Sci-Fi|Thriller

Fig 4: results for 0.9 weight

IV. CHALLENGES FACED

- Long training time- Transformed tightly coupled Code into loosely coupled modular code

to handle issues faced in time-out, breaks in long training time
- Data Analysis - Unable to use AWS Quicksight due to limited access to student accounts; Discussing and working on alternative solutions.
- UI Portal - Checking feasibility of UI portal using AWS bucket to enable the API Call through External UI to enable accessibility to consumers
- IAM Management
- Performance Tuning

V. Future Scope

The advantage of Amazon Personalize is the zero-cost automatic update of the latest model (solution version) is done every two hours in the back-end to include real-time data without creating a new model. The latest item information update adjusts the exploration and future recommendations based on implicit user feedback. This can be set using the training mode parameter set to FULL. Impressions data of lists of items that were visible to a user when they interacted with clicked, watched, purchased can also be integrated into the decision-making process. The hyperparameter optimization (HPO) can done using the following parameters :

1) Recency_mask : Adds more weightage for recent events to gather the latest popularity trends in the Interactions dataset.
2) Hidden_dimension : Increase weightage for the number of hidden variables to generate ranking scores.
3) bptt- Back-propagation through time adds weightage in the recurrent neural network-based algorithms based on several clicks leading to delayed reward.

VI. Conclusion

The expected outcome of this project will be a set of movie recommendations available for a chosen user with their search history and movies they watched in the past using Amazon Personalize. Users are provided with a movie name and a link to the movie source on the console. The service will be available on-demand, making it a real-time recommendation system. Future works could include the machine learning model adapting from the user's choices based on the cumulative previous choices on movie recommendation.

References

[1] "Wikipedia Netflix Prize",[Online],Available:https://en.m.wikipedia.org/wiki/
[2] "Real0time Personalization with AWS Personalize",[Online],Available:https://aws.amazon.com/personalize/
[3] "Group Lens Research",[Online],Available: https://grouplens.org/datasets/movielens/