2020 ECE SUMMER HONORS RESEARCH PROGRAM -

Industrial Recommender Systems in Media & Entertainment

- *By Siddharth Mandgi*
  under Prof Hong Man

WEEK-1

Table of Contents

Introduction

The research topic I have chosen for the *2020 ECE Summer Honors Research Program* is ***Industrial Recommender Systems in Media & Entertainment***.

For my research, I will be providing a weekly analysis of my findings as well as coding support on my GitHub Repository as I keep developing this project. I will also be providing a final report and a presentation at the end to summarize this research.

This research is a part of the Honors Research Program conducted by *Stevens Institute of Technology* and I would like to express my sincere gratitude to ***Prof. Hong Man*** for being my mentor and guiding me through this project.

Recommender Systems simply put, are AI algorithms that utilize Big Data to suggest additional products to consumers based on a variety of features. These recommendations can be based on factors such as past purchases, demographic info, their search history, time spent reviewing the product or a like, dislike or a comment left behind by these consumers.

The idea of Recommender Systems is that if you can narrow down the pool of selection options for your customers to a ***few meaningful and relevant choices***, they are more likely to make a purchase now, as well as come back for more down the road.

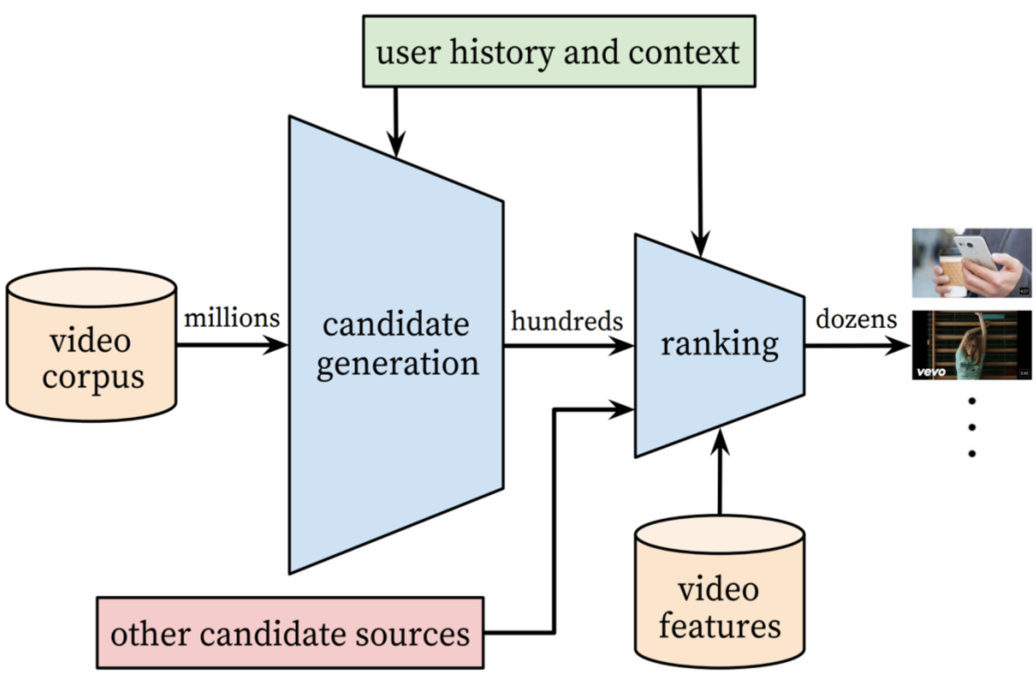
Through the research topic, ***Industrial Recommender Systems in Media & Entertainment,*** I hope to explore about how organizations such as Spotify, Netflix and YouTube leverage Recommender Systems to enjoy a high user share, as well as learn about the essence of Neural Networks and Deep Learning in optimizing user recommendations, making them more relevant and more meaningful.

In this first week of our Research Program we will discuss about how different streaming companies use Recommender Systems to generate precise recommendations for millions of users based on millions of items (Songs, Videos, and Music). Further we will discuss about a blueprint for translating our Algorithms into code and a basic setup needed for the same. We will also be discussing about a few research papers, datasets and source codes that deal with Recommender Systems and lastly, we will be ending this week's report by a brief summary of what we have achieved and what we can expect next week.

Case Studies for Recommender Systems

YouTube

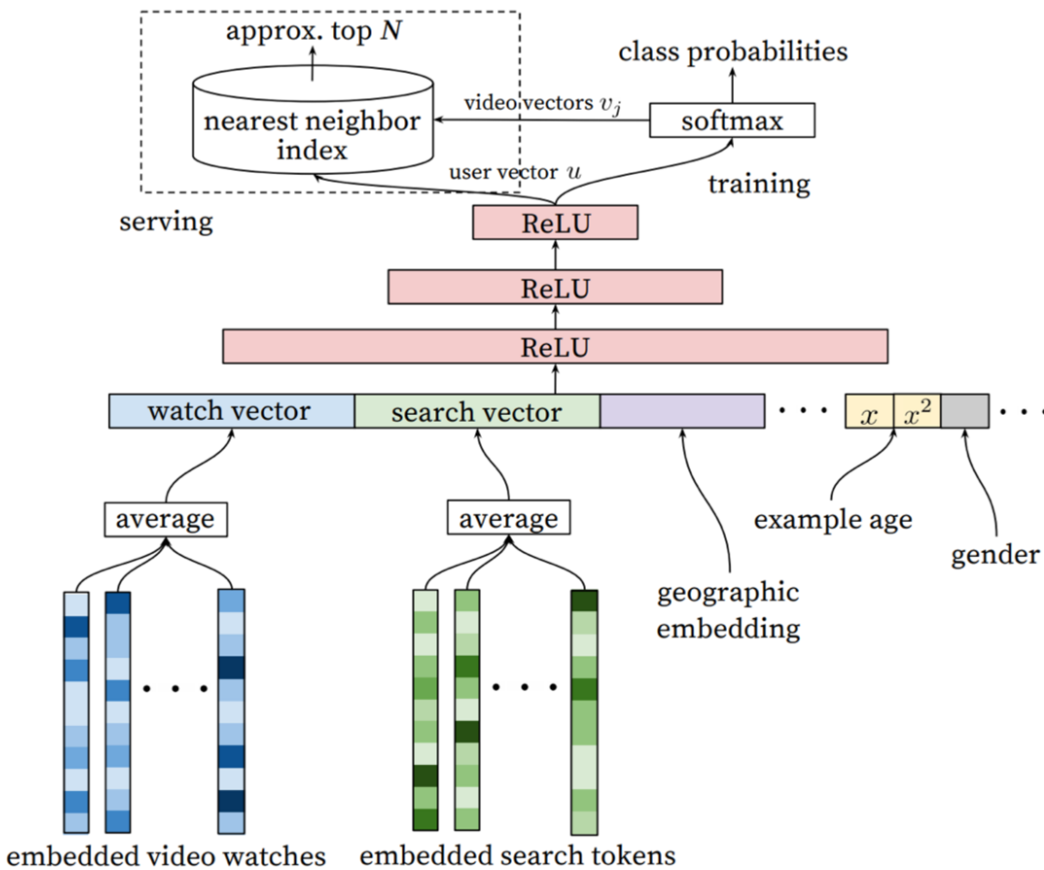
- From the research paper published by Google - Deep Neural Networks for YouTube Recommendations [1], YouTube has a ***Two Stage Process*** for Recommending videos to its users. The First Stage is Candidate Generation and the Second Stage is Ranking. Candidate Generation Algorithm takes millions of videos available on the platform and filters out the videos that the user may like, and the Ranking Algorithm sorts the videos selected by the former algorithm and sorts them in order of relevance.



- For ***Candidate Generation Algorithm*** some implicit features the YouTube uses are as follows: -

  - Watch History: Videos Watched Recently
  - Search History: Queries Typed in Search Bar
  - Location: United States, South Korea or India
  - Device Type: Mac, Mobile, Windows
  - Age
  - Video Freshness: Age of the Video

- These user features are fed into the neural network in terms of vectors which then generates a set of recommended videos.



- For example, the watch history of each video is taken into a fixed vector and these vectors are then averaged to get a final vector for watch history. Similar is the case for all other features.

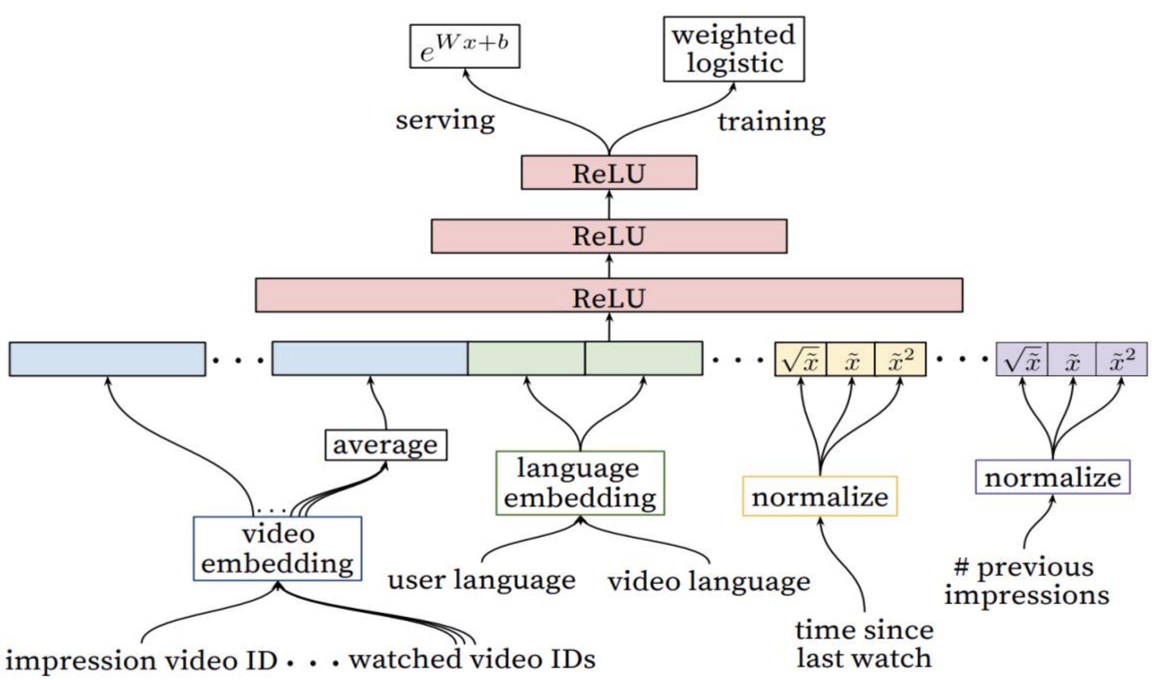
- These averaged final vectors are concatenated and finally given as input to a neural network.

- For the output YouTube does not consider every single output generated since SoftMax in the above network would provide a probability of whether the video was watched completely or not which would imply having an output for every single video, a count which could be in billions. Instead YouTube uses an Algorithm called Candidate Sampling, where only a subset of the entire output is considered by a method called sampling.

- For ***Ranking,*** features used by YouTube Include
    - What videos were watched / searched by the user?
    - How many videos has the user watched from this channel?
    - When was the last time the user watched a video on this topic?

- Ranking can be computationally expensive hence we require Candidate Sampling to make the process simpler.

- In case of YouTube, score is directly proportional to watch time of the video which computed by weighted logistic regression where the 'weights' are the watch times.



Spotify

There are three recommendation models at work on Spotify:

- Collaborative Filtering: This model uses “nearest neighbors” to make predictions about what other users might enjoy. This is similar to Netflix’s model, but Spotify’s engine is not powered by star ratings. Spotify must use implicit feedback signals like stream counts to infer what we like. In fact, Netflix has moved to this approach as it is more reliable than using explicit feedback. If you want to understand people, listen to what they do, not what they say. Each user on Spotify has their own taste profile, shaped by what they listen to, when they listen to it, how often, and so on.

  Discover Weekly has been a phenomenal success for Spotify; over 5 billion tracks were streamed through these playlists in the first year after launch. The idea is as simple as the execution is complex: Find 30 songs that the listener would probably like but has not listened to (on Spotify) yet. The binary system uses the labels 1 (streamed) and 0 (never streamed) for songs, with the latter then organized based on the likelihood the user will engage with them. The list is created automatically and is available every Monday.

- Natural Language Processing: - Similar to Google’s NLP algorithms, Spotify identifies the co-location of individual terms and uses this to predict the meaning of phrases. This insight then feeds into Spotify’s vast web of entities, with highly rewarding results. Their tests have found that recommending songs based on these common adjectives surfaces new links that were previously unseen.

- Audio Models: - The first two models we have assessed have delivered Spotify’s recent success; this third model holds the key to its future success. If the song is from a new artist with a small following, it will generate very few data points. The artist will likely tag the song with category and genre attributes, but this would be sufficient only for an archaic recommender system. In addition to this, Spotify uses the kind of neural network that is employed by search engines to understand the contents of images. These networks process raw audio to produce a range of characteristics, including key, tempo, and even loudness.

Netflix

Netflix estimates the likelihood using ***Collaborative Filtering*** that the user will watch a particular title in their catalog based on a number of factors including:

- The user interactions with the platform (such as your viewing history and how you rated other titles),
- Other users with similar tastes and preferences on our service, and
- Information about the titles, such as their genre, categories, actors, release year, etc.

In addition to knowing what you have watched on Netflix, to best personalize the recommendations we also look at things like:

- the time of day you watch,
- the devices you are watching Netflix on, and
- how long you watch.
- which titles appear in the row, and
- the ranking of those titles.

Netflix adds a lot of personalization in their recommender systems to produce top notch results which have completely disrupted the way TV and media industry operates.

Data Collection & Datasets

Ways to collect data for this project: -

- Web Scraping, Artist and Album info, from the Internet.
- Using APIs from apps like Last FM, Spotify & iTunes.
- Using Existing Datasets.

Datasets: -

- YouTube 3M
- Yahoo Music Datasets
- From Websites like Kaggle, UCI

Coding Environment

For this project, our coding environment includes: -

- MacOS (XNU kernel)
- Python (Version - 3.8)
- IDE - PyCharm & Jupyter Notebook
- Cloud (if required) - Heroku or AWS

Setting up or environment: -

- Download the latest version of **Python** compatible with your operating system from https://www.python.org/downloads/
- Install **Jupyter Notebook** by typing the command *pip install jupyter notebook*
- We might also require PyCharm for our project which can installed from https://www.jetbrains.com/pycharm/download/#section=mac

Summary

In this week's report, we have Introduced our Research Project and discussed case studies of Recommender Systems in organizations such as Spotify, YouTube and Netflix. We also discussed about the different ways of collecting data using APIs along with the coding environment required for this project and ways to setting it up.