# LEARNING TO RANK ALGORITHM

## BY AKSHIT KALANTRI

# *What is Learning to Rank?*

Today, the online world is driven by search. Google has become synonymous to looking up information online, Amazon and other e-commerce marketplaces have sophisticated search results catered to the user’s browsing and purchase history and even airline booking services have their algorithms offer the best deals with both prices and travel time considered. Now this magic behind the scenes is something that is a bit more unique than traditional ML, it is an algorithm termed ‘Learning to Rank’ (LTR). Learning to Rank is a smart algorithm that ranks a list based on the gradient or the degree of its suitability. To do so, it does not rely on a numerical designation for each item in the list, but instead just to what degree the item satisfies the criteria. It is a unique approach to help tackle what can be computationally intensive rankings and accurately predict what the query demands of it.

(Types of LTR- pairwise, listwise...)

# Learning to Rank versus Traditional ML

Learning to Rank is not too dissimilar to the traditional Machine Learning approach but has a few key changes. Both the approaches rely on a large amount of data and learning to improve and also aim to cut down computational load. However, unlike an ML approach, LTR does not seek to numerically rank each item on the list, making it far more efficient in ordering long lists of data. It only aims to rank the items by their relevance to the search, hence giving the best outputs for use cases such as search engines.

# Current Practical Applications of LTR

LTR has become widely popular in the recent years and is now deployed for various applications in the industry.

* The primary target is **search engines**- for e-commerce, web searches and recommendation systems.
* An interesting and useful area of application is **translation**, where the ranking is used to order the most apt translations for a given phrase.
* Another popular use case is the **newsfeed on social media** such as Facebook, by showing the most relevant articles and posts at the top and engaging the user more.
* A practical application outside of the digital world is LTR being used in **Amazon delivery services** to pinpoint the location of a delivery address by using data from previous deliveries, helping a future prospect of drone-based delivery.

(More in-depth about 1-2 applications if needed)

# Dataset Description

In this demo, the dataset being used is the MovieLens 100K, which is a standard benchmark dataset containing 1,00,000 ratings of 1700 movies from 1000 unique users. Each rating ranges from 1-5 and is the parameter that will be used as a label to help determine the ranking.

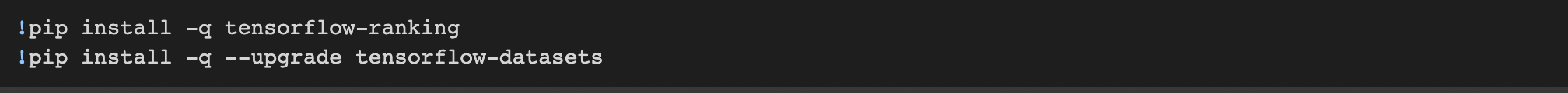
# Dataset Preprocessing

Since the dataset being used here is a standardized tensorflow dataset, it is ready to be used. However, if a custom dataset needs to be used, these steps can help prepare it for the LTR algorithm.

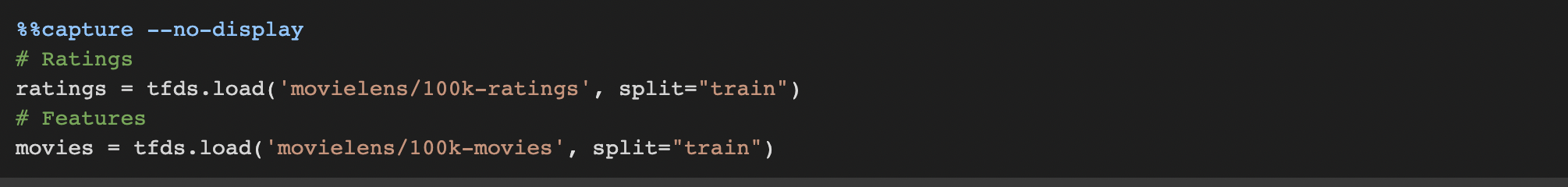
(TBC)

# Code Breakdown

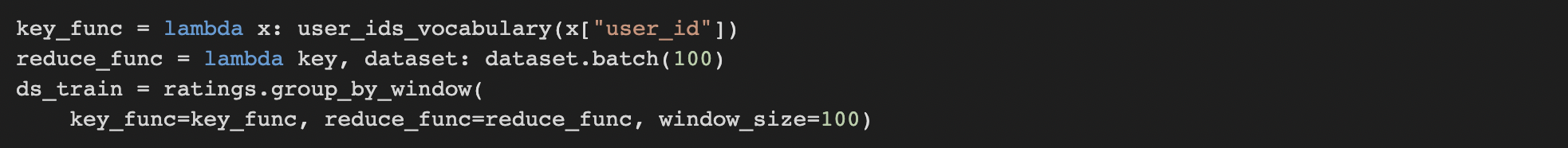
* First, the tensorflow ranking and tensorflow datasets packages need to be installed.



* The packages are then imported, along with the standard typing functions.  
    
  Shape

  Description automatically generated with low confidence
* Then the data from the MovieLens 100K standard datasets is read and stored into two datasets- ratings and movies.  
    
  
* The features needed are then selected as user-id, movie-title and rating. Here, the rating is used as the label as that will be the feature used to rank the lists.  
    
  A picture containing shape

  Description automatically generated
* Once the features are extracted, they need to be converted into integer indices to help the algorithm process them and ultimately rank them. This is done by building a vocabulary for both user-ids and movie-titles, as a list of the unique values in each are needed to pair them using the labels.  
    
  Text

  Description automatically generated
* The unique user-id vocabulary can then be used to group the data into lists.  
    
  
* For example, the items in the first list are shown below.  
    
  Text

  Description automatically generated
* The features and labels are then batched to aid the data processing. A batch refers to the number of lists fed through the model at once.  
    
  Text

  Description automatically generated
* These batches are of size 32 for both user-ids and movie-titles, in lists of 100 except for when 100 items are not available in the list.  
    
  Text

  Description automatically generated
* A standard Keras model is then to be constructed, along with the call method. A dot product of the user and movie integers indices is returned for each iteration, to determine the gradient of ranking.  
    
  Text

  Description automatically generated
* The model is then finished, using SoftMax loss and standard LTR evaluation metrics such as NDCG (Net Discounted Cumulative Gain) and MRR (Mean Reciprocal Rank).  
    
  Text

  Description automatically generated
* The model is then fitted over 3 epochs and the metrics are obtained.  
    
  Text

  Description automatically generated
* The model is finally tested for predictions and attempts to return movie recommendations based on the ranking which it learns from both the user’s ratings and other similar users’ ratings.  
    
  Text

  Description automatically generated