AASD 4004
Machine Learning - II

Applied AI Solutions Developer Program

Module 06 Recommenders

Vejey Gandyer

Agenda

Recommenders

Why Recommenders?

Ways of recommending

How Recommenders work?

Types of Recommenders

Evaluation of Recommenders

Recommenders

What is it?

Recommenders

A recommender filters the data using different algorithms and recommends the most relevant items to users

Captures the past behaviour of a customer (Data collection) and based on that, recommends products which the users might be likely to buy

Why

Why Recommenders?

To create an amazing user experience

To increase product sales by recommending similar or complementary products to the user

To add additional ad stream revenue by recommending specific products

Ways of recommending

Ways of recommending

- Recommend items to a user which are most popular among all the users

Drawback: Every user will see the same recommendations

- Divide the users into multiple segments based on their preferences (user features) and recommend items to them based on the segment they belong to

Drawback: More computation time as number of users increases

Cold Start Problem

How Recommenders work?

How Recommenders work?

- Data is collected from users (Onboarding)
- Models are created from the collected data (Algorithms)
- Recommendations are pushed to the user (Action)

Recommendation Engine = Call-to-Action push from recommendation models trained by user collected data

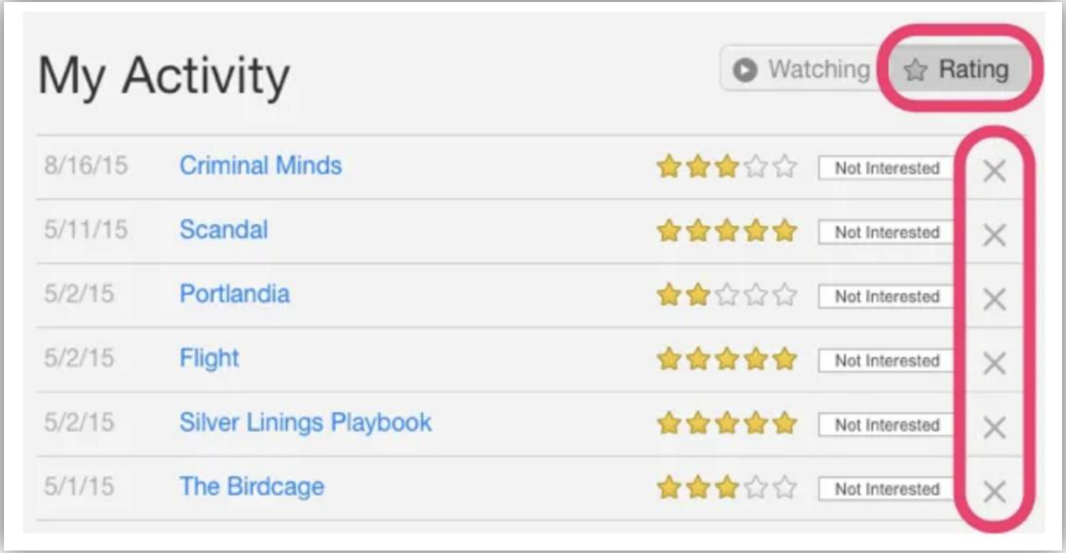
Data Collection (Onboarding)

Explicit

Data provided by users explicitly in the way of Product Ratings, Rankings, Likes, and so on.

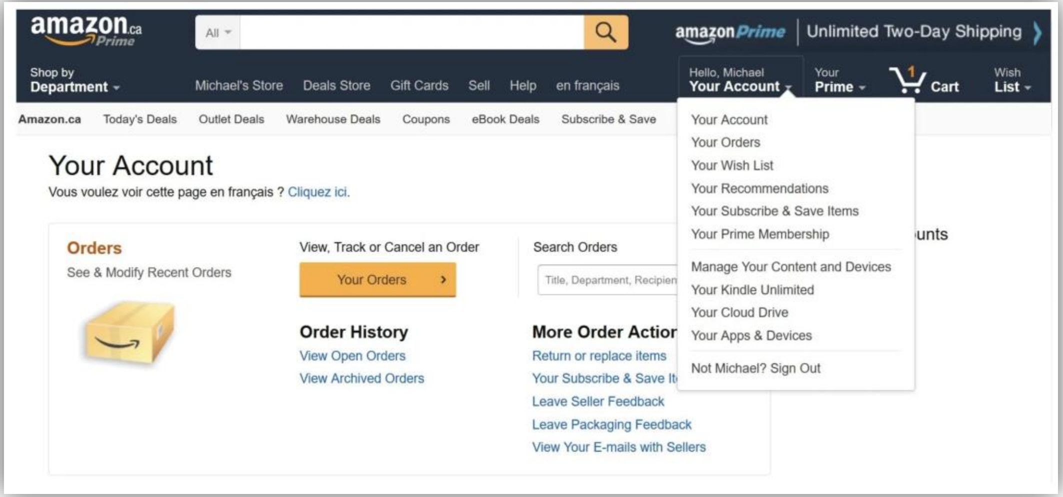
Implicit

Data not provided by users explicitly but collected through user's behaviour with the product in the way of Clickstream, Search History, Order History, Website Navigation, Analytics and so on


My Activity

Watching | Rating

8/16/15	Criminal Minds	★★★☆☆	Not Interested	✕
5/11/15	Scandal	★★★★★	Not Interested	✕
5/2/15	Portlandia	★★☆☆☆	Not Interested	✕
5/2/15	Flight	★★★★★	Not Interested	✕
5/2/15	Silver Linings Playbook	★★★★★	Not Interested	✕
5/1/15	The Birdcage	★★★☆☆	Not Interested	✕


amazon.ca Prime

All

amazonPrime | Unlimited Two-Day Shipping

Shop by Department · Michael's Store · Deals Store · Gift Cards · Sell · Help · en français · Hello, Michael Your Account · Your Prime · Cart · Wish List

Amazon.ca · Today's Deals · Outlet Deals · Warehouse Deals · Coupons · eBook Deals · Subscribe & Save

Your Account

Vous voulez voir cette page en français ? Cliquez ici.

Orders

See & Modify Recent Orders

View, Track or Cancel an Order

Your Orders

Order History

View Open Orders

View Archived Orders

Search Orders

Title, Department, Recipien

More Order Actior

Return or replace items

Your Subscribe & Save It

Leave Seller Feedback

Leave Packaging Feedback

View Your E-mails with Sellers

Your Account

Your Orders

Your Wish List

Your Recommendations

Your Subscribe & Save Items

Your Prime Membership

Manage Your Content and Devices

Your Kindle Unlimited

Your Cloud Drive

Your Apps & Devices

Not Michael? Sign Out

unts

Types of Recommenders

Types of Recommenders

Content Filtering

Collaborative Filtering

Hybrid

Types of Recommenders

Content Filtering

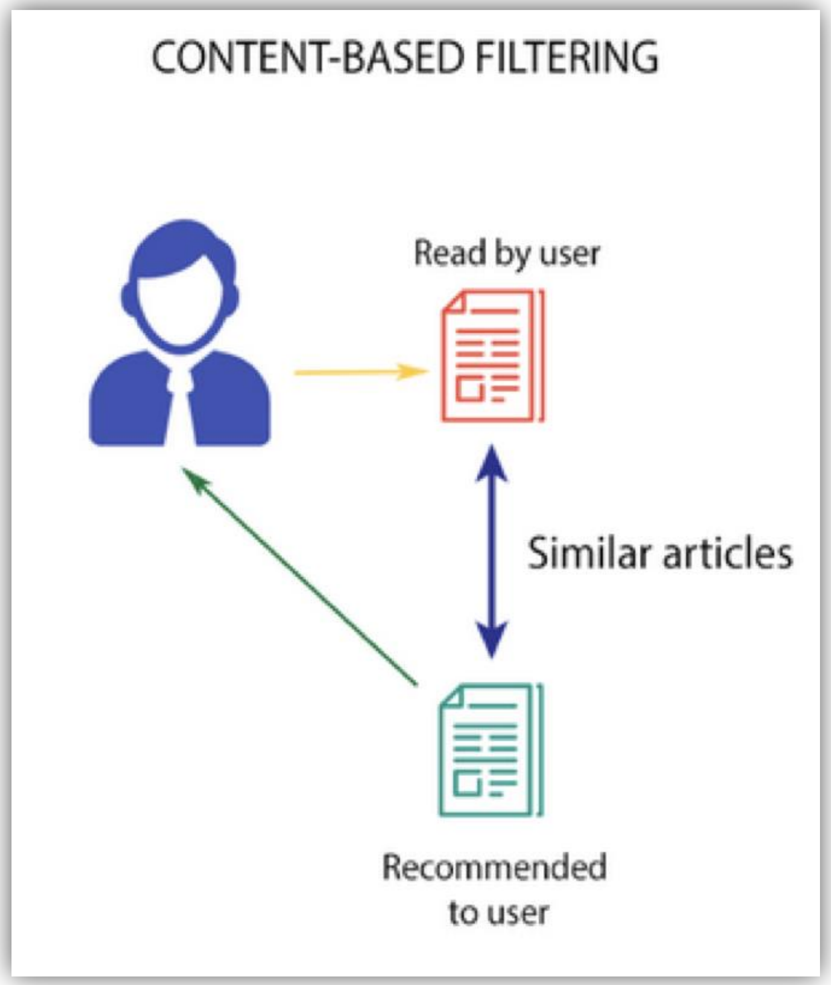
Content-based Filtering

Movie recommendation

- If a person likes "Inception", then algorithm will recommend movies that fall under the same genre

How does the algorithm know "**genre**"

- Profile Vector - Users past behaviour (movies liked/disliked, ratings)
- Item Vector - movies (genre, cast, director, music, etc)



Content-based Filtering

How recommendations are computed?

Similarity measured using one of the distances

- Cosine similarity

$$sim(A,B) = \cos(\theta) = \frac{A \cdot B}{\|A\|\|B\|}$$

- Euclidean similarity

$$\text{Euclidean Distance} = \sqrt{(x_1 - y_1)^2 + \ldots + (x_N - y_N)^2}$$

- Pearsons Correlation

$$sim(u,v) = \frac{\sum(r_{ui} - \bar{r}_u)(r_{vi} - \bar{r}_v)}{\sqrt{\sum(r_{ui} - \bar{r}_u)^2}\sqrt{\sum(r_{vi} - \bar{r}_v)^2}}$$

Content-based Filtering

Top n

- Movies are sorted with respect to similarity and top n movies are recommended

Rating scale

- Threshold is set and all the movies above threshold are recommended

Drawbacks:

- Movies of same genre are recommended
- Movies of different genres are not recommended

Types of Recommenders

Collaborative Filtering

Collaborative Filtering

Algorithm uses **user behaviour** for recommending items

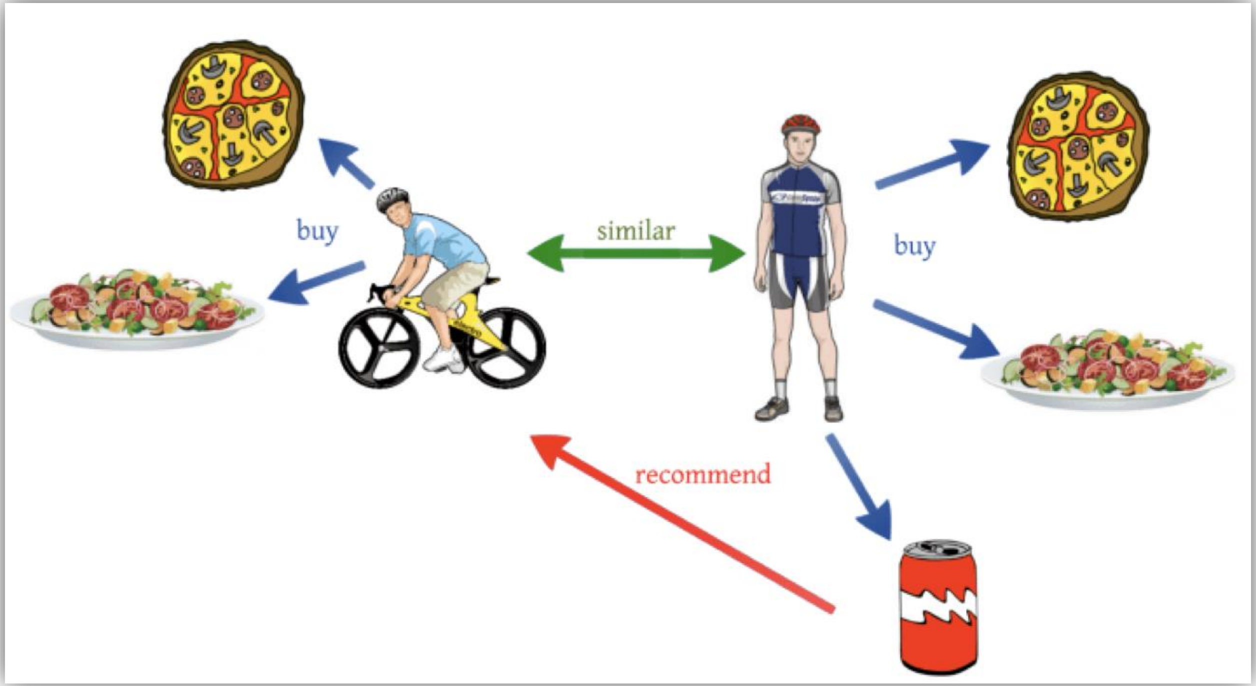
- User-User Collaborative filtering
- Item-Item Collaborative filtering

Types of Recommenders

User-User Collaborative Filtering

User-User Collaborative Filtering

- Finds **Similarity score** between users
- For a user, pick the **most similar users**
- **Recommend the items** which these most similar users bought/liked/rated previously

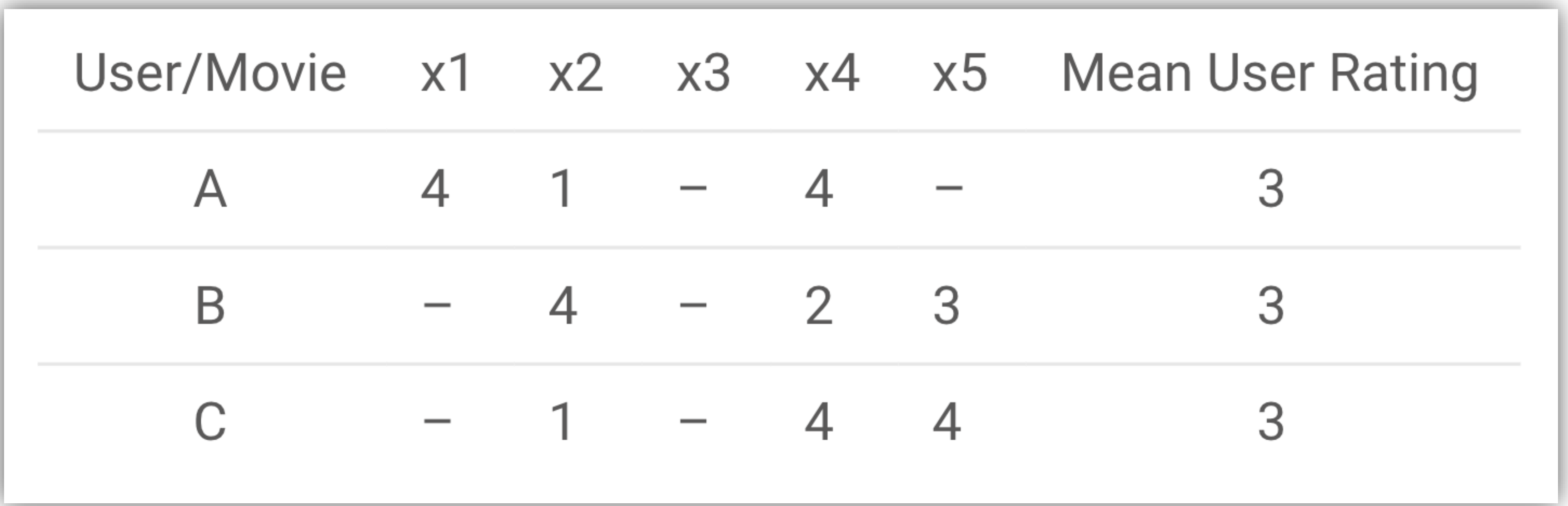


User-User Collaborative Filtering

1. Use Pearson correlation to find similarity between the user u and v.
2. First we find the items rated by both the users and based on the ratings, correlation between the users is calculated.
3. The predictions can be calculated using the similarity values. This algorithm, first of all calculates the similarity between each user and then based on each similarity calculates the predictions. Users having higher correlation will tend to be similar.
4. Based on these prediction values, recommendations are made

$$P_{u,i} = \frac{\sum_v (r_{v,i} * s_{u,v})}{\sum_v s_{u,v}}$$

User-User Collaborative Filtering



User/Movie	x1	x2	x3	x4	x5	Mean User Rating
A	4	1	–	4	–	3
B	–	4	–	2	3	3
C	–	1	–	4	4	3

$r_{AC} = [(1-3)*(1-3) + (4-3)*(4-3)]/[((1-3)^2 + (4-3)^2)^{1/2} * ((1-3)^2 + (4-3)^2)^{1/2}] = 1$

$r_{BC} = [(4-3)*(1-3) + (2-3)*(4-3) + (3-3)*(4-3)]/[((4-3)^2 + (2-3)^2 + (3-3)^2)^{1/2} * ((1-3)^2+(4-3)^2 + (4-3)^2)^{1/2}] = -0.866$

User-User Collaborative Filtering

Drawbacks:

- Time consuming
  - Calculates similarity for each user and then calculates prediction for each similarity score

Solution: (Select few users instead of all)

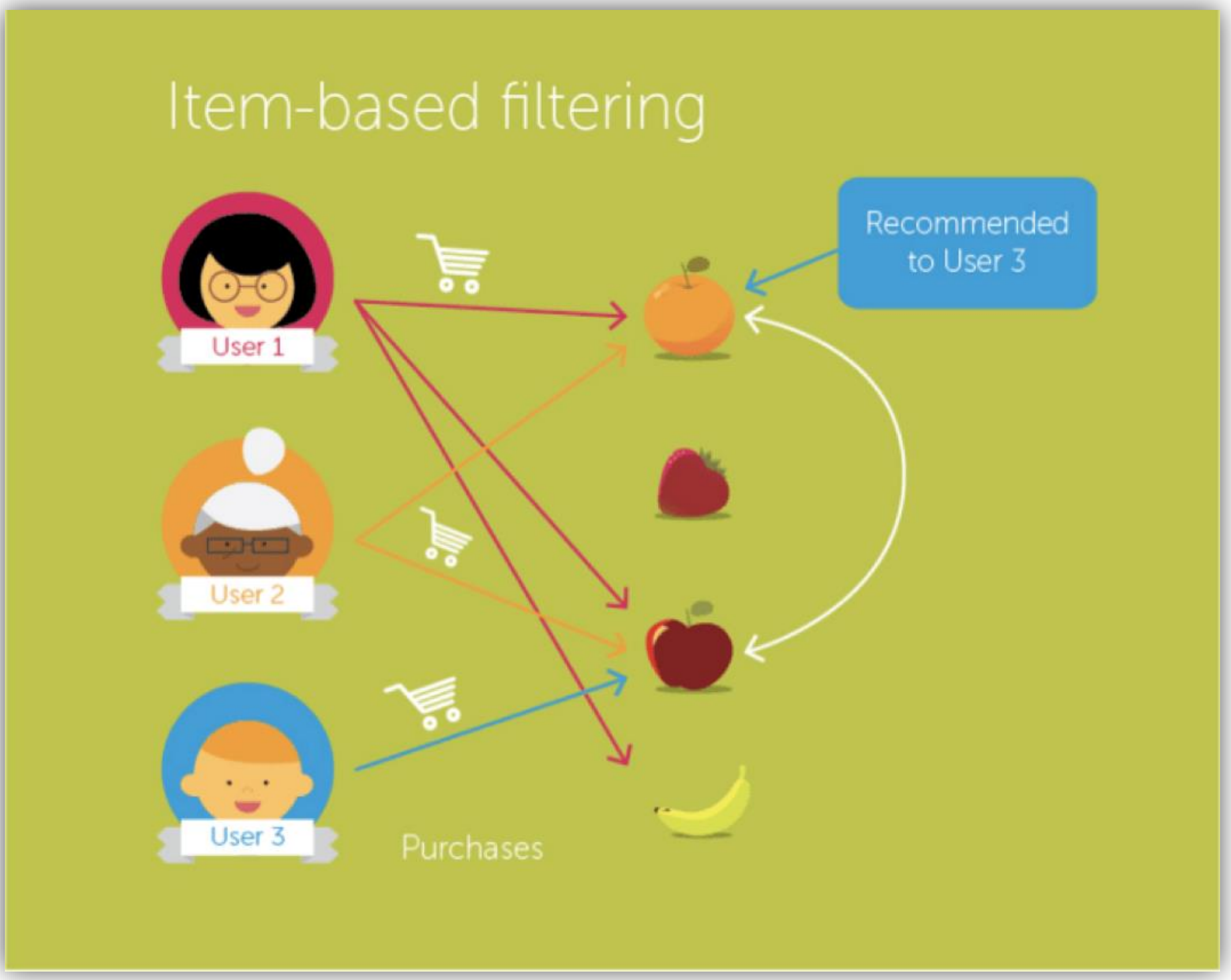
- Select a threshold similarity and choose all users above that value
- Randomly select the users
- Arrange neighbours in descending order of their similarity value and choose top-N users
- Use clustering for choosing neighbours

Types of Recommenders

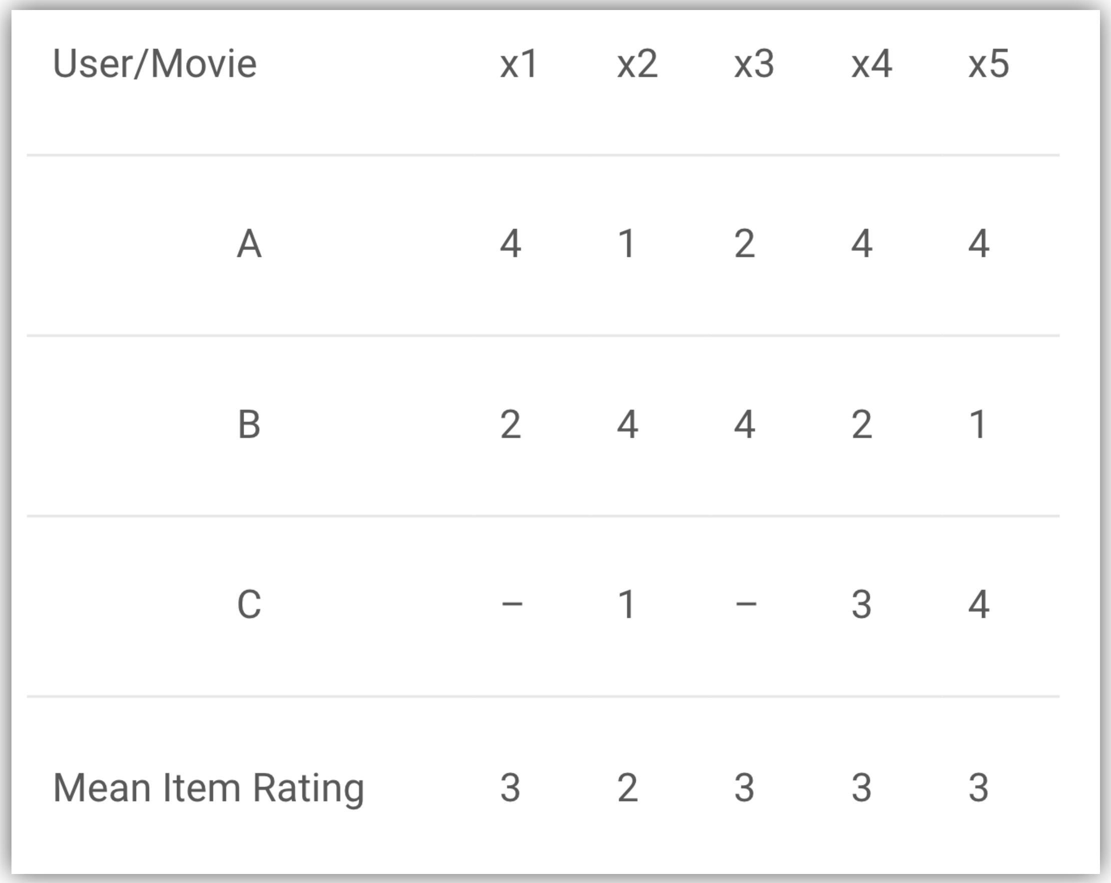
Item-ItemCollaborative Filtering

Item-Item Collaborative Filtering

- Finds **Similarity score** between items
- For an item, pick the **most paired items**
- **Recommend** the items which these most similar users bought/liked/rated previously



Item-Item Collaborative Filtering



User/Movie	x1	x2	x3	x4	x5
A	4	1	2	4	4
B	2	4	4	2	1
C	–	1	–	3	4
Mean Item Rating	3	2	3	3	3

$$C_{14} = [(4-3)*(4-3) + (2-3)*(2-3)] / [((4-3)^2 + (2-3)^2)^{1/2} * ((4-3)^2 + (2-3)^2)^{1/2}] = 1$$

$$C_{15} = [(4-3)*(4-3) + (2-3)*(1-3)] / [((4-3)^2 + (2-3)^2)^{1/2} * ((4-3)^2 + (1-3)^2)^{1/2}] = 0.94$$

Item-Item Collaborative Filtering

What will happen if a new item or a new user comes into the recommendation ecosystem?

- No data of that item or user exists
- No recommendations possible (**Cold Start problem**)

Visitor Cold Start - a new user

- Solution: **Popularity recommender** - Recommend most popular products to the new user

Product Cold Start - a new item

- Solution: **Content-based filtering** - Use content of the new product to recommend

Movie Recommender

https://grouplens.org/datasets/movielens/100k/

Matrix Factorization Recommender

Matrix Factorization Recommender

Problem:

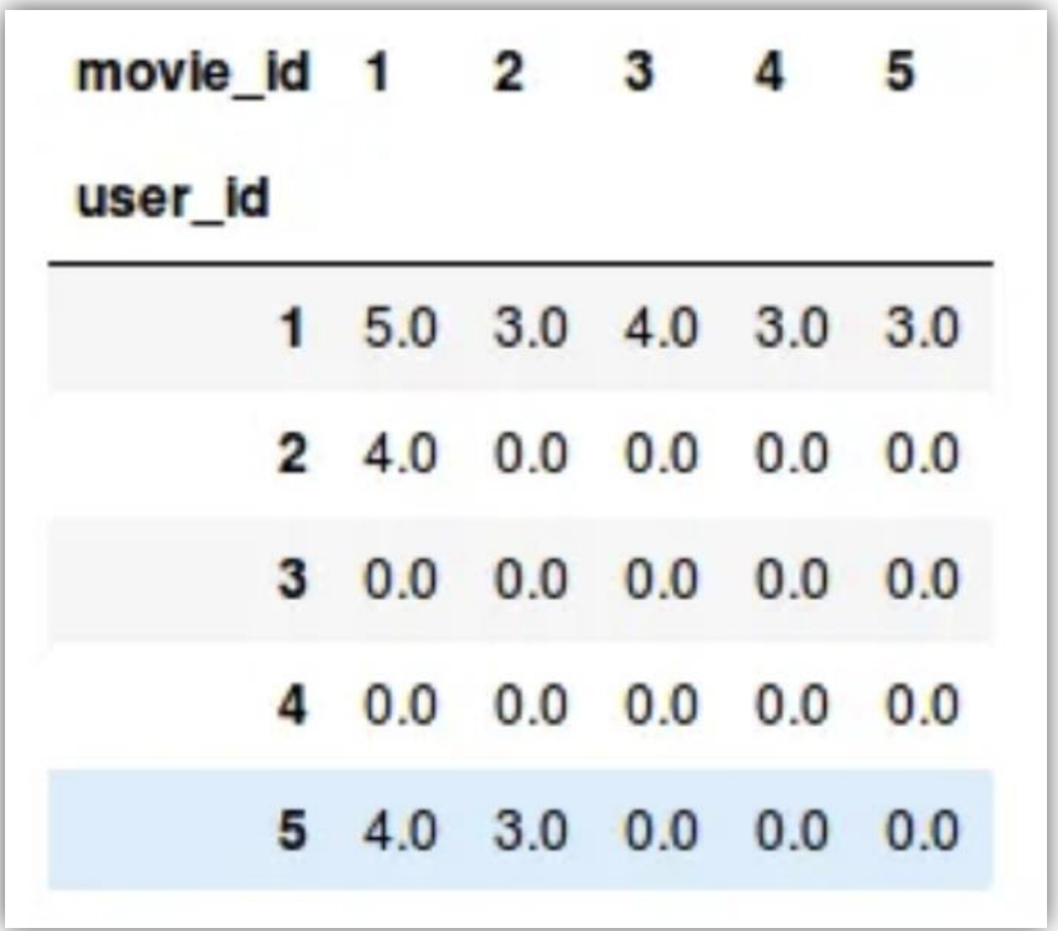
Not every movie is rated by every user. Find a set of features which can define how a user rates the movies. These are called latent features.

We need to find a way to extract the most important latent features from the existing features.

Solution:

Matrix factorization is a technique which uses the lower dimension dense matrix and helps in extracting the important latent features.

Matrix Factorization Recommender



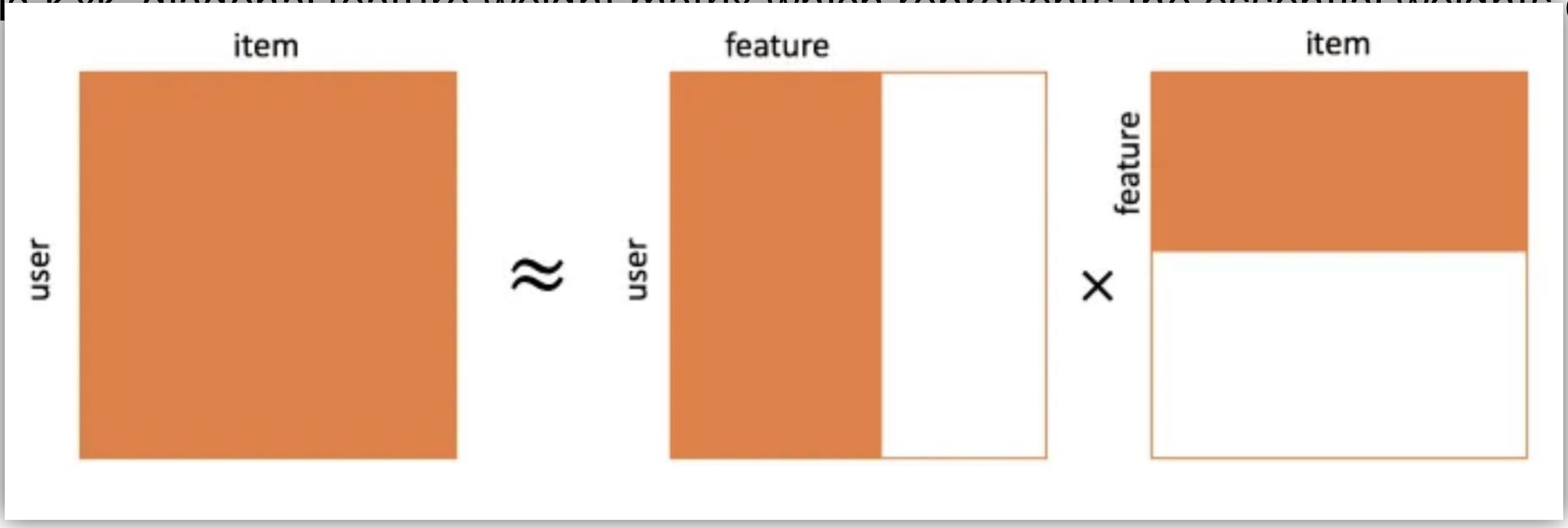
movie_id	1	2	3	4	5
user_id					
1	5.0	3.0	4.0	3.0	3.0
2	4.0	0.0	0.0	0.0	0.0
3	0.0	0.0	0.0	0.0	0.0
4	0.0	0.0	0.0	0.0	0.0
5	4.0	3.0	0.0	0.0	0.0

<u>User-Item matrix</u>
user_id : every user
movie_id : every item
5.0 - 1.0 (Highly liked - Least liked)
0.0 - Movie not rated by the user

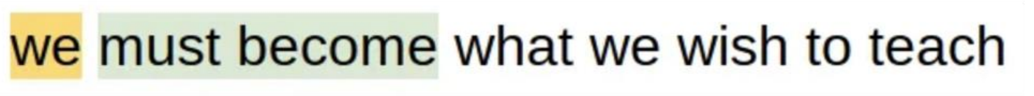
Matrix Factorization Recommender

M is the total number of users
N is the total number of movies
K is the total latent features
R is MxN user-movie rating matrix
P is MxK user-feature affinity matrix which represents association between users and features
Q is NxK item-feature relevance matrix which represents association between movies and features
Σ is KxK diagonal feature weight matrix which represents the essential weights of features

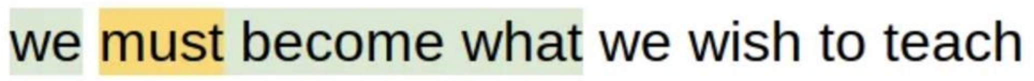
$R=P\Sigma Q^T$



Word2Vec Revisited

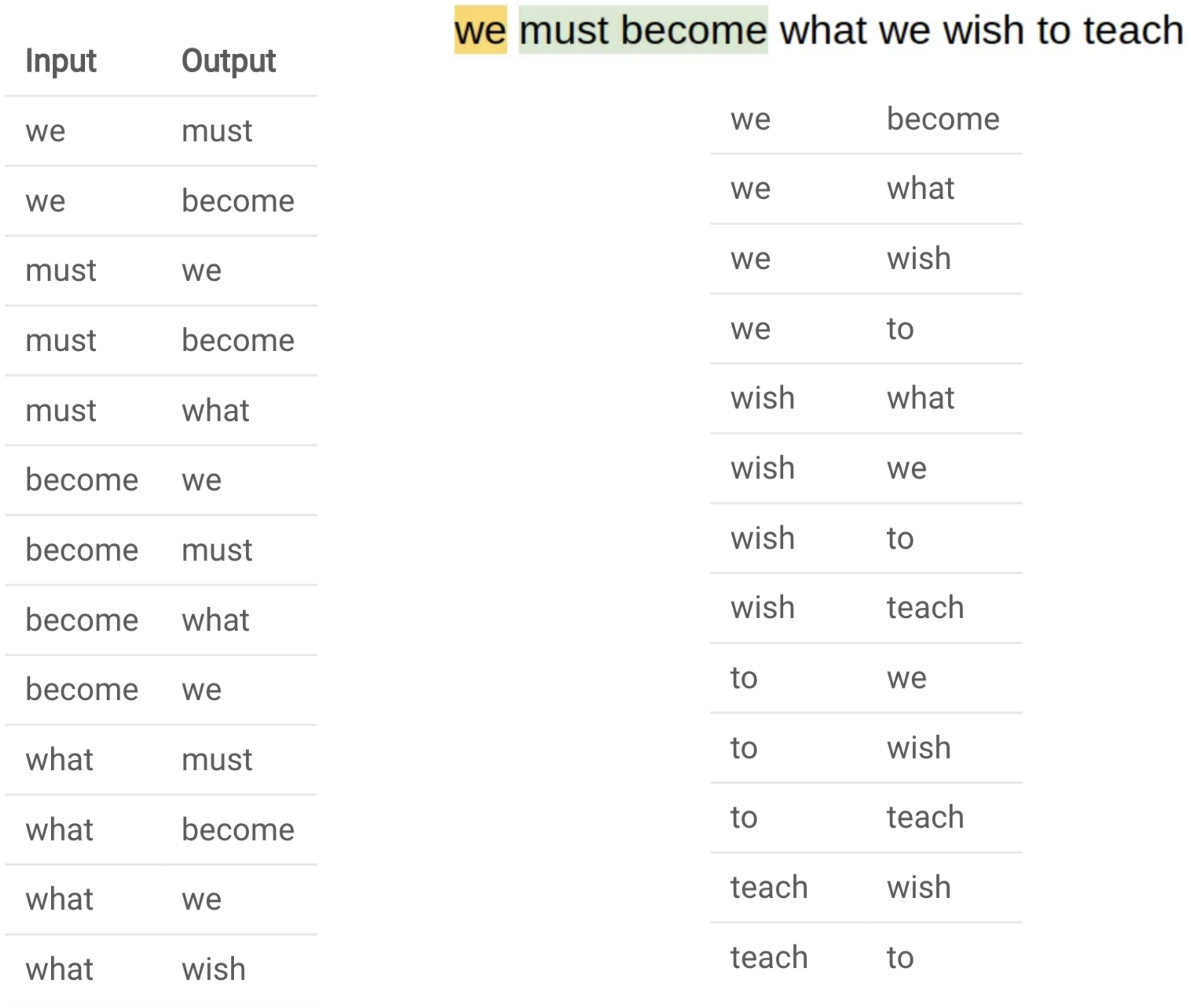

we must become what we wish to teach

Input	Output
we	must
we	become


we must become what we wish to teach

Input	Output
we	must
we	become
must	we
must	become
must	what

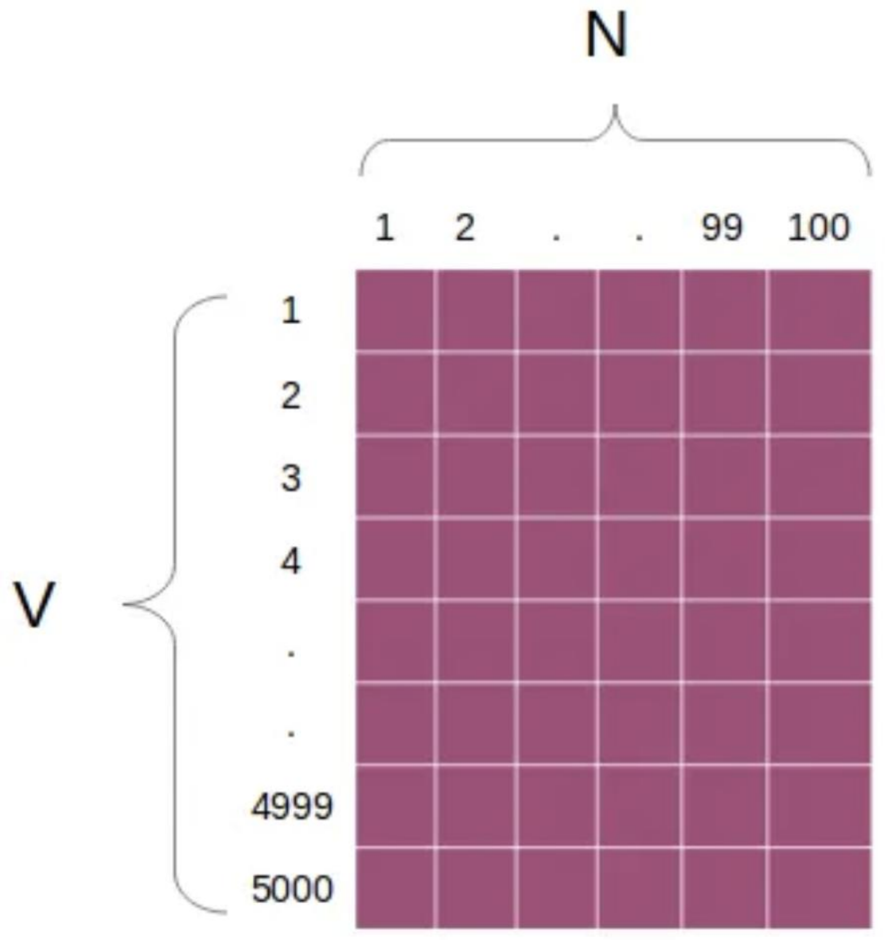
Word2Vec Revisited



we must become what we wish to teach

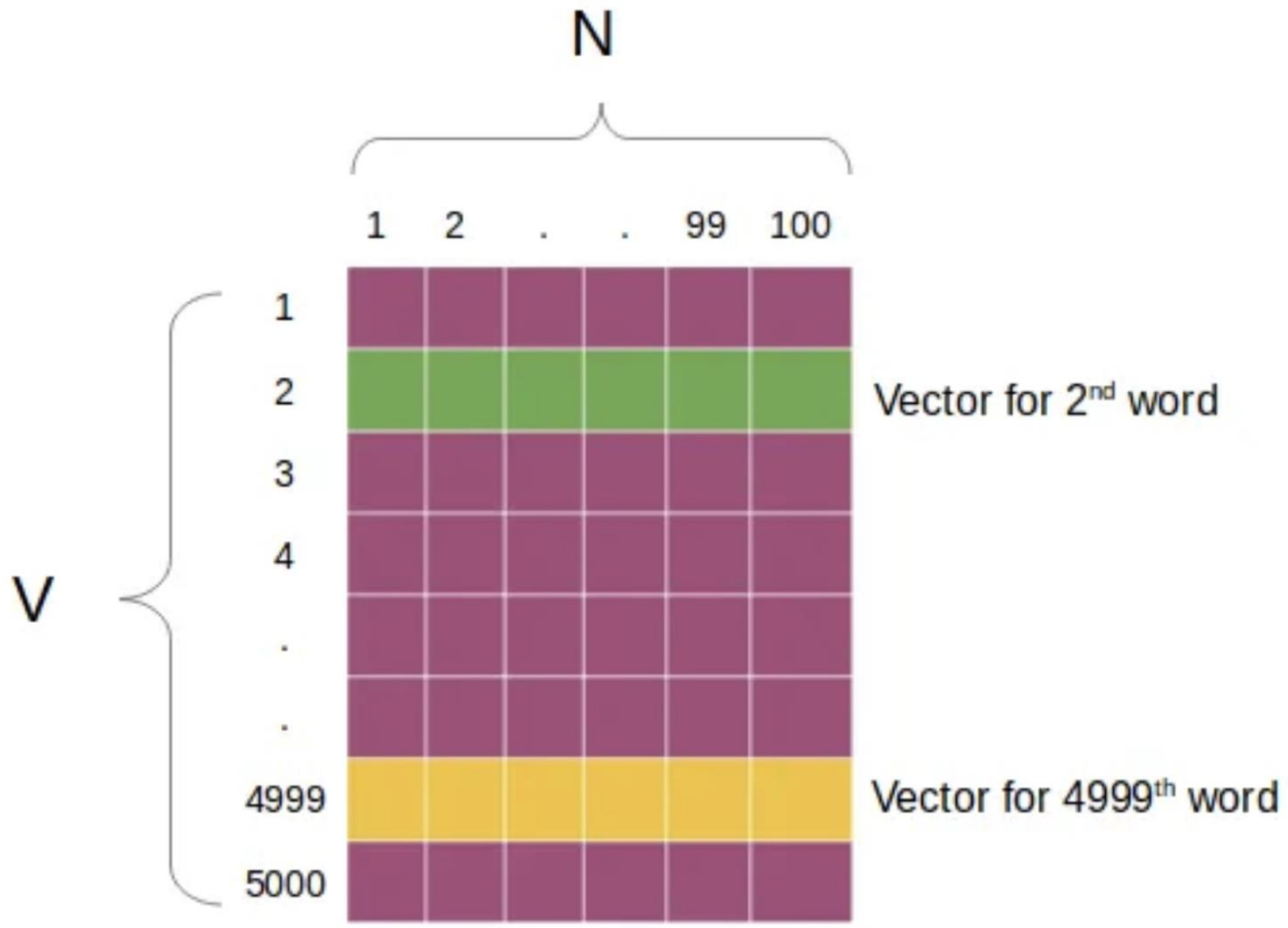
Input	Output
we	must
we	become
must	we
must	become
must	what
become	we
become	must
become	what
become	we
what	must
what	become
what	we
what	wish
we	become
we	what
we	wish
we	to
wish	what
wish	we
wish	to
wish	teach
to	we
to	wish
to	teach
teach	wish
teach	to

Weight Matrix (V x N)

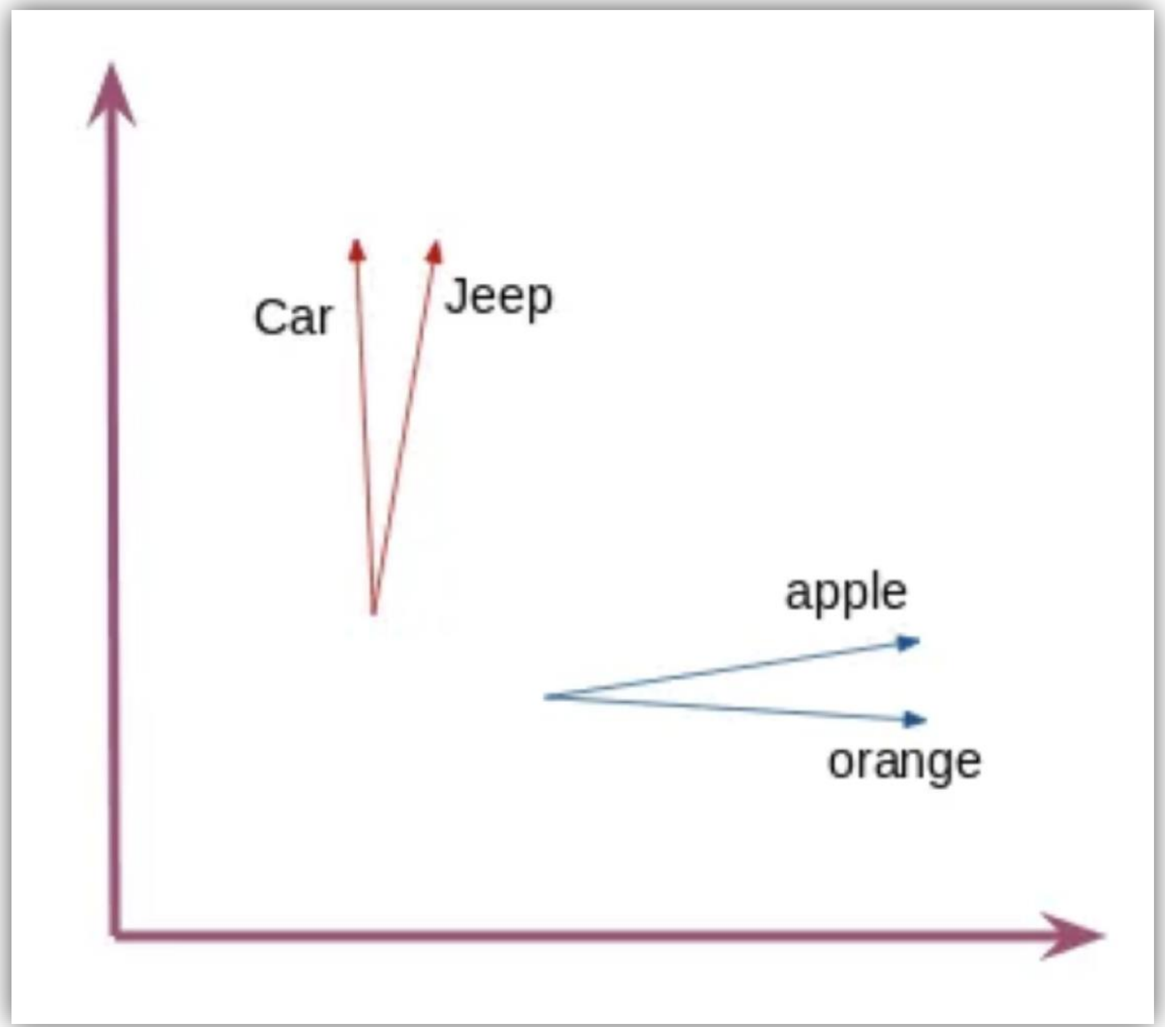


W_{VxN} weight matrix

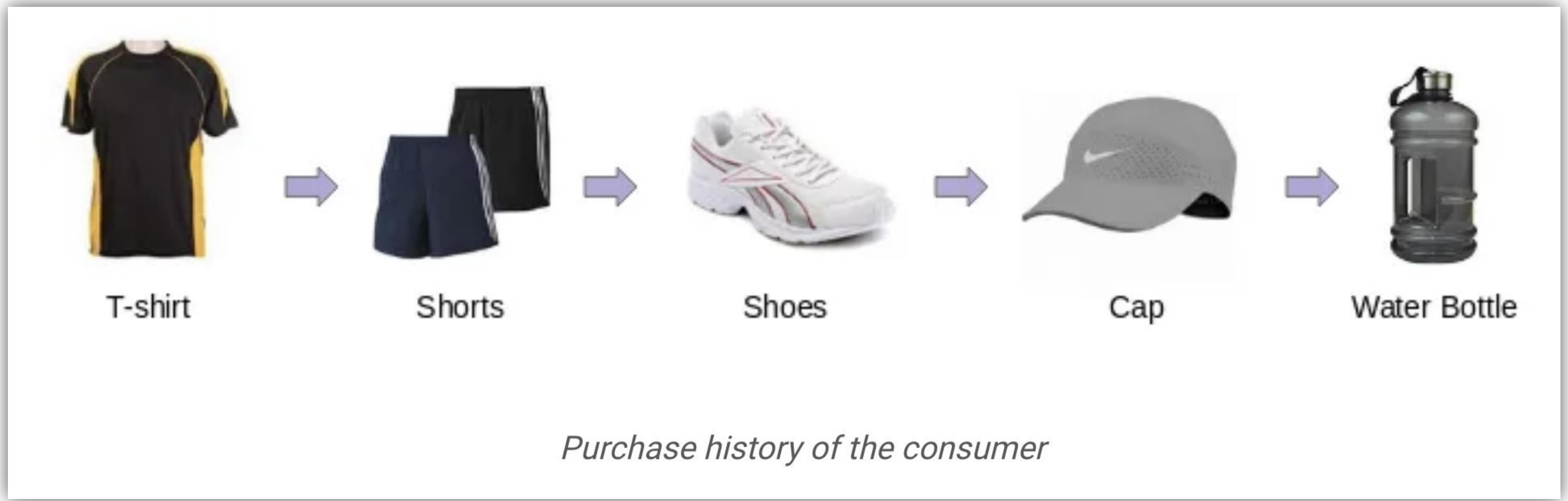
Weight Matrix (V x N)



Word Vectors Cosine Similarity

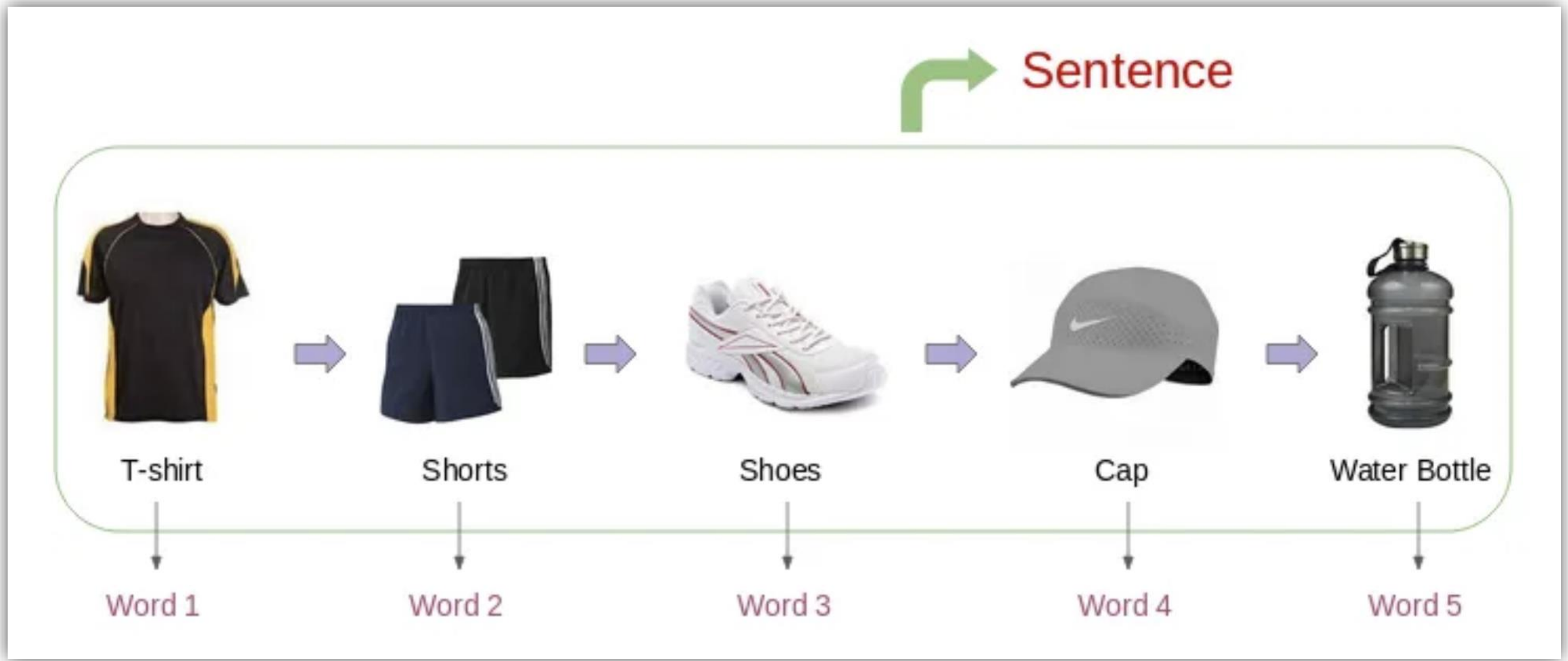


Problem: Product Recommendation



Purchase history of the consumer

Word2Vec Recommendation



Dataset

https://archive.ics.uci.edu/ml/machine-learning-databases/00352/

Simple_Recommendation_System.ipynb

Evaluation Metrics

Recall

Precision

RMSE

Mean Reciprocal Rank

Mean Average Precision

Normalized Discounted Cumulative Gain

Recall

What proportion of items that a user likes were actually recommended

$$\text{Recall} = \frac{tp}{tp + fn}$$

- Here tp represents the number of items recommended to a user that he/she likes and tp+fn represents the total items that a user likes
- If a user likes 5 items and the recommendation engine decided to show 3 of them, then the recall will be 0.6
- Larger the recall, better are the recommendations

Precision

$$\text{Precision} = \frac{tp}{tp + fp}$$

Out of all the recommended items, how many did the user actually like?

- Here tp represents the number of items recommended to a user that he/she likes and tp+fp represents the total items recommended to a user
- If 5 items were recommended to the user out of which he liked 4, then precision will be 0.8
- Larger the precision, better the recommendations
- But consider this case: If we simply recommend all the items, they will definitely cover the items which the user likes. So we have 100% recall! But think about precision for a second. If we recommend say 1000 items and user likes only 10 of them, then precision is 0.1%. This is really low. So, our aim should be to maximize both precision and recall.

Root Mean Square Error

It measures the error in the predicted ratings

$$RMSE = \sqrt{\frac{\sum_{i=1}^{N} (Predicted_i - Actual_i)^2}{N}}$$

- Here, Predicted is the rating predicted by model and Actual is original rating
- If a user has given a rating of 5 to a movie and we predicted the rating as 4, then RMSE is 1
- Lesser the RMSE value, better the recommendations

Drawback: No ordering of products recommended is maintained

Mean Reciprocal Rank

Evaluates the list of recommendations

$$\mathrm{MRR} = \frac{1}{\mathrm{n}} \cdot \sum_{i=1}^{\mathrm{n}} \frac{1}{\mathrm{r}\left(\mathrm{Q}_i\right)}$$

- Suppose we have recommended 3 movies to a user, say A, B, C in the given order, but the user only liked movie C. As the rank of movie C is 3, the reciprocal rank will be 1/3
- Larger the mean reciprocal rank, better the recommendations

Mean Average Precision

Precision and Recall don’t care about ordering in the recommendations

- Precision at cutoff k is the precision calculated by considering only the subset of your recommendations from rank 1 through k

$$MAP_i = \frac{1}{|R_i|} \sum_{k=1}^{|R_i|} P(R_i[k])$$

- Suppose we have made three recommendations [0, 1, 1]. Here 0 means the recommendation is not correct while 1 means that the recommendation is correct. Then the precision at k will be [0, 1/2, 2/3], and the average precision will be (1/3)*(0+1/2+2/3) = 0.38
- Larger the mean average precision, more correct recommendations

Exercise: Build a Text Recommender System

Building a Text Recommender

Create a Text Recommender

Dataset: **MovieLens 20M** dataset

Note: Please use **Kaggle Kernel** or **Google Colab** or **Your own local GPU machine** for this task. It will take days to complete if you use CPU :)

Further Reading

Practical Natural Language Processing

Soumya Vajjala, Anuj Gupta, Harshit Surana, Bodhisattwa Majumder