POLITECNICO
MILANO 1863

SCUOLA DI INGEGNERIA INDUSTRIALE
E DELL'INFORMAZIONE

CIE6006 - Data Analytics

Assignment 2

COMPUTER SCIENCE AND ENGINEERING - INGEGNERIA INFORMATICA

Author: Tito Nicola Drugman

Professor: Prof. Qingyu Li

TA: Weiwen Yuan, Junsheng Yao, Guanlin Wu

Academic year: 2025-2026

---

1. Introduction

This document is the report for the second assignment of the *Course CIE6006/MCE5918 - Data Analytics*.

2. Cosine Similarity and Matrix Derivation

The cosine similarity between two non-zero vectors $\boldsymbol{u}$ and $\boldsymbol{v}$ is a measure of the cosine of the angle between them. It is defined as:

$$\cos(\mathbf{u}, \mathbf{v}) = \frac{\mathbf{u} \cdot \mathbf{v}}{\|\mathbf{u}\|\|\mathbf{v}\|}$$

Where the numerator, $\mathbf{u} \cdot \mathbf{v}$, is the dot product of the two vectors and the denominator, $\|\mathbf{u}\|\|\mathbf{v}\|$, is the product of their Euclidean norms, which serves to normalize the dot product, making the final score independent of the vectors' lengths. The result is a score between -1 and 1, where 1 indicates that the vectors point in the exact same direction.

For this assignment is available an initial matrix $\boldsymbol{R}$ which is the user-item ratings matrix with dimensions $u \times s$, where $u = 9985$ users and $s = 563$ TV shows. Each element R_{ij} is binary, indicating whether a user has watched a show:

$$R_{ij} = \begin{cases} 1, & \text{if user } i \text{ watched show } j \\ 0, & \text{otherwise} \end{cases}$$

The matrix is sparse, containing 758'878 non-zero entries (ones) and 4'862'677 zero entries. From this ratings matrix $\boldsymbol{R}$, I derive two diagonal degree matrices which are crucial for normalization:

- **User degree matrix (P):** a diagonal matrix of size $u \times u$. Each first diagonal element P_{ii} represents the total number of shows watched by user i. Everything else is set to zero.
- **Item degree matrix (Q):** a diagonal matrix of size $s \times s$. Each first diagonal element Q_{jj} represents the total number of users who watched show j. Everything else is set to zero.

These degree matrices are then used to compute respectively the two primary similarity matrices for this project:

- **User similarity matrix ($\mathbf{S}_u$):** an $u \times u$ matrix where the element $(S_u)_{ij}$ stores the cosine similarity between user i and user j.
- **Item similarity matrix ($\mathbf{S}_i$):** an $s \times s$ matrix where the element $(S_i)_{ij}$ stores the cosine similarity between item i and item j.

2.1. Item Similarity Matrix (S_i) Derivation

In this subsection it is possible to see the reasoning and computations behind the cosine similarity formula expressed as:

$$\mathbf{S}_i = \mathbf{Q}^{-1/2}\mathbf{R}^T\mathbf{R}\mathbf{Q}^{-1/2}$$

that leads to the item similarity matrix S_i defined before in Section 2.
The dot product of the cosine similarity is the sum of the element-wise products of two vectors. As assessed before the matrix $\boldsymbol{R}$ is a binary matrix then all the columns (each of length u) will be binary, thus the dot product of two of those vectors will be equal to the number of users who have watched both show i and show j. To compute this value for all possible pairs of item simultaneously it is possible to use the matrix multiplication $\mathbf{R}^T\mathbf{R}$ which gives a matrix of size (s,s).
The denominator of the cosine similarity formula is the product of the Euclidean norms (magnitudes) of the vectors, $\|\mathbf{c}_i\|\|\mathbf{c}_j\|$. The Euclidean norm is equal to the square root of the sum of its squared elements. Since the elements are binary ($1^2 = 1$ and $0^2 = 0$), this computation can be simplified. The sum of the squared elements becomes identical to the count of non-zero entries in the vector.
For an item vector $\mathbf{c}_j$, this count represents the total number of users who have watched show j. This is precisely the definition of the item's degree, which is stored on the diagonal of the item degree matrix $\mathbf{Q}$ at position (j, j). Therefore, the norm of an item vector is:

$$\|\mathbf{c}_j\| = \sqrt{Q_{jj}}$$

The complete normalization term for the pair of items (i, j) is thus $\|\mathbf{c}_i\|\|\mathbf{c}_j\| = \sqrt{Q_{ii}}\sqrt{Q_{jj}}$.
The final step is to divide each element (i, j) of the dot product matrix ($\mathbf{R}^T\mathbf{R}$) by its corresponding normalization term. This element-wise division is achieved efficiently through multiplication with the diagonal normalization matrix, $\mathbf{Q}^{-1/2}$. This matrix is defined as having $1/\sqrt{Q_{kk}}$ on its diagonal for all non-zero Q_{kk}, and 0 otherwise.
The scaling is performed as follows:

- **Pre-multiplying** by $\mathbf{Q}^{-1/2}$ scales each row i of the dot product matrix by $1/\sqrt{Q_{ii}}$.
- **Post-multiplying** the result by $\mathbf{Q}^{-1/2}$ then scales each column j by $1/\sqrt{Q_{jj}}$.

This two-sided multiplication ensures that each element (i, j) is correctly normalized by the full term $\sqrt{Q_{ii}}\sqrt{Q_{jj}}$.

2.2. User Similarity Matrix (S_u) Derivation

The derivation for the user similarity matrix, $\mathbf{S}_u$, follows a parallel logic to the item similarity derivation defined in Subsection 2.1. We begin with the cosine similarity definition applied to two user (row) vectors from the ratings matrix $\mathbf{R}$, denoted as $\mathbf{r}_i$ and $\mathbf{r}_j$.
The dot product $\mathbf{r}_i \cdot \mathbf{r}_j$ represents the number of items that both user i and user j have watched. The complete $u \times u$ matrix of all pairwise dot products is computed by the matrix multiplication $\mathbf{R}\mathbf{R}^T$.
The norm of a user's vector, $\|\mathbf{r}_i\|$, simplifies to the square root of the number of items they have watched. This is the user's degree, which is stored on the diagonal of the user degree matrix $\mathbf{P}$ at position (i, i). Therefore, the norm of a user vector is:

$$\|\mathbf{r}_i\| = \sqrt{P_{ii}}$$

The complete normalization term for the user pair (i, j) is $\|\mathbf{r}_i\|\|\mathbf{r}_j\| = \sqrt{P_{ii}}\sqrt{P_{jj}}$.
Following the same normalization procedure, we use the diagonal matrix $\mathbf{P}^{-1/2}$ to scale the dot product matrix $\mathbf{R}\mathbf{R}^T$. This two-sided multiplication correctly normalizes each element (i, j) by the full term $\sqrt{P_{ii}}\sqrt{P_{jj}}$, leading to the final expression for the user similarity matrix:

$$\mathbf{S}_u = \mathbf{P}^{-1/2}\mathbf{R}\mathbf{R}^T\mathbf{P}^{-1/2}$$

2.3. Verification of Cosine Similarity Expressions

To empirically validate that the matrix-based approach yields results equivalent to the standard cosine similarity definition a direct comparison was performed. The similarity matrices were computed using both the derived formulas and a trusted optimized implementation from the `scikit-learn` library.
The absolute differences between the outputs of the two methods can be seen in Table 1.The results show that the numerical differences are negligible and on the order of machine floating-point precision. This confirms that the matrix

method is a numerically identical implementation of the standard cosine similarity calculation.

Table 1: Comparison of Matrix Formula vs. Standard Cosine Similarity

Metric	User Similarity ($\mathbf{S}_u$)	Item Similarity ($\mathbf{S}_i$)
Shape (Our Formula)	(9985, 9985)	(563, 563)
Shape (scikit-learn)	(9985, 9985)	(563, 563)
Maximum Absolute Difference	2.11×10^{-15}	2.66×10^{-15}

3. Collaborative Filtering Matrix Derivation

Once the similarity matrices $\mathbf{S}_u$ and $\mathbf{S}_i$ have been computed, they are used to generate a full matrix of recommendation scores $\mathbf{\Gamma}$ (of size $u \times s$). The element Γ_{ij} represents the predicted score for user i on item j.

3.1. User-User Filtering Derivation

The user-user approach predicts a user's preference for an item based on the preferences of similar users. The score for a user u on an item s, denoted Γ_{us}, is calculated as a weighted sum of other users' ratings for that item. The weights are the similarity scores between user u and all other users (x). This is expressed by the summation:

$$\Gamma_{us} = \sum_{x \in \text{users}} \text{cos\_sim}(u, x) \cdot R_{xs}$$

In this formula, $\text{cos\_sim}(u, x)$ is the element $(S_u)_{ux}$ from the user similarity matrix. To compute this score for a single user u across all possible items, we would perform a vector-matrix multiplication of user u's similarity vector (row u from $\mathbf{S}_u$) with the entire ratings matrix $\mathbf{R}$.
To generalize this calculation for all users at once, we can express it as a single matrix multiplication. The resulting recommendation matrix $\mathbf{\Gamma}$ is obtained by multiplying the complete user similarity matrix $\mathbf{S}_u$ by the ratings matrix $\mathbf{R}$:

$$\mathbf{\Gamma} = \mathbf{S}_u\mathbf{R}$$

3.2. Item-Item Filtering Derivation

The item-item approach works in a complementary fashion. It predicts a user's preference for an item based on its similarity to other items the user has already rated positively. The score for user u on item s is a weighted sum of that user's own ratings for other items (x), where the weights are the similarity scores between item s and those other items. This is expressed as:

$$\Gamma_{us} = \sum_{x \in \text{items}} R_{ux} \cdot \text{cos\_sim}(x, s)$$

Here, $\text{cos\_sim}(x, s)$ is the element $(S_i)_{xs}$ from the item similarity matrix. For a single user u, this score is computed with a vector-matrix multiplication of their rating vector (row u from $\mathbf{R}$) with the item similarity matrix $\mathbf{S}_i$.
To generate scores for all users and all items simultaneously, this process is generalized into the matrix expression:

$$\mathbf{\Gamma} = \mathbf{R}\mathbf{S}_i$$

3.3. Matrix-Level Expressions and Verification

The derivations above result in the following concise, matrix-level expressions for the complete recommendation score matrix, $\mathbf{\Gamma}$:

- **For User-User Filtering:** $\mathbf{\Gamma} = \mathbf{S}_u\mathbf{R}$
- **For Item-Item Filtering:** $\mathbf{\Gamma} = \mathbf{R}\mathbf{S}_i$

These expressions operate on the full matrices, avoiding the need for explicit element-wise summations and providing a more efficient computational model.

The assignment provides a hint to verify the item-item recommendation matrix as $\mathbf{\Gamma} = \mathbf{R}\mathbf{Q}^{-1/2}\mathbf{R}^T\mathbf{R}\mathbf{Q}^{-1/2}$. We can verify this by substituting our derived expression for the item similarity matrix, $\mathbf{S}_i$, into our recommendation formula.
Our derived expression for the item-item recommendation matrix is:

$$\mathbf{\Gamma} = \mathbf{R}\mathbf{S}_i$$

From Section 2.1, we derived the item similarity matrix as:

$$\mathbf{S}_i = \mathbf{Q}^{-1/2}\mathbf{R}^T\mathbf{R}\mathbf{Q}^{-1/2}$$

By substituting the expression for $\mathbf{S}_i$ into the formula for $\mathbf{\Gamma}$, we get:

$$\mathbf{\Gamma} = \mathbf{R}(\mathbf{Q}^{-1/2}\mathbf{R}^T\mathbf{R}\mathbf{Q}^{-1/2})$$

Assuming standard matrix multiplication associativity, this can be written as:

$$\mathbf{\Gamma} = \mathbf{R}\mathbf{Q}^{-1/2}\mathbf{R}^T\mathbf{R}\mathbf{Q}^{-1/2}$$

This confirms that our derived matrix-level expression for the item-item recommendation scores is identical to the fully expanded expression provided in the hint.

4. Implementation and Analysis

4.1. Implementation Approach and Numerical Considerations

During implementation, several key numerical considerations were accounted for:

- **Handling Zero Divisions:** The most critical numerical issue is the potential for division by zero when calculating the normalization matrices ($\mathbf{P}^{-1/2}$ and $\mathbf{Q}^{-1/2}$). This occurs if a user has watched zero shows or a show has been watched by zero users, resulting in a degree of zero. To prevent this, the implementation first extracts the diagonal from the degree matrix into a writable array (using `.copy()`), then it identifies any zero-valued degrees and replaces them with `1.0`[1].
- **Sparse Matrices:** While the dataset has a sparsity of 86.5%, standard dense NumPy arrays were used for the implementation. For the given scale (9985×563), NumPy's optimized C backend was in my opinion sufficiently performant. For significantly larger and sparser datasets, a next step in optimization would be to use a specialized sparse matrix library.

[1] Since $1/\sqrt{1.0} = 1.0$, this modification should prevents numerical errors without affecting the similarity scores for valid entries.

4.2. Insights and Translation of Theory to Practice

The two collaborative filtering methods were executed for the target user, Alex (ID 499), to generate Top-5 recommendations from a candidate pool of 100 shows. The results are presented below.

User-User Recommendations:

1. "FOX 28 News at 10pm" (Score: 908.48)
2. "Family Guy" (Score: 861.18)
3. "2009 NCAA Basketball Tournament" (Score: 827.60)
4. "NBC 4 at Eleven" (Score: 784.78)
5. "Two and a Half Men" (Score: 757.60)

Item-Item Recommendations:

1. "FOX 28 News at 10pm" (Score: 31.36)
2. "Family Guy" (Score: 30.00)
3. "NBC 4 at Eleven" (Score: 29.40)
4. "2009 NCAA Basketball Tournament" (Score: 29.23)
5. "Access Hollywood" (Score: 28.97)

The experiment yielded several key insights into the behavior of the two algorithms:

- **High Agreement:** the methods produced highly similar results, with a 4 out of 5 overlap in their recommendations. This strong agreement indicates that the top shows are robustly good recommendations, as they are both popular among users with tastes similar to Alex's and are also contextually similar to other shows Alex has enjoyed.
- **Different Score Scales:** an intersting difference is the scale of the scores. User-user scores are significantly larger because they represent an aggregation of similarity values from thousands of other users. In contrast, item-item scores are an aggregation of similarity values weighted only by Alex's own binary ratings, naturally resulting in a much smaller numerical scale.

5. Extra

To extend the analysis beyond the basic requirements I decided to implement a more rigorous quantitative evaluation of the existing models using precision and recall (Subsection 5.1) as well as a short a theoretical discussion of an alternative similarity metric called Pearson Correlation (see Subsection 5.2) and a 3D view of a portion of the data (see Subsection 5.3).

5.1. Quantitative Evaluation with Precision and Recall

While the initial sanity checks confirm the implementation's correctness, a more formal evaluation was conducted using standard information retrieval metrics. For this test defined in the assignment, "relevant" items are the shows within the hidden candidate pool (the first 100) that Alex had actually watched in the original dataset.

- **Precision@5** measures the accuracy of the recommendations: it can be considered answering to the question *of the 5 shows recommended, what fraction were relevant?* Precision@5 = $\frac{|\text{Relevant Items} \cap \text{Recommended Items}|}{5}$
- **Recall@5** measures the coverage: *of all relevant shows that existed, what fraction did we find?* Recall@5 = $\frac{|\text{Relevant Items} \cap \text{Recommended Items}|}{|\text{Relevant Items}|}$

These metrics were calculated for both collaborative filtering methods. The evaluation revealed a crucial insight about the dataset: the target user, Alex, had not watched any of the first 100 shows. Therefore, the set of "relevant items" for this test was empty. The "*results*" are summarized in Table 2.

Table 2: Precision and Recall @ 5 for Recommendation Models on User 499

Metric	User-User CF	Item-Item CF
Relevant Items in Test Set	0	0
Hits (out of 5)	0	0
Precision@5	0.00	0.00
Recall@5	0.00	0.00

The results in Table 2 are all zero. This is not an indication of poor model performance, but rather a consequence of the test data for this specific user. Since there were no positive examples to rediscover in the hold-out set, it was impossible to score any "hits." While the recommendation engines still produced valid predictions of shows Alex might like, these specific metrics are uninformative for this user. To get meaningful Precision and Recall scores, one would need to select a different user who had watched some shows in the 1-100 range.

Analysis of a different user (User 3). To obtain meaningful metrics, the experiment was repeated for User 3This allowed for a proper quantitative comparison, with the results summarized in Table 3.
For User 3, the Item-Item model demonstrated strong performance, achieving a Precision@5 of 0.40 and a Recall@5 of 0.67. This means it correctly "rediscovered" two of the three relevant shows. In stark contrast, the User-User model failed to find any of the relevant items, resulting

Table 3: Precision and Recall @ 5 for Recommendation Models on User 3

Metric	User-User CF		Item-Item CF	
Relevant Items in Test Set	3			
Hits (out of 5)	0		2	
Precision@5	0.00		0.40	
Recall@5	0.00		0.67	

in a score of zero for both metrics. This analysis highlights a key finding: the effectiveness of a collaborative filtering approach can be highly user-dependent. For this user, recommendations based on item similarity proved to be significantly more accurate than those based on the tastes of similar users.

5.2. Alternative Similarity Metric: Pearson Correlation

A limitation of Cosine Similarity in a ratings context is that it treats the absence of a rating (a '0' in our matrix) as a negative signal. It does not account for user-specific rating biases, such as "easy raters" who watch many shows versus "picky raters" who watch few.

A powerful alternative is the Pearson Correlation Coefficient which improves upon Cosine Similarity by first accounting for user rating behavior through mean-centering. The process involves adjusting the ratings matrix $\mathbf{R}$ before applying the similarity logic. For each user u, their mean rating $\bar{R}_u$ is calculated. A new matrix, $\mathbf{R}'$, is then created where each element is $R'_{uj} = R_{uj} - \bar{R}_u$. The Pearson Correlation is then mathematically equivalent to applying the Cosine Similarity formula to this mean-centered matrix, $\mathbf{R}'$.

By using this metric, the system should no longer be measuring the similarity of raw viewing history, but rather the similarity in how users' preferences *deviate from their own average behavior*. This would likely lead to more nuanced similarity scores, as it prioritizes agreement on items that are uniquely preferred by users, rather than on items that are universally popular. This provides a promising direction for future model improvement.

5.3. 3D visualization

Lastly, extend the analysis beyond the basic requirements, an interactive 3D visualization of the user and item similarity spaces was developed and hosted online[2]. This extension provides an intuitive way to explore the complex relationships discovered by the collaborative filtering models. A screenshot can be seen in Figure 1 Since high-dimensional similarity matrices ($\mathbf{S}_i$ is 563×563 and $\mathbf{S}_u$ is 9985×9985) are impossible to visualize directly. To project this data into a navigable 3D space, a dimensionality reduction technique was employed. The visualization includes several interactive features to explore the recommendation model:

- **User Selection:** a dropdown menu allows the selection of a specific user from the visualized sample.
- **Watched History:** upon selecting a user, yellow lines are drawn from the user's cube to the spheres of all the TV shows they have watched, visually representing their taste profile within the item space.
- **Recommendation Visualization:** for the target user Alex, the system also highlights his Top-5 recommended shows in bright green and draws distinct magenta lines to them. This provides an immediate, intuitive understanding of the recommendation output, showing where the recommended items lie in relation to the user and the items they have already seen.

[2] https://titonicoladrugman.github.io/TV-Show-Recommender/code/student/tv-show-visualizer/

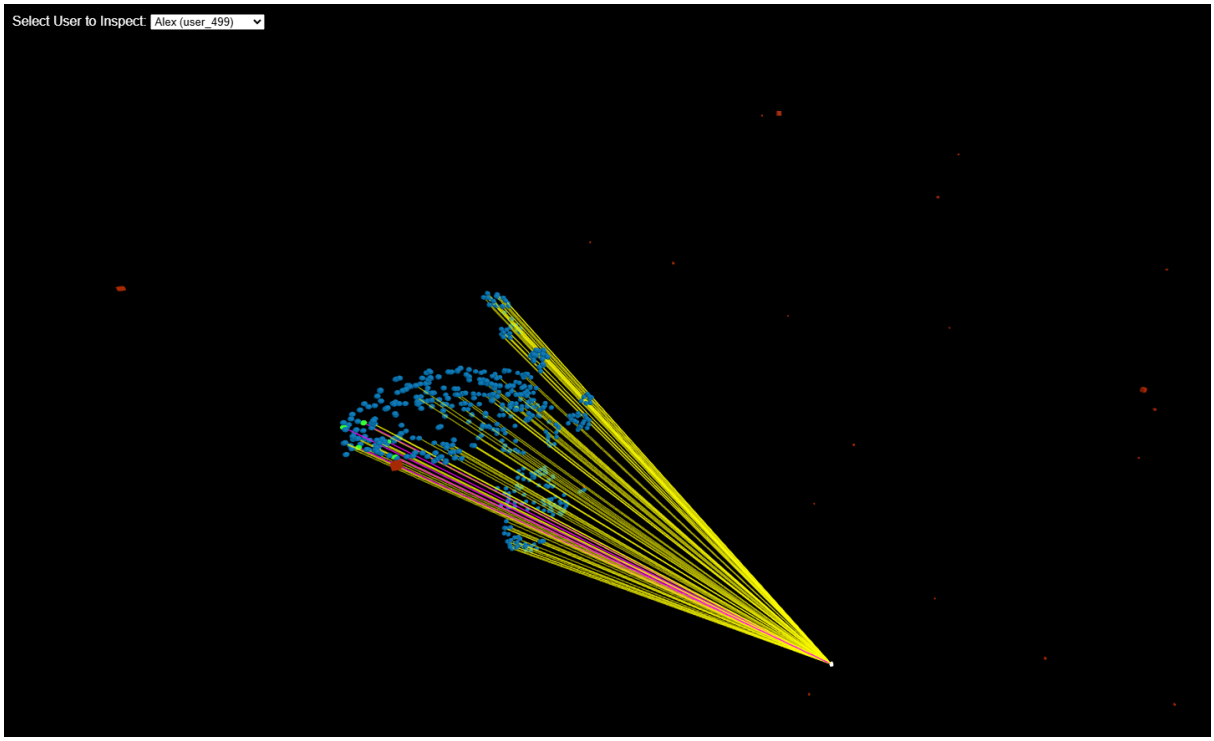

Figure 1: Example of the website.