Project Title:

Members: Zhengqi Dong,

# Problem Description:

“What movie should I watch tonight?” Have you ever heard your friends or family members ask you this question? As for me — yes, and more than once. The need to build robust movie recommendation systems is extremely important given the huge demand for personalized content of modern consumers. An example of recommendation system is such as this:

* User A watches Game of Thrones and Breaking Bad.
* User B does search on Game of Thrones, then the system suggests Breaking Bad from data collected about user A.

# Objective:

* Predict the rating that a user would give to a movie that he hasn’t yet rated. The movie for which the predicted rating is highest(or top\_n highest) will be recommended to the user. In the simplest formulations, these systems are trained to estimate some score  y\_ij , such as an estimated rating, given a user  u\_i  and movie m\_j . Given such a model, then for any given user, we could retrieve the set of objects with the largest scores  y\_ij , which could then be recommended to the customer.
* Minimize the difference between predicted and actual rating (RMSE and MAPE)

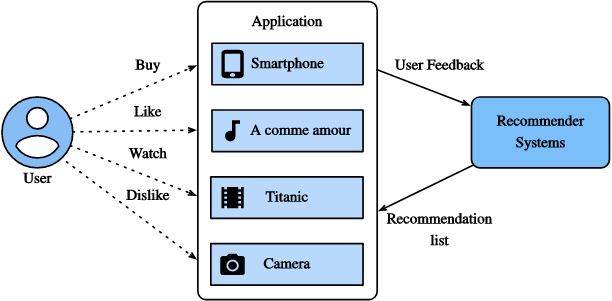


Fig. 16.1.1 Illustration of the Recommendation Process, <https://d2l.ai/chapter_recommender-systems/recsys-intro.html>

# Dataset description

We current have two alternative dataset for building this project:

* The **MovieLens Dataset**: (<https://grouplens.org/datasets/movielens/>)
  + MovieLens is a non-commercial web-based movie recommender system. It is created in 1997 and run by GroupLens, a research lab at the University of Minnesota, in order to gather movie rating data for research purposes. MovieLens data has been critical for several research studies including personalized recommendation and social psychology. GroupLens Research has collected and made available rating data sets from the MovieLens web site (http://movielens.org). The data sets were collected over various periods of time, depending on the size of the set.
    - MovieLens 25M Dataset: dataset released by 12/2019:This dataset contains 25M movie rating and 1M tag applications applied to 62K movies created by 162K users from 01/09/1995 to 11/21/2019. The data are contained in the 6 files genome-scores.csv, genome-tags.csv, links.csv, movies.csv, ratings.csv and tags.csv.
    - MovieLens Latest Datasets: This dataset has two version: 1) ml\_latest\_small.zip dataset released by 09/2018: This dataset contains over 10K rating and 3.6K tag application acros 9.7K movies created by 610 users from 03/29/1996 to 09/24/2018, and the data are contained in the files links.csv, movies.csv, ratings.csv and tags.csv. 2) ml\_latest.zip dataset released by 09/2018: This dataset contains over 27M rating and 1.1M tag application acros 58K movies created by 280K users from 03/29/1996 to 09/26/2018, and the data are contained in the 6 files genome-scores.csv, genome-tags.csv, links.csv, movies.csv, rangs.csv and tags.csv.
    - MovieLens 1M Dataset: MovieLens 1M movie ratings. Stable benchmark dataset. 1 million ratings from 6000 users on 4000 movies. Released 2/2003.
  + Netflix-Movie recommendation dataset (<https://www.kaggle.com/netflix-inc/netflix-prize-data>)
    - This project aims to build a movie recommendation mechanism within Netflix. The dataset I used here comes directly from Netflix. It consists of 4 text data files, each file contains over 20M rows, i.e. over 4K movies and 400K customers. All together over 17K movies and 500K+ customers!

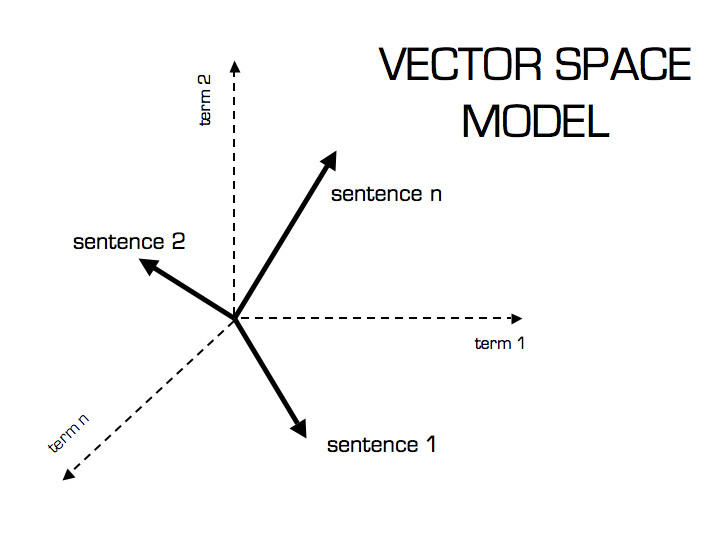
# Other Teminology:

* Explicit Feedback and Implicit Feedback
  + Explicit feedback: start rating(e.g. IMDB), thumbs-up and thumbs-down(e.g. YouTube), likes/heart
  + Implicit feedback is obtained by observing user’s behavior, including purchase history, browsing history, watches list, and even mouse movement. For example, a user that purchased many books by the same author probably indicates the user likes that author, but we only can guess their preference, and the true motives is never known.

Relatively speaking, implicit feedback is often readily available since it is mainly concerned with modeling implicit behavior such as user clicks.

# Related word

1. Content-Based Filtering:
   1. The Content-Based Recommender solely relies on the similarity of the items being recommended. For instance, if you like an item, then you will also like a “similar” item. It generally works well when it’s easy to determine the context/properties of each item.



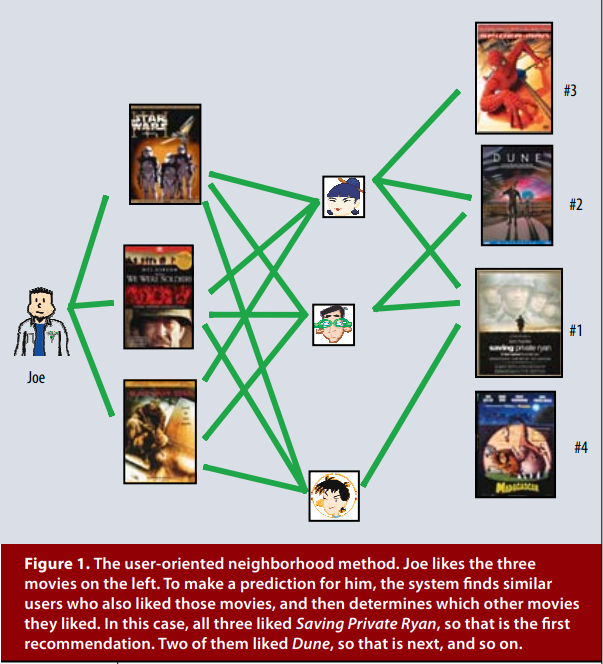
1. Collaborative Filtering/Recommendations:

The collaborative filtering(CF) [[Goldberg et al., 1992]](https://d2l.ai/chapter_references/zreferences.html#goldberg-nichols-oki-ea-1992) is based on the similarity between the preference of users likes and not the content of the product. In a broad sense, it is the process of filtering for information or patterns using techniques involving collaboration among multiple users, agents, and data sources. For instance, if user A likes movies 1, 2, 3 and user B likes movies 2,3,4, then they have similar interests and A should like movie 4 and B should like movie 1.

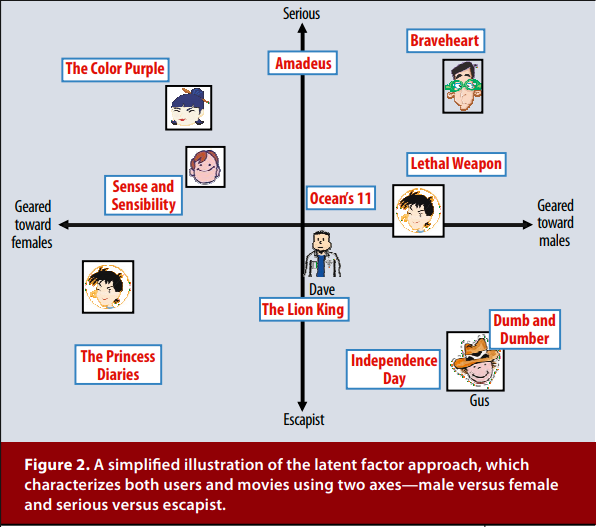
More specifically, the CF techniques can be categorized into: 1) memory-based CF, 2)model-based CF, and 3\_ Hybrid CF  [[Su & Khoshgoftaar, 2009]](https://d2l.ai/chapter_references/zreferences.html#su-khoshgoftaar-2009).

The primary area of collaborative filtering are two: Neighborhood methods, and latent factor model

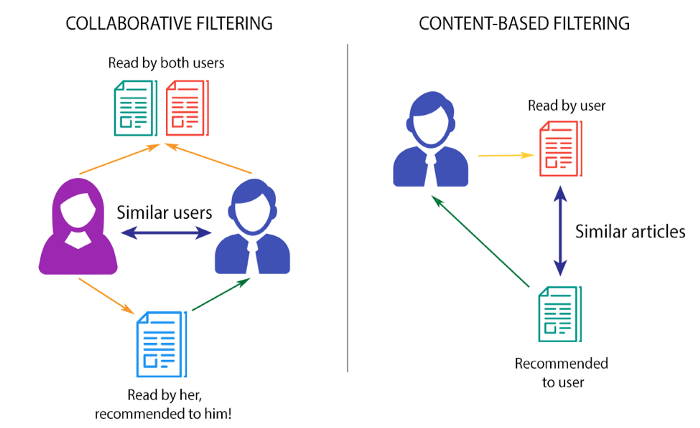
1. memory-based CF: Representative memory-based CF techniques are nearest neighbor-based CF such as user-based CF and item-based CF
2. model-based CF, and
3. Hybrid CF:
4. User-based: measure the similarity between target users and other users
5. Item-based: measure the similarity between the items that target users rates/ interacts with and other items
6. User-oriented Neighborhood methods: computing the relationships between items, or between users. The item-oriented approach evaluates a user’s preference for an item based on ratings of “neighboring” items by the same user. A product’s neighbors are other products that tend to get similar ratings when rated by the same user. For example, consider the movie Saving Private Ryan. Its neighbors might include war movies, Spielberg movies, and Tom Hanks movies, among others. To predict a particular user’s rating for Saving Private Ryan, we would look for the movie’s nearest neighbors that this user actually rated. As Figure 1 illustrates, the user-oriented approach identifies like-minded users who can complement each other’s ratings.



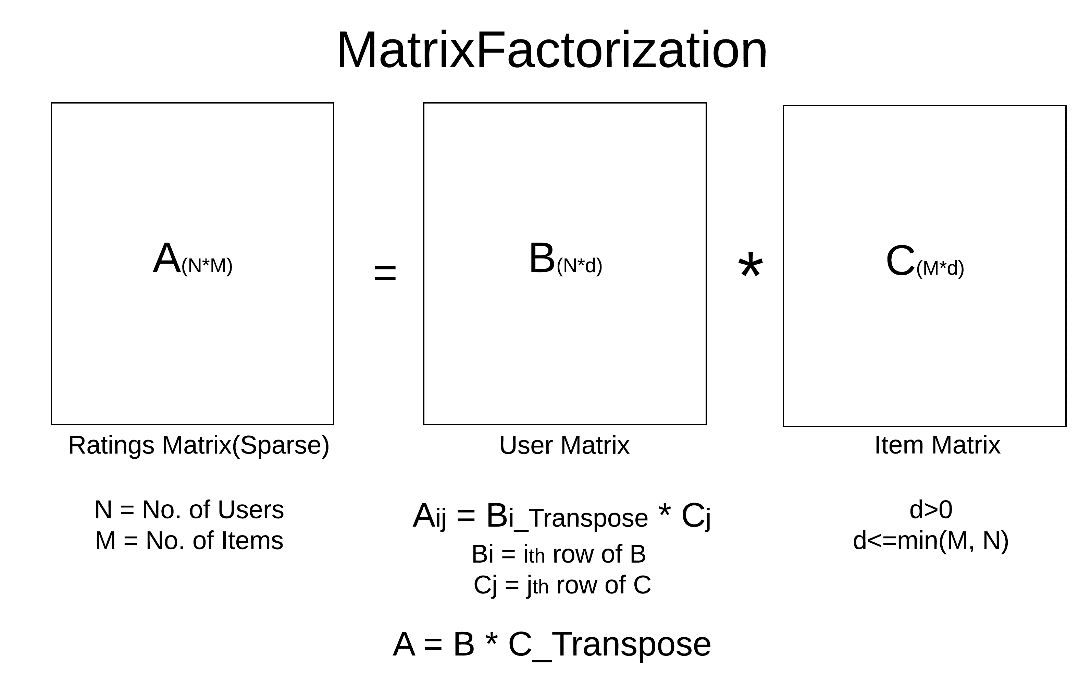
1. Latent factor models: Latent factor models are an alternative approach that tries to explain the ratings by characterizing both items and users on, say, 20 to 100 factors inferred from the ratings patterns. For movies, the discovered factors might measure obvious dimensions such as comedy versus drama, amount of action, or orientation to children; less well-defined dimensions such as depth of character development or quirkiness; or completely uninterpretable dimensions. For users, each factor measures how much the user likes movies that score high on the corresponding movie factor. Figure 2 illustrates this idea for a simplified example in two dimensions. Consider two hypothetical dimensions characterized as female- versus male-oriented and serious versus escapist. The figure shows where several well-known movies and a few fictitious users might fall on these two dimensions. For this model, a user’s predicted rating for a movie, relative to the movie’s average rating, would equal the dot product of the movie’s and user’s locations on the graph. For example, we would expect Gus to love Dumb and Dumber, to hate The Color Purple, and to rate Braveheart about average. Note that some movies—for example, Ocean’s 11—and users—for example, Dave—would be characterized as fairly neutral on these two dimensions.



* + Drawback:
    - Doesn’t scale well to massive dataset, especially for real-time recommendations based on user behavior similarities, which can take a lot of comptations.
    - Don’t work well when the new users or items don’t have any rating entered system



1. Matrix Factorization:
   * Matrix factorization is a class of collaborative filtering models. Specifically, the model factorizes the user-item interaction matrix (e.g., rating matrix) into the product of two lower-rank matrices, capturing the low-rank structure of the user-item interactions.



Where matrix A is a User-Iterm matrix, each cell is the rating score given the user and item. The matrix B and C are the factor of matrix A, and now Aij is a product of Bi\_Transpose\*Cj.

* + Reference:
  + MATRIX FACTORIZATION TECHNIQUES FOR RECOMMENDER SYSTEMS, <https://datajobs.com/data-science-repo/Recommender-Systems-%5BNetflix%5D.pdf>

1. Deep Learning/Neural Network:
   1. It used the embedded vector space an
   2. Reference:
      1. Deep Neural Networks for YouTube Recommendations, <https://static.googleusercontent.com/media/research.google.com/zh-CN//pubs/archive/45530.pdf>

# Challenges:

* Scale: many existing recommendation algorithms proven to work well on small dataset, but fail to operate on large scale system, e.g. YouTube, Netflix, Amazon, Spotify, IMDB, Google, just to name a few.
* Freshness: For a newly published video there is not too much rating, or for a new user that doesn't have too many liked/favorite movies, the recommendation system has a hard time to make a responsive and accurate recommendation for the latest actions taken by the user.
* Noise: There are many external factors that might influence the satisfaction of a recommended movie, and the level of satisfaction can also be changed rapidly, so it’s hard to obtain the ground truth value of user satisfaction.
* Only observe censored feedback. Users preferentially rate movies that they feel strongly about: you might notice that items receive many 5 and 1 star ratings but that there are conspicuously few 3-star ratings. Moreover, current purchase habits are often a result of the recommendation algorithm currently in place, but learning algorithms do not always take this detail into account. Thus it is possible for feedback loops to form where a recommender system preferentially pushes an item that is then taken to be better (due to greater purchases) and in turn is recommended even more frequently. <https://d2l.ai/chapter_introduction/index.html#fig-deeplearning-amazon>

# Tutorial

* The 4 Recommendation Engines That Can Predict Your Movie Tastes, <https://medium.com/@james_aka_yale/the-4-recommendation-engines-that-can-predict-your-movie-tastes-bbec857b8223>
* How to Build a Recommender System(RS), <https://medium.com/datadriveninvestor/how-to-built-a-recommender-system-rs-616c988d64b2>
* Introduction to Recommender System. Part 1 (Collaborative Filtering, Singular Value Decomposition), <https://hackernoon.com/introduction-to-recommender-system-part-1-collaborative-filtering-singular-value-decomposition-44c9659c5e75>