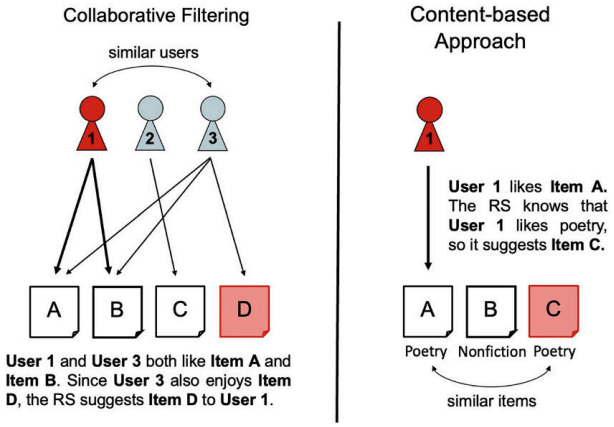
Content-based filtering

Collaborative filtering vs Content-based filtering



https://www.researchgate.net/figure/Collaborative-filtering-vs-content-based-approach_fig1_376715505

Collaborative Filtering:

- Recommends items based on ratings or behavior of *similar users*.
- Uses only user-item interaction data like ratings, clicks, or purchases.
- Learns **user preferences and item features implicitly** by analyzing the collective user behavior.
- Provides recommendations by finding patterns in user ratings (e.g., users who liked item A also liked item B).

Content-Based Filtering:

- Recommends items based on the **features of both users and items** explicitly.
- Requires having feature information about users (e.g., age, gender,..) and items (e.g., genre, release year, average rating).
- Constructs feature vectors ($x_u^{(j)}$) users j and items i ($v_m^{(i)}$) based on known attributes and preferences.
- Predicts a match between a user and an item by computing the **dot product** of their feature vectors.

Example:

User features: User vector might represent preferences (e.g., how much they like romance or action movies).

- Age
- Gender *(1 hot)*
- Country *(1 hot, 200)*
- Movies watched *(1000)*
- Average rating per genre
- ...

Movie features: Movie vector might represent how much the movie fits these categories.

- Year
- Genre/Genres
- Reviews
- Average rating

- ...

Content-based filtering: Learning to match

The rating of user j on movie i is predicted as (no b more):

$$w^{(j)} \cdot x^{(i)} \rightarrow v_u^{(j)} \cdot v_m^{(i)}$$

- $v_u^{(j)}$ computed from $x_u^{(j)}$
- $v_m^{(i)}$ computed from $x_m^{(i)}$

User preferences:

$$X_u^{(j)} \rightarrow V_u^{(j)} = \begin{bmatrix} 4.9 \text{ (likes)} \\ 0.1 \text{ (likes)} \\ \vdots \\ 3.0 \end{bmatrix}$$

Movie features:

$$X_m^{(i)} \rightarrow V_m^{(i)} = \begin{bmatrix} 4.5 \text{ (romance)} \\ 0.2 \text{ (action)} \\ \vdots \\ 3.5 \end{bmatrix}$$

$\Rightarrow$ Both have **the same dimension**.

Key Differences:

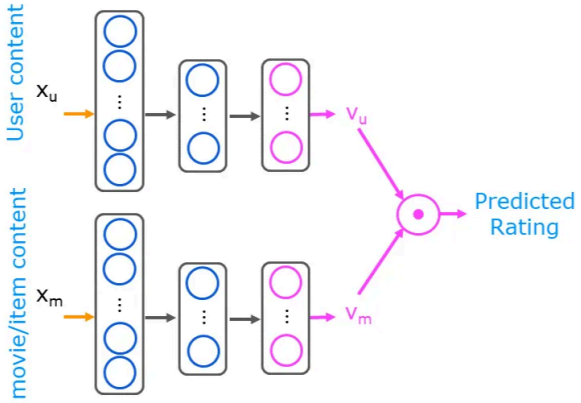
- **Collaborative filtering** uses only observed user behaviors and tastes, content-based filtering uses descriptive features of items and users.
- **Content-based filtering** usually requires manual or derived features for users and items.
- **Collaborative filtering** can discover **hidden patterns** without needing explicit features.
- **Content-based filtering** can recommend **new items** with good feature data even without user ratings.

Deep learning for content-based filtering

Neural Network architecture

An effective approach to building a content-based algorithm is through deep learning. A neural network generates feature vectors v by taking the initial features w, x as inputs. By passing them through multiple layers, it compresses the information into a final output vector of a chosen size.

Finally, these outputs are connected by using a node that perform **dot product** $\cdot$



With $g(z)$ is a sigmoid function. **Prediction:** $v_u^{(j)} \cdot v_m^{(i)}$

$$g\left(v_u^{(j)} \cdot v_m^{(i)}\right) \text{ to predict the probability } y^{(i,j)} = 1$$

Cost function

$$J = \sum_{(i,j):r(i,j)=1} \left(v_u^{(j)} \cdot v_m^{(i)} - y^{(i,j)}\right)^2 + \text{NN regularization term}$$

Learned user and item vectors:

- $v_u^{(j)}$ is a vector of length 32 that describes user j with features $x_u^{(j)}$
- $v_m^{(i)}$ is a vector of length 32that describes movie i with features $x_m^{(i)}$

Finding similar items

To find a similar item j to item i,

$$distance = ||v_m^{(k)} - v_m^{(i)}||^2$$

Note, this can be pre-computed ahead of time. In other words, the calculation of similar items can be done prior to executing the model algorithm itself.

Recommending from a large catalogue

How to effieciently find recommendation from a large set of items?

Today a large movie streaming site may have thousands of movies or a system that is trying to decide what ad to show. May have a catalog of millions of ads to choose from. Or a music streaming sites may have 10s of millions of songs to choose from.

- Movies: $1000+$
- Ads: $1m+$
- Songs: $10m+$
- Products: $10m+$

> In real-world scenarios, recommendation systems deal with millions of users and billions of items. To manage this huge scale, they are usually structured into two sequential model types—**retrieval** and **ranking**—with each stage progressively reducing the pool of candidate items.

Two steps: Retrieval and Ranking

1. **Retrieval:**

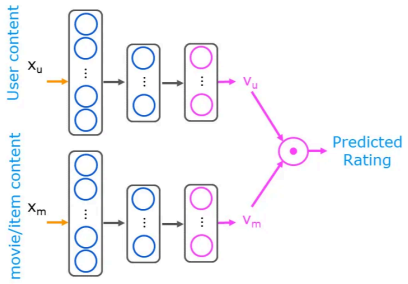
- Generate large list of **plausible item** candidates

  e.g.

  1. For each of the last 10 movies watched by the user, find 10 most similar movies $||v_m^{(k)} - v_m^{(i)}||^2$
  2. For most viewed 3 genres, find the top 10 movies
  3. Top 20 movies in the country
- Combine retrieved items into list, removing duplicates and items already watched/purchased

2. **Ranking**

- Use a trained model to order the retrieved items by relevance
- Present the ranked list to the user



Increasing the number of items retrieved in the first stage can **improve recommendation** quality but **slows down response time**.

To **balance** this **trade-off**, **offline experiments** are run to test whether retrieving more items actually leads to more relevant suggestions. (i.e., p(yij)) = 1 of items displayed to user are higher).

Ethical use of recommender systems

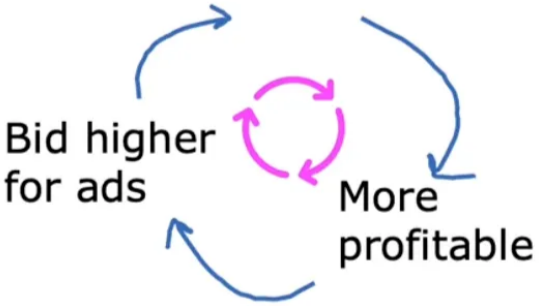
What is the goal of the recommender system?

Recommend:

- Movies most likely to be rated 5 stars by user
- Products most likely to be purchased
- Ads most likely to be clicked on *(+ high bid)*
- Products generating the largest profit
- Video leading to maximum watch time

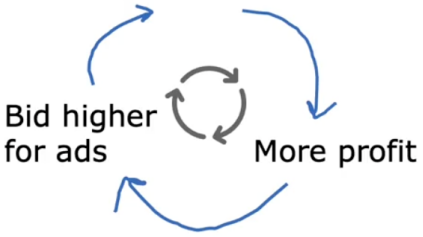
Travel industry

Good travel experience to more users



Payday loans

Squeeze customers more



Amelioration: Do not accept ads from exploitative businesses

Other problematic cases:

- Maximizing user engagement (e.g. watch time) has led to large social media/video sharing sites to amplify conspiracy theories and hate/toxicity

→ **Amelioration** : Filter out problematic content such as hate speech, fraud, scams and violent content

- Can a ranking system maximize your profit rather than users' welfare be presented in a transparent way?

  → **Amelioration** : Be transparent with users

TensorFlow implementation

```
user_NN = tf.keras.models.Sequential([
    tf.keras.layers.Dense(256, activation='relu'),
    tf.keras.layers.Dense(128, activation='relu'),
    tf.keras.layers.Dense(32)
])

item_NN = tf.keras.models.Sequential([
    tf.keras.layers.Dense(256, activation='relu'),
    tf.keras.layers.Dense(128, activation='relu'),
    tf.keras.layers.Dense(32)
])

# create the user input and point to the base network
input_user = tf.keras.layers.Input(shape=(num_user_features))
vn = user_NN(input_user)
vn = tf.linalg.l2_normalize(vn, axis=1)

# create the item input and point to the base network
input_item = tf.keras.layers.Input(shape=(num_item_features))
vm = user_NN(input_item)
vm = tf.linalg.l2_normalize(vm, axis=1)

# measure the similarity of the two vector outputs
output = tf.keras.layers.Dot(axes=1)([vn, vm])

# specify the inputs and output of the model
model = Model([input_user, input_item], output)

# Specify the cost function
cost_fn = tf.keras.losses.MeanSquaredError()
```

Vector x_u and vector x_m must be of the same dimension, where x_u is the input features vector for a user (age, gender, etc.) x_m is the input features vector for a movie (year, genre, etc.) True or false?

- True
- False 👍👍

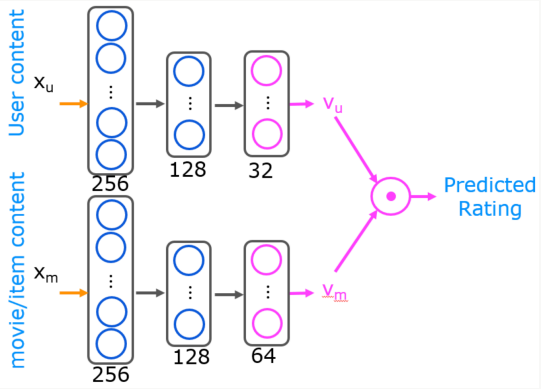
Explain: These vectors can be different dimensions.

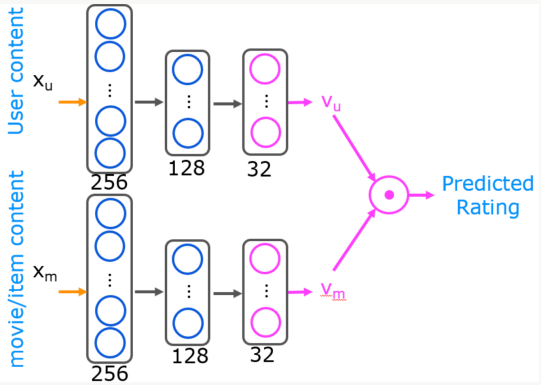
If we find that two movies, i and k, have vectors $v_m^{(i)}$ and $v_m^{(k)}$ that are similar to each other (i.e., $||v_m^{(k)} - v_m^{(i)}||^2$ is small), then which of the following is likely to be true? Pick the best answer.

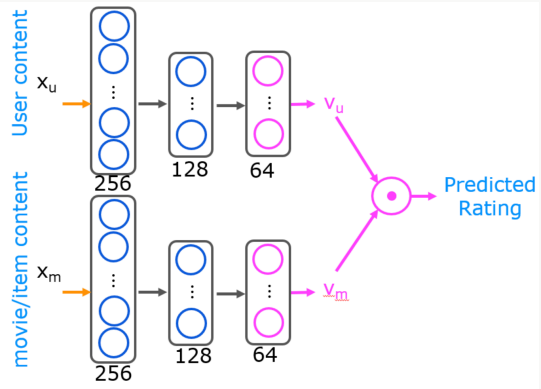
- A user that has watched one of these two movies has probably watched the other as well.
- The two movies are very dissimilar.
- We should recommend to users one of these two movies, but not both.
- The two movies are similar to each other and will be liked by similar users. 👍👍

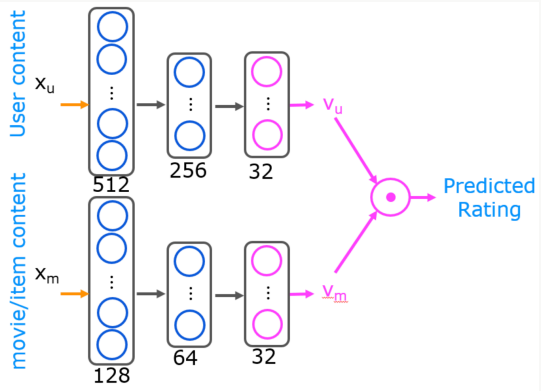
Explain: Similar movies generate similar v_m

Which of the following neural network configurations are valid for a content based filtering application? Please note carefully the dimensions of the neural network indicated in the diagram. Check all the options that apply:









Except the first one because v_u, v_m must have the same size.

You have built a recommendation system to retrieve musical pieces from a large database of music, and have an algorithm that uses separate retrieval and ranking steps. If you modify the algorithm to add more musical pieces to the retrieved list (i.e., the retrieval step returns more items), which of these are likely to happen? Check all that apply.

- [x] ~~The system's response time might increase (i.e., users have to wait longer to get recommendations)~~

  Explain: A larger retrieval list may take longer to process which may *increase* response time.
- [ ] The system's response time might decrease (i.e., users get recommendations more quickly)
- [ ] The quality of recommendations made to users should stay the same or worsen.
- [x] ~~The quality of recommendations made to users should stay the same or improve.~~

  Explain: A larger retrieval list gives the ranking system more options to choose from which should maintain or improve recommendations

To speed up the response time of your recommendation system, you can pre-compute the vectors v_m for all the items you might recommend. This can be done even before a user logs in to your website and even before you know the or $v_u$vector. True/False?

- True 👍👍
- False

Explain: The output of the item/movie neural network, v_m is not dependent on the user network when making predictions. Precomputing the results speeds up the prediction process.