Alternative Drug Recommendation Using Singular Value Decomposition (SVD)

Ray Owen Martin - 13524033[1,2]
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
[1]13524033@std.stei.itb.ac.id, [2]rayowenmartin2018@gmail.com

***Abstract*—Just like how drugs meant to be used for treating HIV/AIDS were used for treating COVID-19 in the past, it is important to know alternative drugs for a disease, which are rarely being thought to be similar to the commonly used drug for the disease, yet might still be a good substitute. One way to find this alternative drug is by using a recommender system. As the most common algorithm for recommender systems, Singular Value Decomposition (SVD) will be used for the project. The algorithm, with the aid of cosine similarity as the similarity metric, will run on a rating-based drug dataset, capturing the hidden correlation between drugs. This paper will present how the algorithm suggested can be applied neatly for the recommender system.**

***Keywords*—Alternative Drug, Linear Algebra, Recommendation System, Singular Value Decomposition, Truncated SVD**

I. Introduction

Drugs are one of the most underrated technologies people often take for granted. Not many people understand that before the discovery of penicillin, the world's first antibiotic, by Alexander Fleming in 1928, bacterial infections were as fatal as they could be, killing lots of lives. If not for penicilin, one of the most influential drugs, more people would've died unnecessarily.

The COVID-19 world pandemic became a case where we would appreciate the technology of drugs, since there were no cure. A study on February 22nd [2] revealed an anti-HIV drug, Nirmatrelvir, as a potential drug for curing COVID-19, due to its effectiveness against the corona virus. It is still a treatment for COVID-19 patients, though only for mild-to-moderate cases. This shows that an alternative drug, chosen from drugs that cure similar disease (as both COVID-19 and HIV compromises immune system), might just be the drug needed for a new or even existing disease.

It would be promising to have a system that can find an alternative, unknowingly similar drug for such cases, for cases of drug scarcity, or even for a disease with no known cure. The system might just find a similar behavior between drugs, hence become cues for researchers, medics or anyone interested to study further the alternative drug recommended.

II. Theoretical Framework

A. Matrices

Matrix is one of the most important concepts in Informatics and Computer Science, including Linear Algebra, as it represents information, whether in science, business or mathematics, organized into rows and columns, forming rectangular arrays. A 3×3 matrix, named matrix V, is usually represented as:

$$V = \begin{bmatrix} v_{11} & v_{12} & v_{13} \\ v_{21} & v_{22} & v_{23} \\ v_{31} & v_{32} & v_{33} \end{bmatrix}$$

An element in a matrix can be selected using their respective row and column. For example, to select the element in first row, second column of the matrix V, we can write it as v_{12}. This will be used for the latter concepts of matrix.

There are a few important matrix concepts, especially for this paper:

- Square matrix
  A square matrix is a matrix with the same amount of rows and columns. The 3×3 matrix above is an example. There will be some matrix operations that can only be operated if the matrix being operated on is a square matrix.

- Column matrix
  A column matrix is a matrix with only one column. It has n rows and 1 column, meaning it is a n×1 matrix.

- Row matrix
  A row matrix is a matrix with only one row. It has 1 row and n columns, meaning it is a 1×n matrix.

- Transpose
  Transpose is an operation that can be done on a matrix. It changes each row into column, column into row. If a matrix A is transposed, the result will be A^T. An example is shown below.

$$A = \begin{bmatrix} 2 & 4 & 8 \\ 0 & 3 & 4 \end{bmatrix} \rightarrow A^T = \begin{bmatrix} 2 & 0 \\ 4 & 3 \\ 8 & 4 \end{bmatrix}$$

Hence, if a row matrix is transposed, it would result in

a column matrix. Subsequently, transposing a column matrix will result in a row matrix.

- Primary diagonal and secondary diagonal
  A primary diagonal, also namingly main diagonal, consists of the elements v_{ii} of a matrix, where i is the row/column index. Hence, for a 3×3 matrix, the primary diagonal is (v_{11}, v_{22}, v_{33}).
  A secondary diagonal is the other diagonal of the same matrix, hence the secondary diagonal of the 3×3 matrix is (v_{31}, v_{22}, v_{13}).

- Diagonal matrix
  A diagonal matrix is a matrix with elements which are not the primary diagonal, are all zeroes (0). An example of 3×3 diagonal matrix D is provided below.

$$D = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 4 \end{bmatrix}$$

- Matrix Addition and Subtraction
  Addition and subtraction can be done to matrices. When a matrix B is added to or subtracted by a matrix C whereas matrix B and C are of the same size, then the elements in the fitting positions are added or subtracted. An example is provided below.

$$B = \begin{bmatrix} 2 & -5 \\ 6 & 4 \end{bmatrix};\ C = \begin{bmatrix} 0 & 5 \\ 9 & 2 \end{bmatrix}$$

$$B + C = \begin{bmatrix} 2+0 & -5+5 \\ 6+9 & 4+2 \end{bmatrix} = \begin{bmatrix} 2 & 0 \\ 15 & 6 \end{bmatrix}$$

$$B - C = \begin{bmatrix} 2-0 & -5-5 \\ 6-9 & 4-2 \end{bmatrix} = \begin{bmatrix} 2 & -10 \\ -3 & 2 \end{bmatrix}$$

- Matrix Multiplication
  Multiplication for two matrices, for example matrix A×B can be done when the number of columns in A is equal to the number of rows in B, resulting in a matrix of the size (number of rows in A) × (number of columns in B). The elements are calculated using the row-column rule, whereas:

$$(AB)_{ij} = a_{i1}b_{1j} + a_{i2}b_{2j} + \cdots + a_{ir}b_{rj}$$

with entry $(AB)_{ij}$ is in row i column j.

B. Vectors and Vector Spaces

Vectors are physical quantities that have value and direction. They can be symbolized with bold characters (e.g. **u**, **v**, **w**) or with arrows (e.g. $\vec{u}$, $\vec{v}$, $\vec{w}$), geometrically as directed lines.

A space where vectors are defined is called a vector space, also known as Euclidean space. Were there vectors in the Euclidean space R^n, they can be declared as:

$$v = (v_1, v_2, \ldots, v_n) \text{ or } v = \begin{bmatrix} v_1 \\ v_2 \\ \vdots \\ v_n \end{bmatrix}$$

The left notation is called comma-delimited form. Due to vectors being a list of components of a specific order, they can also be represented as matrices just like the right side, called as row-vector form.

The length of a vector v is known as norm v, symbolized with $\|v\|$. A norm v in the Euclidean space R^n can be computed as:

$$\|v\| = \sqrt{{v_1}^2 + {v_2}^2 + \cdots + {v_n}^2}$$

When a vector has norm 1, it is a unit vector, symbolized as $\hat{v}$. This is useful to specify a direction when length is irrelevant. This can be obtained from any vectors simply by dividing each elements of the vector with their norm, formulated as:

$$\hat{v} = \frac{1}{\|v\|}v \text{ or simply } \hat{v} = \frac{v}{\|v\|}$$

This process is called normalizing a vector v.

C. Orthogonal Vectors and Matrices

2 vectors are said to be orthogonal to each other if the dot product (summation of multiplication between 2 corresponding elements) equals to 0. Suppose we have 2 vectors as follows:

$$v_1 = \begin{bmatrix} 0 \\ 2 \\ -0.5 \end{bmatrix};\ v_2 = \begin{bmatrix} 9 \\ 1 \\ 4 \end{bmatrix}$$

The dot product between the two factors is:

$$v_1 \cdot v_2\ (0)(9) + (2)(1) + (-0.5)(4) = 0$$

Hence, we can say that the two factors are orthogonal.

When all the columns of a matrix is orthogonal to each other, it would then be called as an orthogonal matrix. An example is provided below.

$$V = \begin{bmatrix} 0 & 9 & {}^{17}/_{9} \\ 2 & 1 & -1 \\ -0.5 & 4 & -4 \end{bmatrix}$$

2 vectors are said to be orthonormal if and only if both vectors are unit vectors and both are orthogonal to each other. The same concept applies to an orthonomal matrix, whereas all columns is orthonormal to each other.

D. Eigenvalues and Eigenvectors

Eigenvectors can be defined as follows: a nonzero vector x in the vector space R^n which can act like a scalar towards a vector A in the same vector space. It can be formulated as follows:

$$Ax = \lambda x$$

whereas λ is the eigenvalue of A and x is said to be an eigenvector corresponding to λ.

The multiplication Ax when x is an eigenvector, does not change the direction of A, unlike the general case of x being another vector. Hence, an eigenvector is said to be special. To find eigenvectors and eigenvalues, we must first find the characteristic equation of A using the formula:

$$\det(\lambda I - A) = 0$$

and then solve for λ. The eigenvectors corresponding to an eigenvalue λ are the nonzero vectors that satisfy:

$$(\lambda I - A)x = 0$$

Note that the characteristic equation usually have few different solutions, in which case there will be multiple values of eigenvectors for the different solutions of λ which are the eigenvalues.

E. Singular Values

Singular values are simply the square roots of the eigenvalues sorted descending, symbolized as σ_i with i as index. They can be formulated as:

$$\sigma_i = \sqrt{\lambda_i}; i = 1, 2, \cdots, \text{n}$$

with

$$\lambda_1 \geq \lambda_2 \geq \lambda_2 \geq \cdots \geq \lambda_n$$

and n as the number of eigenvalues.

F. Singular Value Decomposition (SVD)

A matrix can be decomposed into several other matrices, meaning it equals to the multiplication of several other matrices. Singular value decomposition (SVD) serves as one of the methods. Suppose a $m \times n$ matrix A, SVD will decompose the matrix into:

$$A = U\Sigma V^T$$

with:

- U = an orthogonal $m \times m$ matrix, which consists of left singular vectors $u_1, u_2, \cdots, u_k$
- V = an orthogonal $n \times n$ matrix, which consists of right singular vectors ${v_1}^T, {v_2}^T, \cdots, {v_k}^T$
- Σ = a $m \times n$ diagonal matrix with the primary diagonal as the singular values of the matrix A

Note that the primary diagonal of a $m \times n$ matrix consists of v_{ii} with i being the smallest number between m and n.

One way to do SVD is by following these steps:

1. Calculate the right singular vectors ${v_1}^T, {v_2}^T, \cdots, {v_k}^T$ which correspond to the eigenvalues of A^TA. Normalize and transpose. These become row components of V^T. Rank(A) becomes k.
2. Calculate the singular values to make $\sigma_1, \sigma_2, \cdots, \sigma_k$ which becomes the primary diagonal of Σ. Σ has the size of $m \times n$ and is a diagonal matrix, so fill the values other than primary diagonal as 0.
3. For every nonzero singular values, calculate the left singular vectors $u_1, u_2, \cdots, u_k$ using the formula:
$$u_i = \frac{1}{\sigma_i} A v_i; i = 1, 2, \cdots, k$$
then normalize.
4. If $n > k$, which means there are singular values of zero, calculate $v_{k+1}, \cdots, v_n$ by finding eigenvectors of A^TA for $\lambda = 0$ and normalizing it.
5. If $m > k$, which means there are singular values of zero, calculate $u_{k+1}, \cdots, u_m$ by finding eigenvectors of AA^T for $\lambda = 0$ and normalizing it.
6. $A = U\Sigma V^T$

G. Truncated Singular Value Decomposition

Truncated SVD is an alternative method of SVD, where we can specify the number of columns as we want to, for example k, and the SVD results will return the top k vectors and values, written as:

$$A_k = U_k \Sigma_k V_k^T$$

The decomposition will just be the same as SVD with a difference in:

- U_k will take the first k left-singular vectors of A whilst keeping all the rows.
- Σ_k will be a $k \times k$ matrix with k largest singular values as its primary diagonal.
- V_k will take the first k right-singular vectors of A while still keeping all the rows.

H. Recommendation System with SVD

One common method for recommendation systems is latent factor model, which can use SVD (preferably truncated SVD) to extract correlation in a user-item matrix. It approximates the original matrix as a product of U (as user features), Σ (importance of features) and V^T (item features). Additionally, using truncated SVD will take only the most important features, reducing noise and dimentionality to identify latent patterns in user behavior and item characteristics.

I. Cosine Similarity

Cosine similarity is a measurement of how similar or close two non-zero vectors are by calculating the cosine of the actual angle between them. This metric is often used in text analysis and works marvellously for sparse matrices. The similarity is calculated with the formula as follows.

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

The result will be in range from -1 to 1, with -1 meaning the vectors are aligned but of opposite direction, 0 meaning the vectors are unrelated and 1 meaning the vectors are aligned in the same direction.

III. Proposed Method

The proposed method to make the alternative drug recommendation system is to use truncated SVD on a dataset. It will use Jupyter Notebook service to run algorithms in Python 3. The method can be divided into few parts as follows:

A. Requirements Installation

A few libraries including NumPy, Pandas and Scikit-Learn are used in order to complete the algorithm. They can be installed and imported using the code:

```
!pip install numpy
!pip install pandas
!pip install scikit-learn
import numpy as np
import pandas as pd
from sklearn.metrics.pairwise import cosine_similarity
```

Fig 1. Requirements installation and library importing code (Source: Author's Document)

Numpy will be used for the SVD operations since it has a few automated functions needed for matrices. Pandas will be used mainly for the dataFrame type data which holds the dataset altogether. Scikit-Learn's cosine_similarity will be used to calculate the similarity between drug scores.

B. Understanding the Dataset

The dataset is downloaded from [1], which has 17 columns and 2931 rows. Each row describes a drug with its 16 features, as the first feature is the name of the drug itself. However, due to the focus on the topic of alternative drug recommendation which relates to how drugs treat the same condition with the similar rating, only these few columns will be used:

- drug_name: the name of the drug
- medical_condition: the medical condition the drug is prescribed for
- rating: how well the drug treats the medical condition

Based on visual reading of the dataset, it is important to note that there are many NaN values in the column rating. This problem will be handled later.

C. Loading and Cleaning Dataset

In order to load the dataset into the program, the code as follows can be used.

```
# Create New Dataset (1)
# Make a dataframe with drug name as rows, medical condition as columns
# and average rating score of drug for the medical condition as elements
dataset = pd.read_csv("../data/drugs_side_effects_drugs_com.csv")

dataset.info()
cols = ["drug_name", "medical_condition", "rating"]
dataset_cleaned = dataset[cols].copy()
dataset_cleaned["rating"] = pd.to_numeric(dataset_cleaned["rating"], errors="coerce")
dataset_cleaned = dataset_cleaned.dropna(subset=['rating'])
dataset_cleaned.info()

dataset_rows = dataset_cleaned["drug_name"].unique() # Take row name
dataset_cols = dataset_cleaned["medical_condition"].unique() # Take column name
print(dataset_rows)
print(dataset_cols)

new_dataset = pd.DataFrame(0.0, index=dataset_rows, columns=dataset_cols)
# Initialize new dataframe with elements as 0
```

Fig 2. Loading and cleaning dataset code (1) (Source: Author's Document)

As the dataset is in .csv format, it is advised and common practice to use *pd.read_csv* and put it into the variable *dataset* as a DataFrame. It is possible to check the condition of *dataset* to make sure it is loaded well with *.info*. After that, the columns needed are specified, in this case, *drug_name*, *medical_condition* and *rating*, then a copy of the DataFrame of only the columns specified before is created. This becomes *dataset_cleaned*.

As rating should be filled with float data type yet there are many NaN values in the column, it is important to change the column into numeric values with *pd.to_numeric*. The rows with NaN values are deleted with *.dropna* to do computation well as intended. As this is just a small, simple model, there is no need to keep these rows. It is possible to recheck the results with *.info* and see that *dataset_cleaned* now has 1586 rows. By printing the columns, we can compute roughly how many unique conditions are in the dataset, in this case 47. This shows how limitted the dataset is and will impact the results later on.

Next, another DataFrame named *new_dataset* is created to change the format of the dataset to *drug_name* as the rows, *medical_condition* as the columns and 0.0 as the elements, which will then be filled accordingly. The code is as follows.

```
# Create New Dataset (2)

# Input the new dataframe with the values of drug for medical conditions in the row and column
def get_value(dname, mcond):
    for row in dataset_cleaned.itertuples():
        if (row.drug_name == dname and row.medical_condition == mcond):
            return row.rating
    return 0.0

for row in dataset_rows:
    for col in dataset_cols:
        new_dataset.at[row, col] = get_value(row, col)


new_dataset.describe()

# output = new_dataset
# output.to_csv('newdataset.csv', index=True) # to check new form of dataset
```

Fig 3. Loading and cleaning dataset code (2) (Source: Author's Document)

To visualize the results, this new DataFrame can be outputted in .csv format.

D. Applying Truncated SVD

The main algorithm for this project is truncated SVD. For the sake of modularity, it is best to make SVD

algorithm first, then followed by truncated SVD, both as functions.

```python
# SVD Operation
def createSVD(A):
    # A is a np.array

    AtA = np.dot(A.T, A)
    # dot: dot product, in this case, matrix multiplication; T = transposed array
    eigvals, eigvecs = np.linalg.eig(AtA) #Calculate eigen values and vectors
    ncols = np.argsort(eigvals) [::-1]
    #sorts the eigenvalues in descending order and stores the corresponding indices in ncols

    Vt = eigvecs[:,ncols].T
    # the same order as the sorted eigenvalues, dont forget to transpose

    sigma = np.sqrt(np.abs(eigvals[ncols]))

    U = np.dot(A, Vt.T) / (sigma + 10**(-10))
    # V is used rather than Vt, hence we retranspose Vt

    return U, sigma, Vt

#Truncated SVD Operation
def truncateSVD(A, k):
    U, sigma, Vt = createSVD(A)
    U_trunc = U[:, :k] # Take only the first k rows
    sigma_trunc = np.diag(sigma[:k]) # To make into diagonal matrix
    Vt_trunc = Vt[:k, :] # Take only the first k cols
    new_A = U_trunc @ sigma_trunc @ Vt_trunc
    return new_A
```

Fig 4. SVD and Truncated SVD code (Source: Author's Document)

The SVD operation is done with the help of Numpy library to enchance execution speed and assure highest accuracy. It returns all three U, Σ *and* V^T. Note that A must be a *np.ndarray* data type for the function to run as intended. Following the steps of SVD, we calculate A^TA first. This is a matrix multiplication operation, hence the use of *np.dot*. The result is another *np.ndarray* as expected.

The next step of SVD is to calculate eigenvalues of A^TA which we already calculated. This is done with *np.linalg.eig* which returns eigenvalues and eigenvectors respectively. The eigenvalues must then be sorted in a descending order.

Now, V^T can be created by inserting the eigenvectors which are sorted, then transpose the matrix. Σ can also be created at this step, by inserting the square root of the eigenvalues (as singular values). It is important to add *np.abs* to change negative eigenvalues into positive just in case they appear. For the sake of simplicity, Σ is not a diagonal matrix as of this step, it only contains the sorted singular values. U is inserted with $u_1, u_2, \cdots, u_k$ using the formula:

$$u_i = \frac{1}{\sigma_i} A v_i; i = 1, 2, \cdots, k$$

As the program focuses on simple implementations and the SVD operation will only be used in truncated SVD operation with guaranteed k being less than the size of the matrices related, 0 eigenvalues are ignored. To avoid division by zero, the code adds the σ_i in the equation with 10^{-10}. Note that this particular action will result in an imperfect SVD result, yet still enough for this project.

Subsequently, truncated SVD operation is just an addition to the SVD function. U_k is created by simply taking the first k rows. Σ_k is a diagonal matrix with the primary diagonal being Σ from the SVD result which are cut into only k elements. The diagonal matrix is calculated using *np.diag*. V_k^T is created by taking the first k columns.

```python
# Make the truncated matrix
dataset_array = new_dataset.to_numpy()
dataset_array_trunc = truncateSVD(dataset_array, 50)
```

Fig 5. Applying SVD and Truncated SVD code (Source: Author's Document)

With the functions done, the next step is just calling the functions. However, *new_dataset* is of a DataFrame data type, so it is required to change it into a *np.ndarray* as stated before. This is done with *.to_numpy*. The *truncateSVD* function can then be called.

E. Calculating Cosine Similarity

For the code to finally work, it is crucial to use a certain similarity metric. In this project, cosine similarity is chosen due to its ability of detecting correlation in sparse matrices. The code simply uses the function *cosine_similarity* we imported earlier from Scikit-Learn as shown below.

```python
# Find Cosine Similarity
item_similarity = cosine_similarity(pd.DataFrame(dataset_array_trunc))

item_sim_df = pd.DataFrame(item_similarity, index=new_dataset.index, columns=new_dataset.index)
# item_sim_df.to_csv('sim50.csv', index=True) # To check the similarity matrix
```

Fig 6. Calculating cosine similarity code (Source: Author's Document)

The result is again changed into a DataFrame type for further computation purposes.

F. Find the Alternative Drugs

Now that the code is ready, a driver is needed to call and compute the expected result. The driver code is shown below.

```python
def find_similar_drugs(drug_name, top_k=10):
    print("Finding drugs...")
    # Check if drug exists
    if drug_name not in item_sim_df.index:
        print(f"Error: Drug '{drug_name}' not found.")
        return None

    # Take similarity scores
    drug_scores_unfiltered = item_sim_df.loc[drug_name]
    drug_scores = drug_scores_unfiltered[drug_scores_unfiltered<0.99999]
    # If the similarity score is very high (near to 1), it means they are already used for similar purposes
    drug_scores = drug_scores[drug_scores_unfiltered > 0.1]
    # If the similarity score is very low (near to 0), it means they are completely different type of drugs

    # Sort
    sorted_drugs = drug_scores.sort_values(ascending=False).iloc[: top_k]

    # Show result
    if (sorted_drugs.empty):
        print(f"Alternative drugs for {drug_name} not found")
    else:
        print(f"Alternative drugs for {drug_name}")

    rank = 1
    for similar_drug, score in sorted_drugs.items():
        print(f"{rank}. {similar_drug} \t\t(Similarity: {score:.4f})")
        rank += 1


# To test
drug_name = input()
find_similar_drugs(drug_name)
# Try "erythromycin", "caffeine", "amoxicillin", "aspirin"
```

Fig 6. Finding alternative drug code (Source: Author's Document)

top_k is first specified to determine the maximum amount of drugs being recommended. For flexibility, it can be included as a parameter when calling the function to find the similar/alternative drugs, but when not included as a parameter, the code will still work.

The drug to be found the alternative is inputted first.The algorithm will then find the name of the drug first, if is not available in the dataset alongside its rating, the program will show an error. If it is available, the algorithm will try to find the alternative drugs with two conditions. One, the drug must not be too similar (in this code specified as the similarity score more than or equals to 0.99999), as it would mean that the drug is already being used for the

same diseases. Two, the drug must not be too different, (in this code specified as similarity score less than or equals to 0.1), as it would be not promising, showing the drug is unlikely to be an alternative. If there are no drugs that meet the conditions, it will mean that the system fails to find any meaningful, possible alternative drug. If there are, the results will be outputted.

IV. Results and Discussion

Since the code is not definitive and just outputs recommendation of drugs to be studied as alternative drugs, the accuracy of the algorithm is hard to calculate. For this, a qualitative, literature study from [18] and [19] is held. A few test cases that can be concluded as success are:

1. Erythromycin
   Input: erythromycin

```
Finding drugs...
Alternative drugs for erythromycin
1. doxycycline          (Similarity: 0.8000)
2. spironolactone               (Similarity: 0.8000)
3. minocycline          (Similarity: 0.8000)
4. Accutane             (Similarity: 0.8000)
5. Aldactone            (Similarity: 0.8000)
6. tretinoin            (Similarity: 0.8000)
7. adapalene            (Similarity: 0.8000)
8. isotretinoin                 (Similarity: 0.8000)
9. Bactrim              (Similarity: 0.8000)
10. Retin-A             (Similarity: 0.8000)
```

   Fig 7. Results for test case 1 (Source: Author's Document)

   Literature study shows:
   - Erythromycin, doxycycline, minocycline, bactrim are antibiotics to treat bacterial infections.
   - Spironolactone and aldactone are for high blood pressure.
   - Accutane, tretinoin, adapalene, isotretinoin, retin-A are for acne.

   It is safe to conclude that since 3 of the 10 drugs are in the similar group, the program works moderately fine.

2. Caffeine
   Input: caffeine

```
Finding drugs...
Alternative drugs for caffeine
1. ciclesonide          (Similarity: 0.7380)
2. triamcinolone                (Similarity: 0.7096)
```

   Fig 8. Results for test case 2 (Source: Author's Document)

   Literature study shows:
   - Caffeine can be used to treat sleep apnea, but its particular type, caffeine citrate is used to treat neonatal apnea (breathing problems)
   - Ciclesonide is used to treat nasal symptoms of allergy
   - Triamcinolone is used to treat breathing disorders

   It is safe to conclude both drugs are in the similar group, which treat nasal symptoms, either breathing problems or nasal symptoms of allergy, hence the program works well for this test case.

3. Amoxicillin
   Input: amoxicillin

```
Finding drugs...
Alternative drugs for amoxicillin
1. beclomethasone               (Similarity: 0.6549)
2. erythromycin                 (Similarity: 0.6000)
```

   Fig 9. Results for test case 3 (Source: Author's Document)

   Literature study shows:
   - Amoxicillin and erythromycin can be used to treat bacterial infections
   - Beclomethasone, as cream can be used to treat skin conditions, while as inhaler can be used to prevent asthma attack

   It is safe to conclude one of the two alternative drugs found works in a similar group, meaning there are associations between the amoxicillin and erythromycin, hence the algorithm works okay here.

4. Claritin
   Input: Claritin

```
Finding drugs...
Alternative drugs for Claritin
1. beclomethasone               (Similarity: 0.7557)
2. budesonide           (Similarity: 0.7375)
3. cromolyn             (Similarity: 0.7311)
4. ciclesonide          (Similarity: 0.6748)
5. mometasone           (Similarity: 0.5417)
6. triamcinolone                (Similarity: 0.4573)
```

   Fig 10. Results for test case 4 (Source: Author's Document)

   Literature study shows:
   - Claritin can be used to treat allergy symptoms like sneezing, runny nose, watery eyes, skin rash
   - Beclomethasone, as cream can be used to treat skin conditions, while as inhaler can be used to prevent asthma attack
   - Budesonide can be used as inhalers
   - Cromolyn and mometasone can be used to treat itchy skin
   - Ciclesonide is used to treat nasal symptoms of allergy
   - Triamcinolone is used to treat breathing disorders

   Of all the alternative drugs recommended, they can be used to treat either nasal problems or skin conditions, rash or itch. It is safe to conclude the two alternative drugs found works in a similar group. Hence, it can be concluded that the algorithm works moderately fine here.

Despite the 4 test cases provided, there will be many cases where the system does not provide any alternative drug recommendation. This can mean that the dataset is not complete enough, the drugs do rarely have substitutes or

alternatives, or most possibly due to the small amount of unique conditions (47) relative to the number of drugs (more than 1500).

V. Conclusion

Drugs are one of the most important technologies of mankind. When combined with matrix operations as one of the most impactful mathematics applications, it is possible to further enchance the power and benefits of drugs. After applying truncated SVD towards a cleaned dataset of drugs, computing the similarity and correlation between drugs with cosine similarity, a small, simple alternative drug recommender system is created. This is meant to help find cues and clues of "emergency" drug substitutes and alternatives. However, it is important to note that this system cannot be 100% accurate. The system will find associations, which does not point to causality directly. Further exploration, research and study will be needed to improve this system. More accurate and complete dataset will provide the base, stronger and more suitable algorithm will further the accuracy, while more accurate metrics will help tune the algorithm even better.

VI. Appendix

The github repository created for this project can be accessed on https://github.com/Tensai-033/Alternative-Drug-Recommendation-Using-SVD.

VII. Acknowledgment

The author is extremely grateful to God for all the guidance during the process of learning leading to the competion of this paper. The author would as well thank his lecturer of Linear Algebra and Geometrics IF2123, Dr. Rinaldi Munir, Rila Mandala, Ph.D., Dr. Arrival Dwi Sentosa and Kridanto Surendro, Ph.D., for sharing their knowledge and for their patience throughout the whole lecture. Furthermore, the author treasures the constant support from his loved ones, family and friends, which undoubtedly complements his efforts.

References

[1] Hernandez, Daniela, "Penicillin: 83 Years Ago Today", [Online]. Available: https://www.publichealth.columbia.edu/research/center-infection-and-immunity/penicillin-83-years-ago-today [Accessed 24-Dec-2025]

[2] Hung et al., "Oral Nirmatrelvir/Ritonavir Therapy for COVID-19: The Dawn in the Dark?" Antibiotics, vol. 11, no. 2, 1 Feb. 2022, p. 220, https://doi.org/10.3390/antibiotics11020220 [Accessed 24-Dec-2025]

[3] Mugwagwa, Tendai, et al. "Nirmatrelvir/Ritonavir Treatment for COVID-19: An Economic Value Systematic Literature Review." Journal of Medical Economics, Winter 2025, pp. 1–25, https://doi.org/10.1080/13696998.2025.2579407. [Accessed 24-Dec-2025]

[4] Anton, Howard. Elementary Linear Algebra. John Wiley & Sons, 19 Nov. 2018.

[5] GeeksforGeeks, "Singular Value Decomposition (SVD)", [Online]. Available: https://www.geeksforgeeks.org/machine-learning/singular-value-decomposition-svd/ [Accessed 13-Dec-2025]

[6] GeeksforGeeks, "SVD in Recommendation Systems", [Online]. Available: https://www.geeksforgeeks.org/machine-learning/svd-in-recommendation-systems/ [Accessed 13-Dec-2025]

[7] GeeksforGeeks, "Singular Value Decomposition (SVD)", [Online]. Available: https://www.geeksforgeeks.org/machine-learning/singular-value-decomposition-svd/ [Accessed 13-Dec-2025]

[8] Gupta, Ritik, "How Singular Value Decomposition (SVD) is used in Recommendation Systems, "Clearly Explained".", [Online]. Available: https://medium.com/@ritik_gupta/how-singular-value-decomposition-svd-is-used-in-recommendation-systems-clearly-explained-201b24e175db [Accessed 16-Dec-2025]

[9] https://numpy.org/doc/1.26/reference/routines.linalg.html

[10] Yehoshua, Roi, "Singular Value Decomposition (SVD), Demystified", [Online]. Available: https://towardsdatascience.com/singular-value-decomposition-svd-demystified-57fc44b802a0/ [Accessed 19-Dec-2025]

[11] Munir, Rinaldi, "Vektor di Ruang Euclidean (bagian 1)", [Online]. Available: https://informatika.stei.itb.ac.id/~rinaldi.munir/AljabarGeometri/2025-2026/Algeo-11-Vektor-di-Ruang-Euclidean-Bag1-2025.pdf [Accessed 20-Dec-2025]

[12] Munir, Rinaldi, "Vektor di Ruang Euclidean (Bagian 2)", [Online]. Available: https://informatika.stei.itb.ac.id/~rinaldi.munir/AljabarGeometri/2025-2026/Algeo-12-Vektor-di-Ruang-Euclidean-Bag2-2025.pdf [Accessed 20-Dec-2025]

[13] Munir, Rinaldi, "Nilai Eigen dan Vektor Eigen (Bagian 1)", [Online]. Available: https://informatika.stei.itb.ac.id/~rinaldi.munir/AljabarGeometri/2025-2026/Algeo-19-Nilai-Eigen-dan-Vektor-Eigen-Bagian1-2025.pdf [Accessed 20-Dec-2025]

[14] Munir, Rinaldi, "Singular Value Decomposition (SVD) (Bagian 1)", [Online]. Available: https://informatika.stei.itb.ac.id/~rinaldi.munir/AljabarGeometri/2025-2026/Algeo-21-Singular-value-decomposition-Bagian1-2025.pdf [Accessed 20-Dec-2025]

[15] Munir, Rinaldi, "Singular Value Decomposition (SVD) (Bagian 2)", [Online]. Available: https://informatika.stei.itb.ac.id/~rinaldi.munir/AljabarGeometri/2025-2026/Algeo-22-Singular-value-decomposition-Bagian2-2025.pdf [Accessed 20-Dec-2025]

[16] Chen, Denise, "Recommender System — singular value decomposition (SVD) & truncated SVD" [Online]. Available: https://medium.com/data-science/recommender-system-singular-value-decomposition-svd-truncated-svd-97096338f361 [Accessed 23-Dec-2025]

[17] Krantz, Tom, and Alexandra Jonker, "Cosine Similarity" [Online]. Available: https://www.ibm.com/think/topics/cosine-similarity [Accessed 24-Dec-2025]

[18] www.nhs.uk/medicines/

[19] https://www.drugs.com/

Pernyataan

Dengan ini saya menyatakan bahwa makalah yang saya tulis ini adalah tulisan saya sendiri, bukan saduran, atau terjemahan dari makalah orang lain, dan bukan plagiasi.

Bandung, 24 December 2025

Ray Owen Martin
13524033