Recommender Systems

Subash Gandyer
Data Scientist, HealthChain

Agenda

- What is a Recommendation Engine?
- Why Recommendations?
- How does a Recommendation Engine work?
- Types of Recommendation
- Examples
    - Simple Recommender
    - Movie Recommender
    - Collaborative Filtering Recommender
    - Matrix Factorization Recommender
- Evaluation Metrics for a Recommendation Engine

Recommendation Engine

A recommendation engine filters the data using different algorithms and recommends the most relevant items to users. It first captures the past behaviour of a customer and based on that, recommends products which the users might be likely to buy.

Ways of recommending items to users

- 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

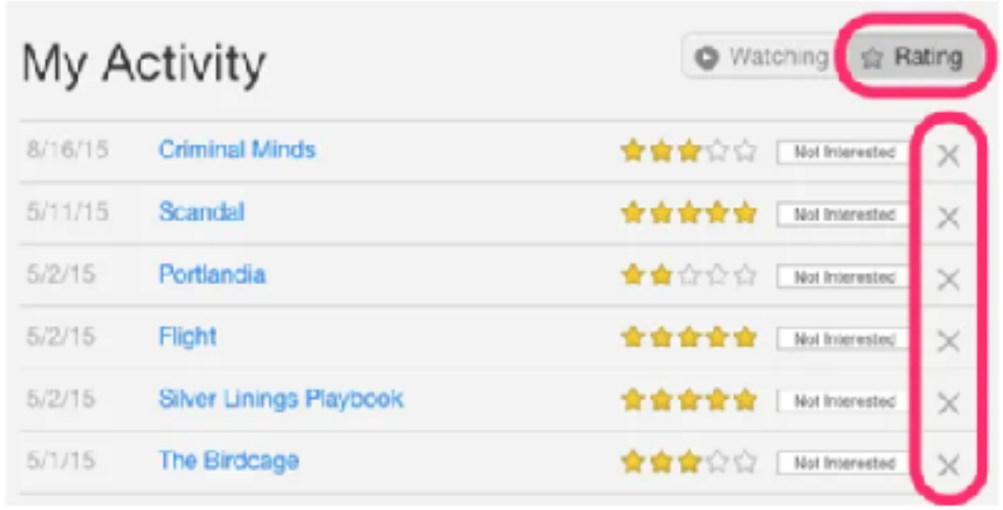
Drawback:

More computation time as number of users increases

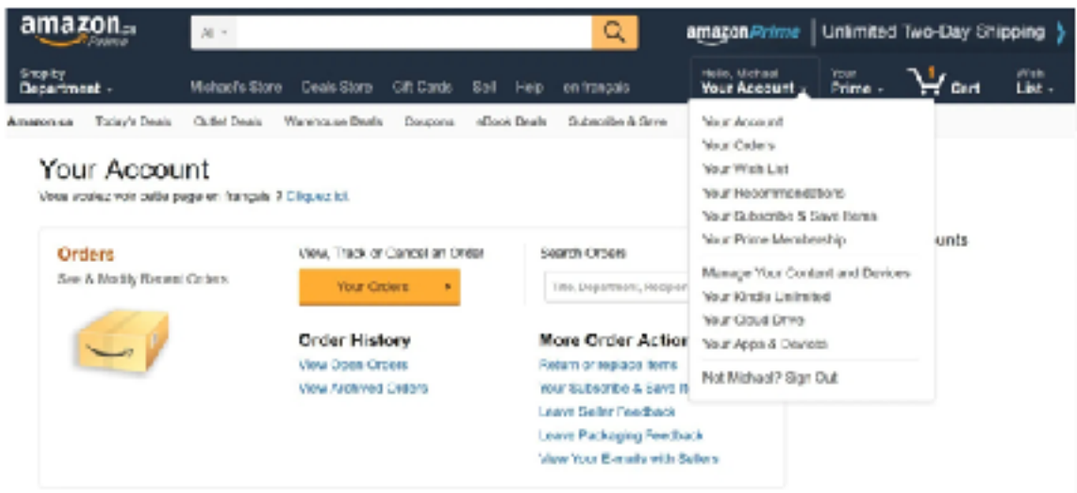
Cold Start Problem

Data Collection

- Explicit
  - Data provided by users explicitly like Product Ratings, Rankings, etc



- Implicit
  - Data like Clickstream, Search History, Order History, Website Navigation, Web Analytics and so on

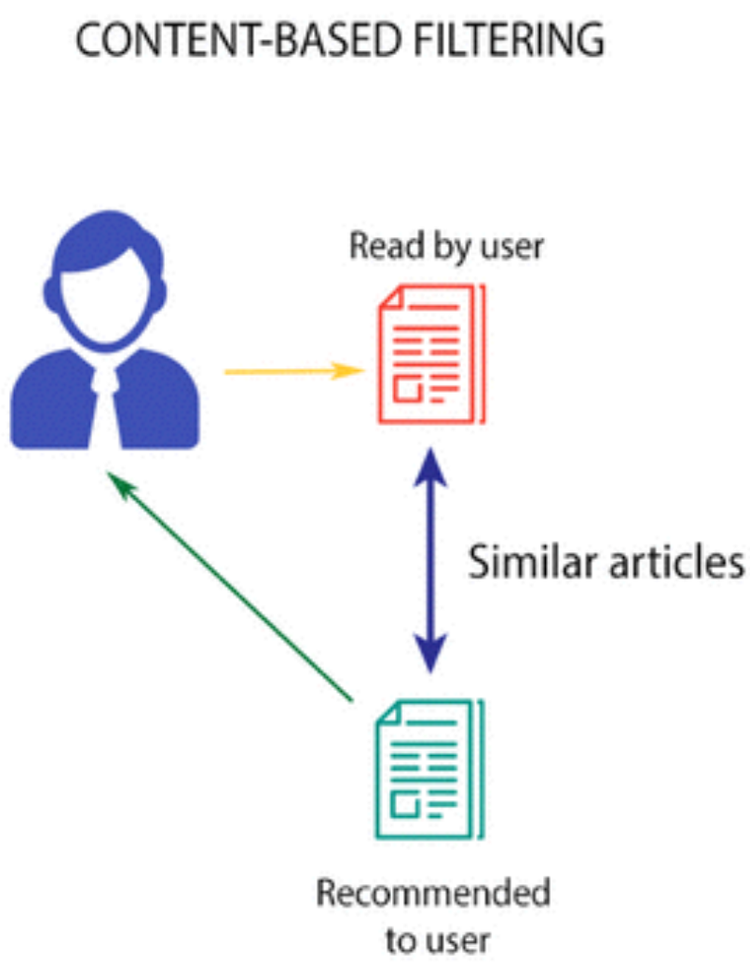


Types of Filtering

- Content-based Filtering
- Collaborative Filtering
- Hybrid

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 (2)

- How recommendations are done?
  - 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 (3)

- 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

Content based filtering (4)

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

Collaborative Recommender

Collaborative filtering

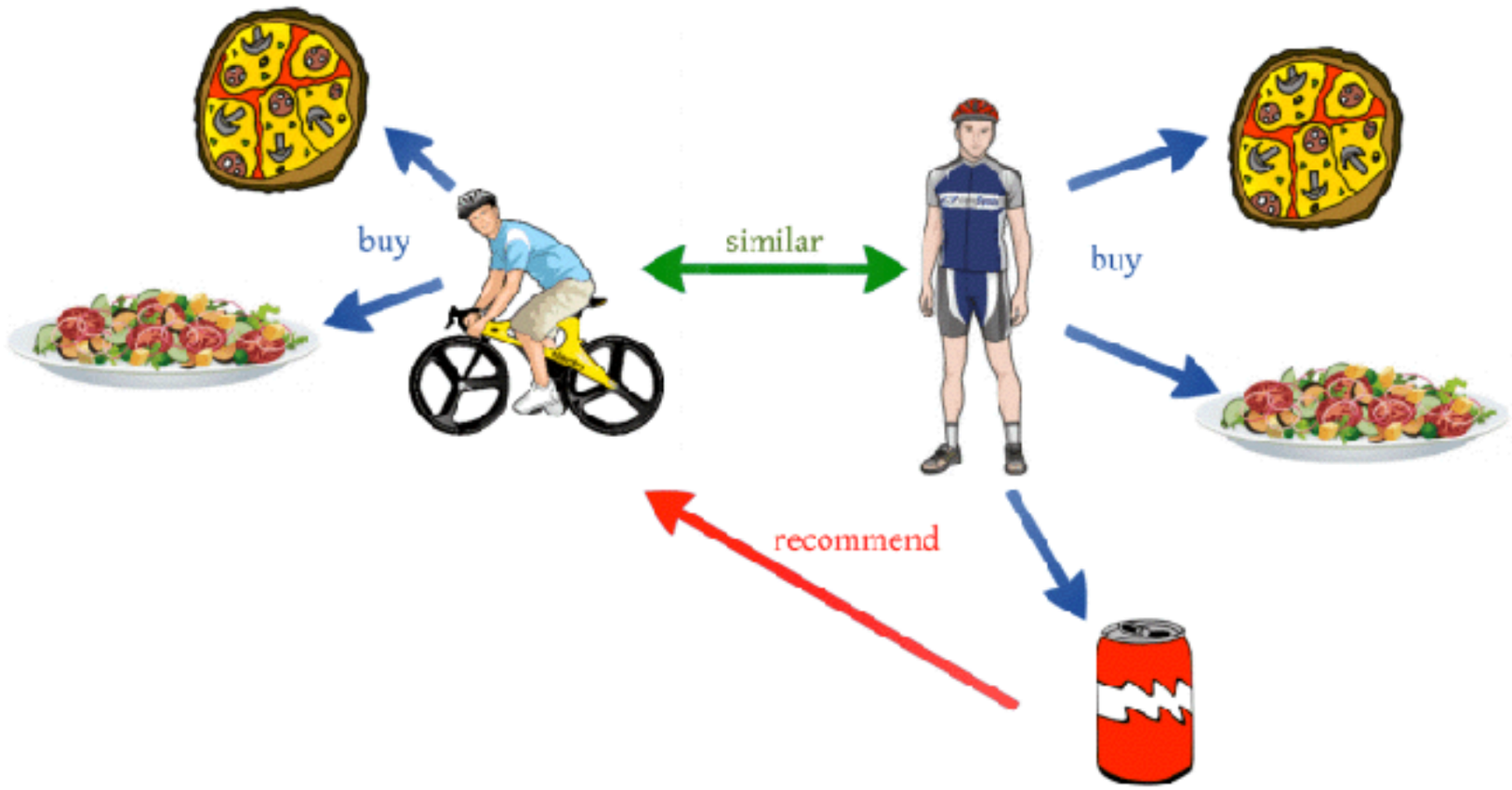
- Algorithm uses **user behaviour** for recommending items

- User-User Collaborative filtering

- Item-Item Collaborative filtering

User-User Collaborative

- 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 (2)

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 (3)

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 (4)

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

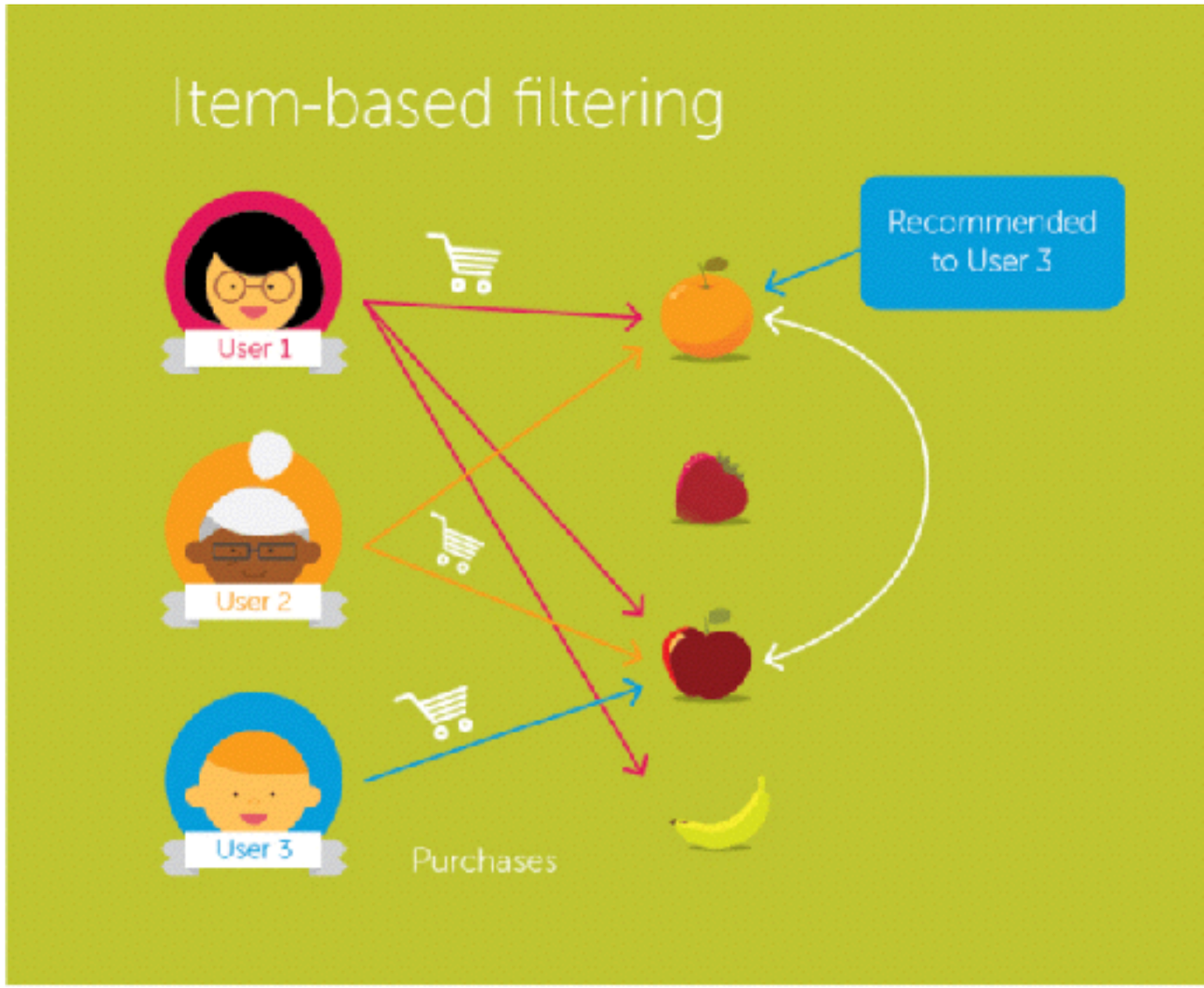
- Solution: (Select few users instead of all)
    - Select a threshold similarity and choose all users above that value
    - Randomly select the users
    - Arrange the neighbours in descending order of their similarity value and choose top-N users
    - Use clustering for choosing neighbours

Item-Item Collaborative

- 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 (2)

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 (3)

- Movie pairs x1 and x4 are highly correlated than x1 and x5 pair

- When a user searches for movie x1, x4 movie is recommended to him and vice versa

Cold Start problem

- 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

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 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 (2)

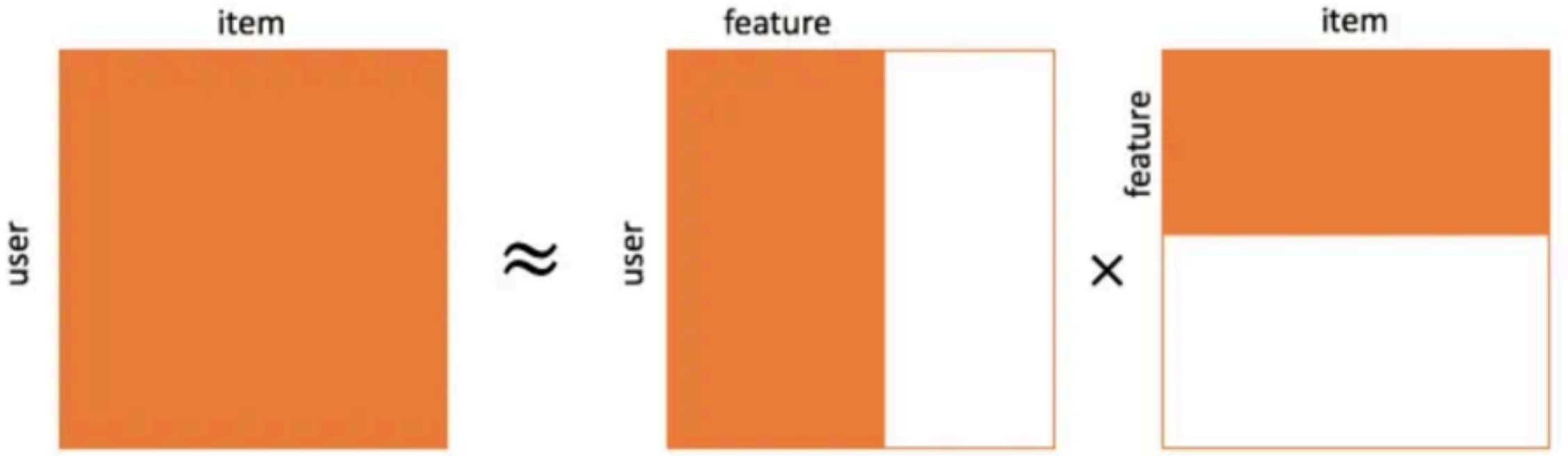
movie_id	1	2	3	4	5
user_id					
1	5.0	3.0	4.0	3.0	3.0
2	4.0	0.0	0.0	0.0	0.0
3	0.0	0.0	0.0	0.0	0.0
4	0.0	0.0	0.0	0.0	0.0
5	4.0	3.0	0.0	0.0	0.0

- User-Item matrix
  - 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 (3)

$$R=P\Sigma Q^T$$

- 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



Simple Recommender

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 (2)

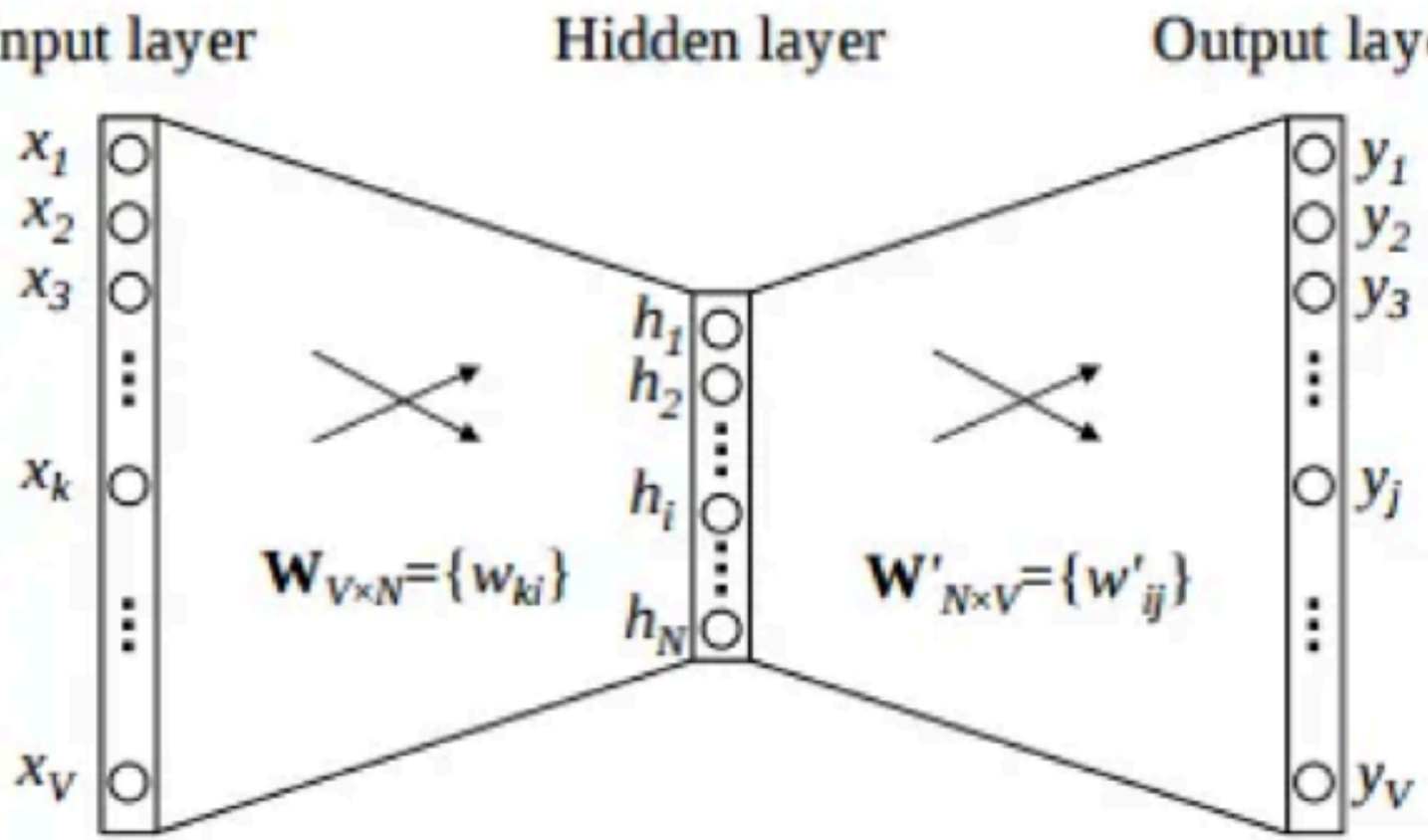
we must become what we wish to teach

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

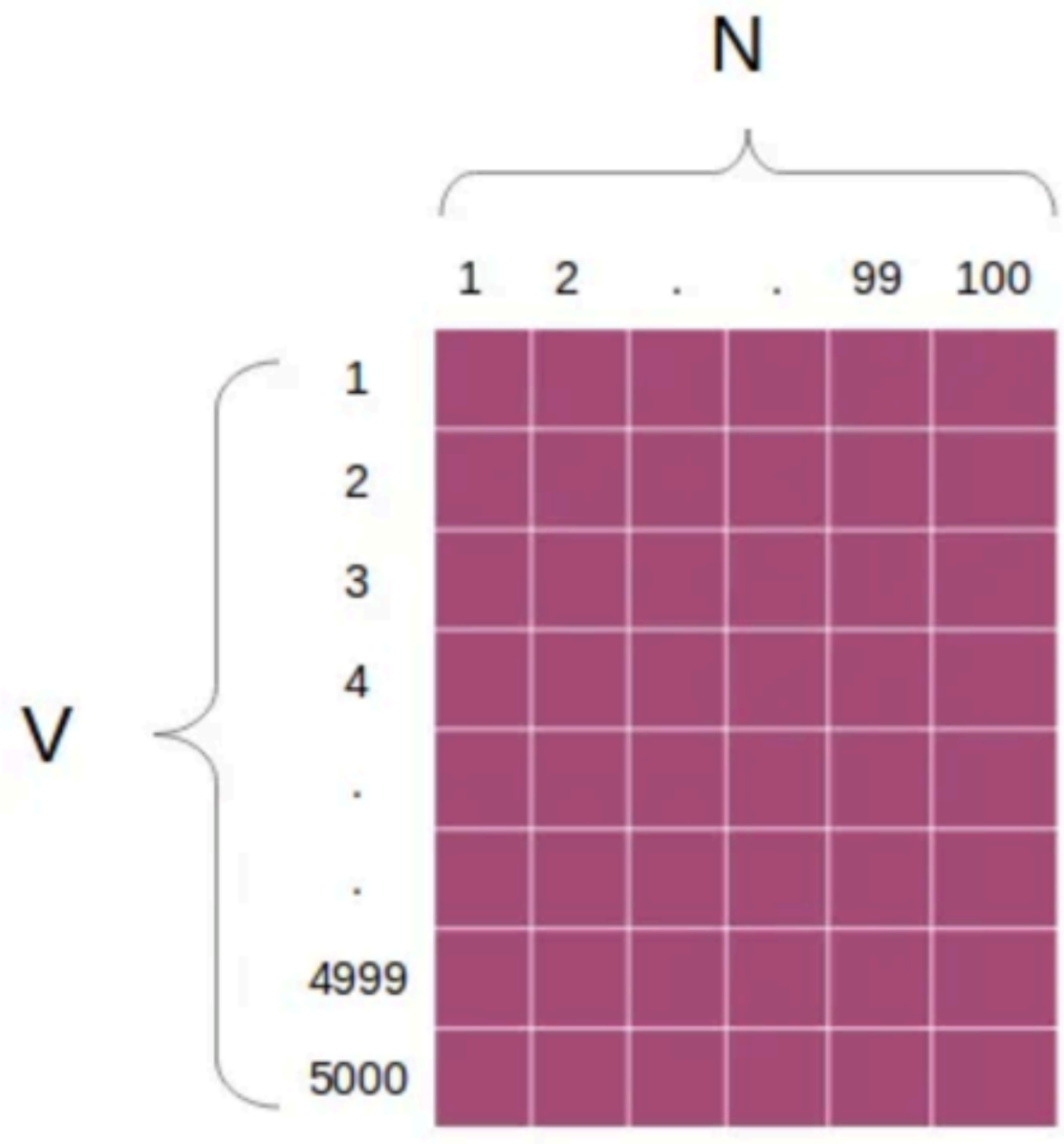
Architecture

- V = 5000 (Size of Vocabulary)
- N = 100 (No. Of hidden units / Length of word embeddings)

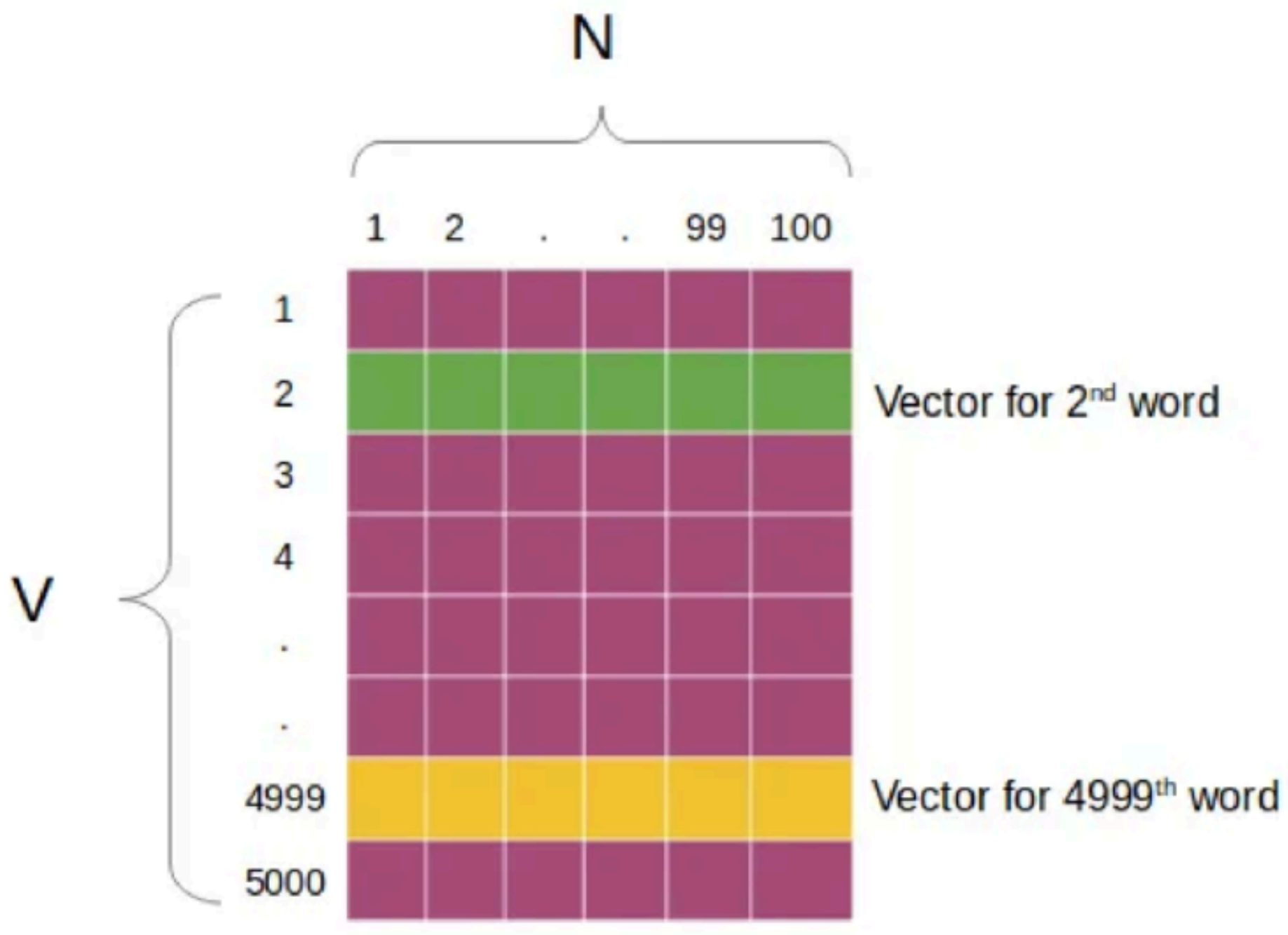


word2vec model architecture

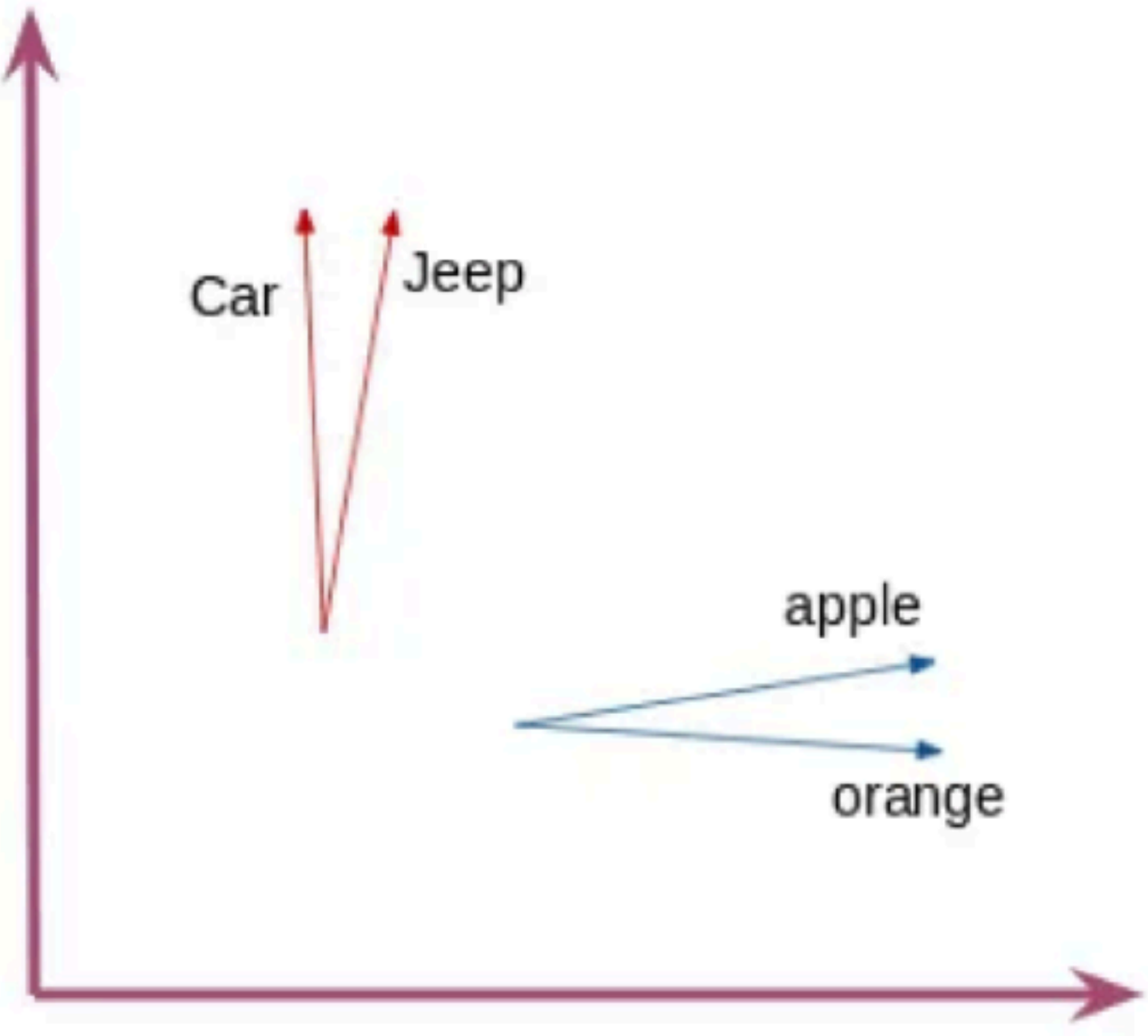
Weight Matrix (V x N)



W_{VxN} weight matrix


N
1
2
.
.
99
100
V
1
2
3
4
.
.
4999
5000
Vector for 2nd word
Vector for 4999th word

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/

Lets build a Simple Recommendation System

Simple_Recommendation_System.ipynb

Evaluation Metrics

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

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

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

  - 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 the model and Actual is the 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

$$\mathrm{MRR} = \frac{1}{n} \cdot \sum_{i=1}^{n} \frac{1}{r(Q_i)}$$

- 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 will be the recommendations