



Problem Statement

Let's get acquainted with the task of building a recommendation system.

We'll cover the following ^

- Introduction
- Problem statement
- Visualizing the problem
- Scope of the problem
- Problem formulation
 - Types of user feedback

Introduction#

Recommendation systems are used by most of the platforms we use daily.

For example:

1. The Amazon homepage recommends personalized products that we might be interested in.
2. The Pinterest feed is full of pins that we might like based on trends and our historical browsing.
3. Netflix shows movie recommendations based on our taste, trending movies, etc.

In this chapter, we will go over the Netflix movie recommendation problem, but a similar technique can be applied to nearly all other recommendation systems. ☀

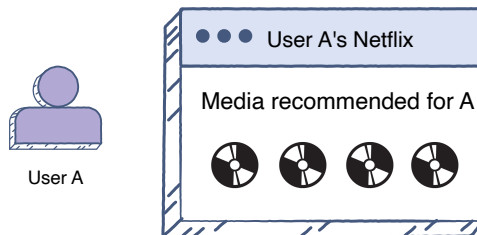


Problem statement#

The interviewer has asked you to **display media (movie/show) recommendations for a Netflix user**. Your task is to make recommendations in such a manner that the chance of the user watching them is maximized.

Display media recommendations (show/movie) for user A

How to recommend content such that the chance of a user watching recommended content is maximized




Visualizing the problem#

The prime factor that led to Netflix's success was its recommendation system. The algorithm does a great job of bringing the right kind of content to the right users. Unlike Netflix's recommendation system, a simple recommendation system would have simply recommended the trending movies/shows with little regard for the particular user's preferences. At most, it would look at the viewer's past watches and recommend movies/shows of the same genre.

Another key aspect of Netflix's approach is that they have found ways to recommend content that seemed different from the user's regular choices. However, Netflix's recommendations are not based on wild guesses, but they are based on *other users' watch histories*, these users share some common



patterns with the concerned user. This way, customers got to discover new content that they wouldn't have found otherwise.

 80% of the shows watched on Netflix are driven by its recommendations, as opposed to someone searching for a particular show and watching it.

The task at hand is to create such a recommendation system that keeps the viewers hooked and introduces them to varied content that expands their horizons.


Scope of the problem#

Now that you know the problem at hand, let's define the scope of the problem:

1. The total number of subscribers on the platform as of 2019 is 163.5 million.
2. There are 53 million international daily active users.

Hence, you have to build a system for a large number of users who require good recommendations on a daily basis.

One common way to set up the recommendation system in the machine learning world is to pose it as a classification problem with the aim to predict the probability of user engagement with the content. So, the problem statement would be:

“Given a user and context (time, location, and season), predict the probability of engagement for each movie and order movies using that score.” 



Problem formulation#

We established that you will predict the probability of engagement for each movie/show and then order/rank movies based on that score. Moreover, since our main focus is on getting the users to watch most of the recommendations, the recommendation system would be based on implicit feedback (having binary values: the user has watched movie/show or not watched).

Let's see why we have opted for ranking movies using "implicit feedback" as our probability predictor instead of using "explicit feedback" to predict movie ratings.



Build a recommender system that predicts the user rating for the media. The goal would be to show such recommendations where users give high ratings.

Types of user feedback#

Generally, there are two types of feedback coming from an end-user for a given recommendation.

1. Explicit feedback:

A user provides an explicit assessment of a recommendation. In our case, it would be a star rating, e.g., a user rates the movie four out of five stars.

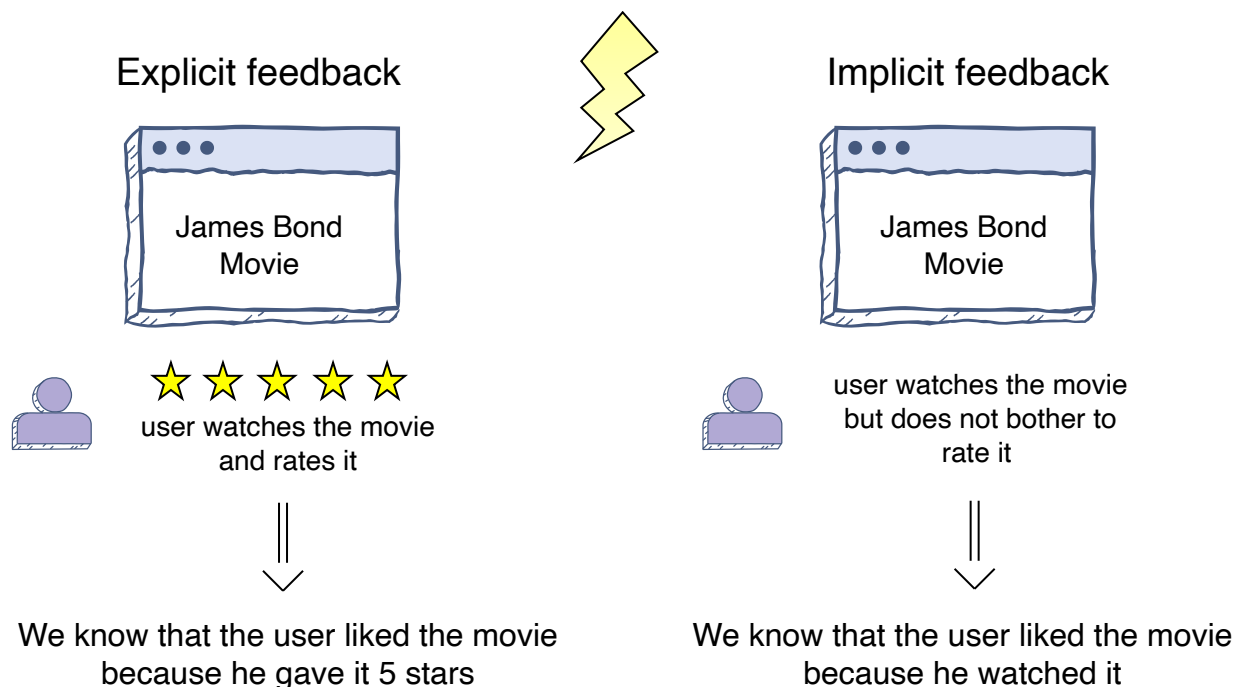


Here, the recommendation problem will be viewed as a rating prediction problem.

2. Implicit feedback:

Implicit feedback is extracted from a user's interaction with the recommended media. Most often, it is binary in nature. For instance, a user watched a movie (1), or they did not watch the movie (0).

Here, the recommendation problem will be viewed as a ranking problem.

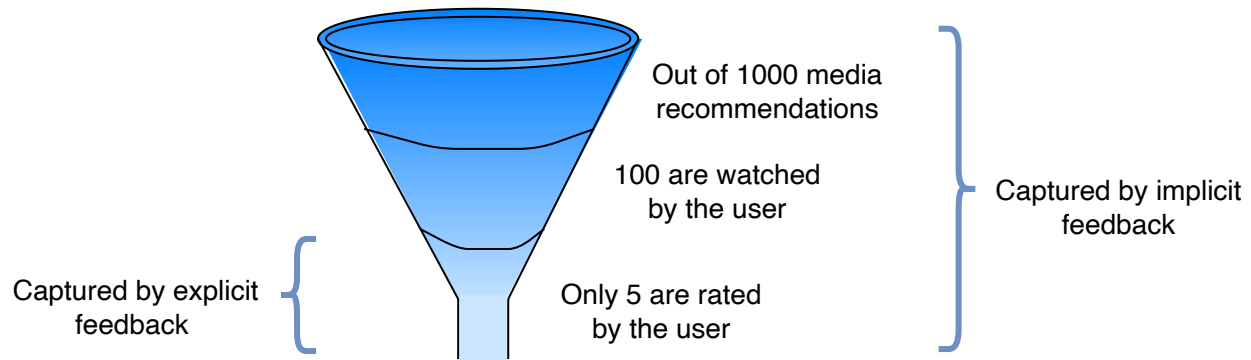


Explicit feedback is provided by user, whereas implicit feedback is assumed by user action

One key advantage of utilizing implicit feedback is that it allows collecting a large amount of training data. This allows us to better personalize recommendations by getting to know our users more.

However, this is not the case with explicit feedback. People seldom rate the movies after watching them, as depicted by the funnel below.





The difference in data available to a system built on explicit feedback versus a system built on implicit feedback.

Explicit feedback faces the *missing not at random* (MNAR) problem. Users will generally rate those media recommendations that they liked. This means $\frac{4}{5}$ or $\frac{5}{5}$ star ratings are more common than $\frac{1}{5}$, $\frac{2}{5}$ or $\frac{3}{5}$. Therefore, we won't get much information on the kind of movies that a user does not like. Also, movies with fewer ratings would have less impact on the recommendation process.

[← Back](#)[Next →](#)[Online Experimentation](#)[Metrics](#)☒ Mark as Completed[! Report an Issue](#)