MINIPROJECT-1

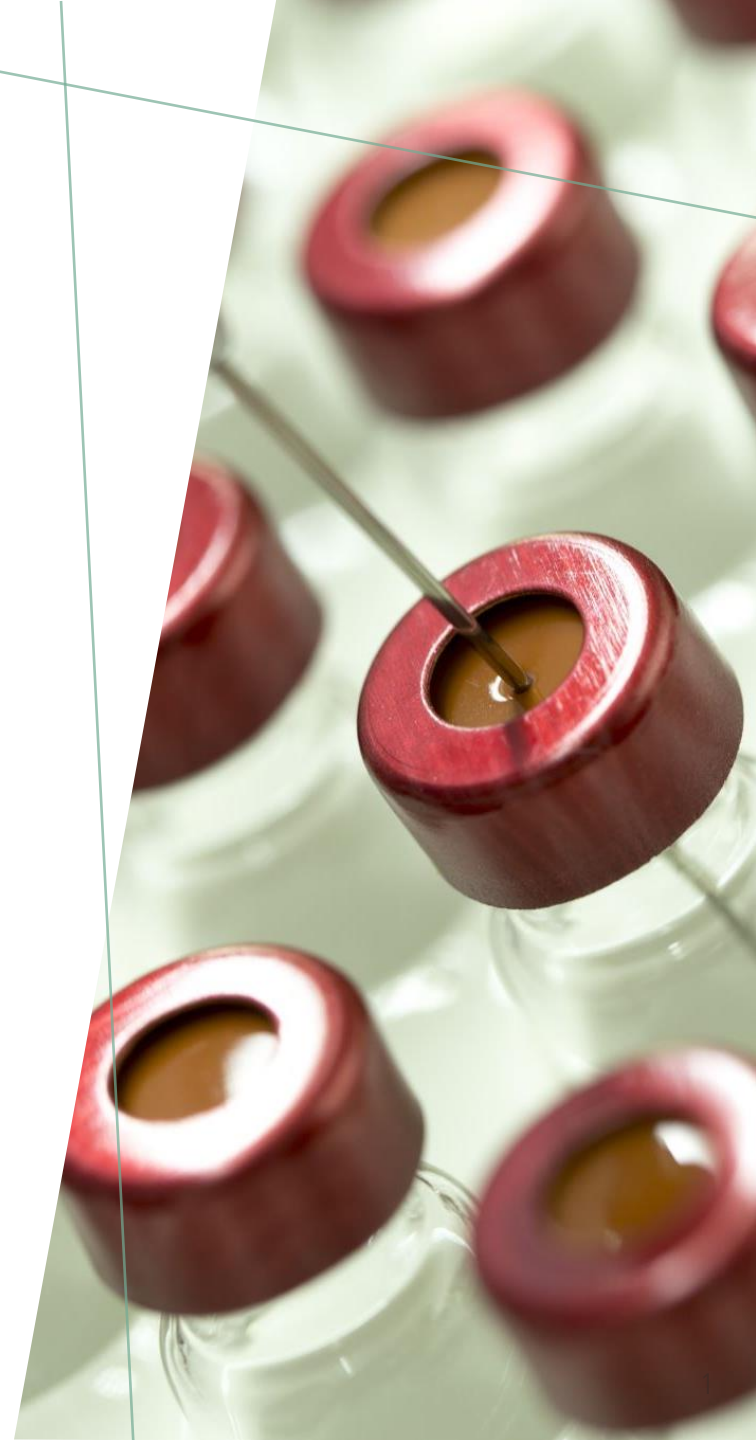
PRESENTATION

SC6613(51) RECOMMENDER SYSTEM

PRESENTED BY

0615971 MIN HEIN KHANT

2024-08-17



MAIN IDEA OF THE PROJECT

- The main idea of Mini Project 1 is to develop and enhance a movie recommendation system using different algorithms across three parts.
- Each part progressively builds on the previous one, improving the recommendation process by refining user profiles and the methods used to compare them with movie data.

PART I - SIMPLE RECOMMENDATION ALGORITHM

1. **Loading Data**:
    1. Import user data from movie.users.txt.
    2. Load movie data from movie.metadata_plot_summary_test.txt.
2. **Creating User Profiles Using Simple Unary**:
    1. Analyze genres of movies each user has watched.
    2. Calculate the normalized frequency of each genre to create user profiles.
3. **Calculating Similarity**:
    1. Compute cosine similarity between user profiles and not-yet-seen movies.
    2. Measure how similar movies are to a user's preferences.
4. **Displaying Recommendations**:
    1. List the top 10 movies with the highest similarity for each user.

IMPORTANT IDEAS FOR THE PART-1

SIMPLE UNARY

Simple Unary refers to a basic method for creating user profiles in a recommendation system by counting the frequency of different features—in this case, movie genres—that a user has interacted with.

Here's how it works:

1. **Frequency Counting**: For each user, you count how many times they have watched movies of each genre. For example, if a user has watched 5 action movies, 3 comedies, and 2 dramas, their genre counts would be [Action: 5, Comedy: 3, Drama: 2].

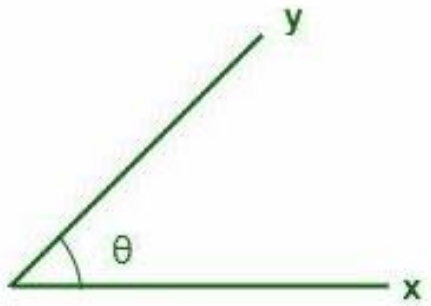
2. **Normalization**: To create a profile that's comparable across users, you normalize these counts. This means converting the raw counts into proportions or percentages. For example, if the total count of movies watched by the user is 10, the normalized profile would be [Action: 0.5, Comedy: 0.3, Drama: 0.2]. This step ensures that the profile reflects the user's preference distribution rather than just the absolute number of movies watched.

3. **Profile Creation**: The resulting normalized values create a vector (a list of numbers) representing the user's preferences across all available genres. This vector is the user's profile and is used to compare with other users or items (like movies) to determine similarity.

EXAMPLE OF SIMPLE UNARY

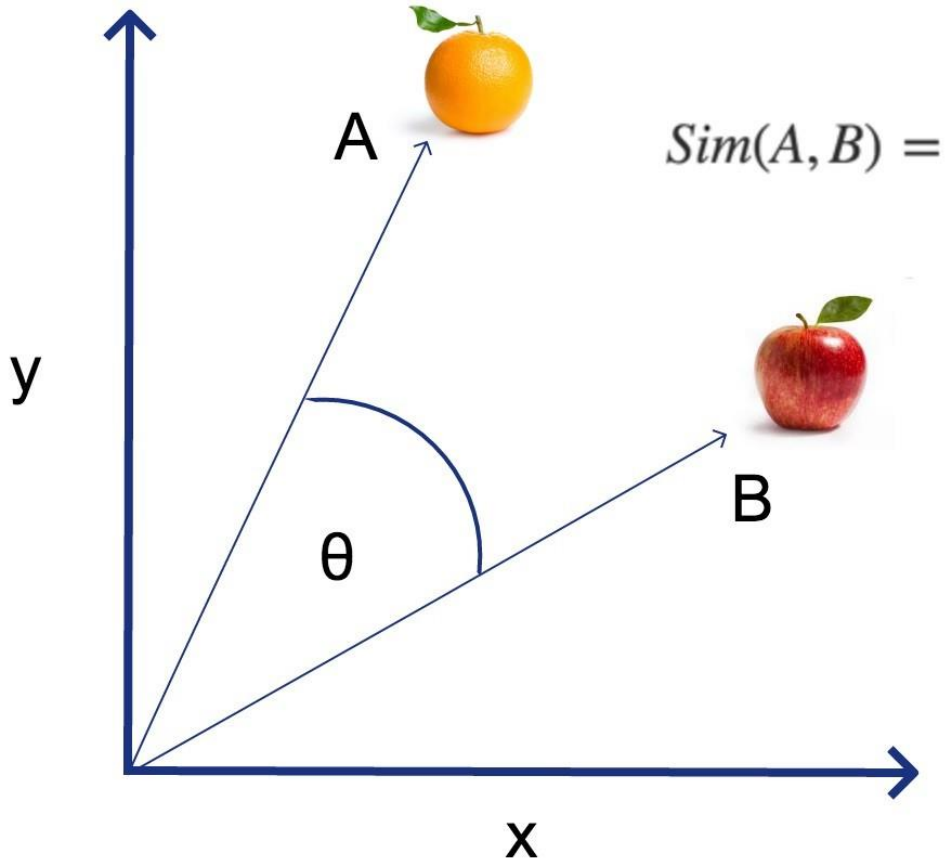
User	Action	Comedy	Drama	Thriller
User 1	1	1	0	0
User 2	0	0	1	1
User 3	1	0	1	0

COSINE SIMILARITY



- **Cosine Similarity** is a measure that determines how similar two vectors are by comparing their directions. In recommendation systems, it’s used to assess how closely a user's preferences match with a movie's attributes. The result is a value between -1 and 1, where 1 indicates perfect similarity, 0 indicates no similarity, and -1 indicates complete dissimilarity.

Cosine Similarity



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

SCAN TO SEE THE PART-1

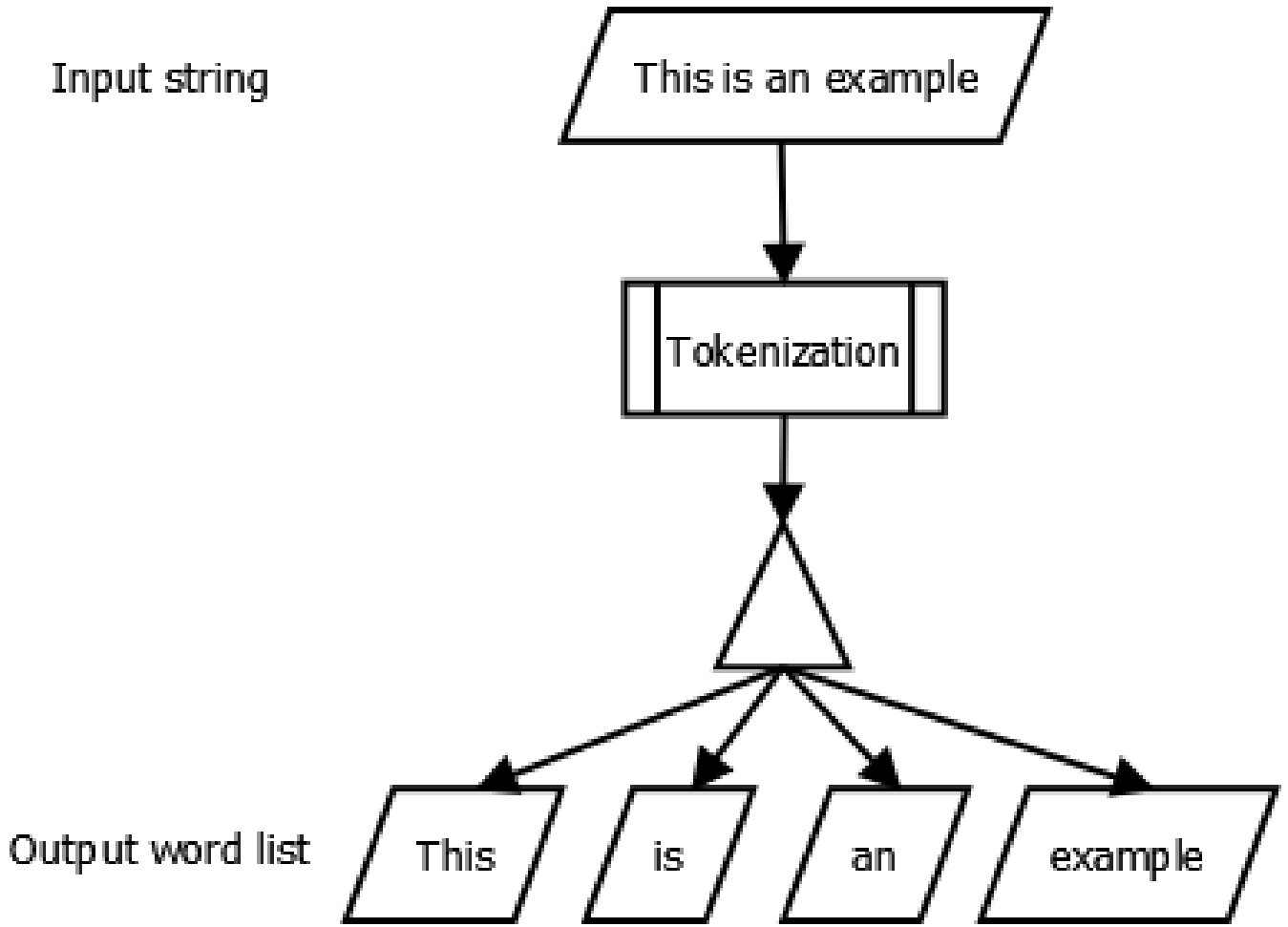
PART II: A SIMPLE CONTENT-BASED FILTERING ALGORITHM

1. **Load Data:**
   - •Load the user profiles from movie.users.txt.
   - •Load the movie metadata (plot summaries) from movie.metadata_plot_summary_test.txt.
2. **Preprocess Text Data:**
   - •Tokenize and normalize the text data.
   - •Compute TF-IDF for both user profiles and movie metadata.
3. **Normalize TF-IDF:**
   - •Normalize the TF-IDF vectors for user profiles.
4. **Compute Cosine Similarity:**
   - •Calculate the cosine similarity between each user profile and the not-yet-seen movies.
5. **Display Results:**
   - •For each user, display the top 10 movies based on similarity.

IMPORTANT IDEAS FOR THE PART-2

TOKENIZE

- **Definition**: Tokenization breaks text into smaller units called tokens, such as words or phrases.
- **Example**: "I love programming" is tokenized into ["I", "love", "programming"].
- **Purpose**: It simplifies and prepares text for further analysis, making it easier to handle in natural language processing tasks.
- **Process**:
  - **Input**: Begin with a block of text.
  - **Splitting**: Divide the text by spaces or punctuation.
  - **Output**: List of tokens for analysis.


Input string
This is an example
Tokenization
Output word list
This
is
an
example

TF-IDF

- **TF-IDF** is a method to measure the importance of a word in a document relative to a collection of **documents**. It combines:
- **Term Frequency (TF)**: How often a word appears in a document.
- **Inverse Document Frequency (IDF)**: How rare a word is across all documents.
- The higher the **TF-IDF** score, the more important the word is in that specific document.

TF-IDF FORMULA

$$TF(t,d) = \frac{number\ of\ times\ t\ appears\ in\ d}{total\ number\ of\ terms\ in\ d}$$

$$IDF(t) = log\frac{N}{1+df}$$

$$TF - IDF(t,d) = TF(t,d) * IDF(t)$$

SCAN TO SEE THE PART-2

PART III: AN ENHANCED ALGORITHM

1. Load Data

- Load user profiles from `movie.users.txt`.
- Load movie metadata (plot summaries) from `movie.metadata_plot_summary_test.txt`.

2. Preprocess Text Data

- Tokenize and normalize the text data (convert to lowercase, remove punctuation).
- Remove stop words using 'Default English stopwords list' and 'MySQL Stopwords'.

3. Compute TF-IDF

- Calculate TF-IDF for both user profiles and movie metadata.

4. Normalize TF-IDF

- Normalize the TF-IDF vectors for user profiles and movie metadata to unit length.

5. Compute Cosine Similarity

- Calculate the cosine similarity between each user profile and the not-yet-seen movies.

6. Display Results

- For each user, display the top 10 movies based on similarity, including User ID, Movie name, and similarity values.

STOP WORDS

- **Definition**: Common words like "the," "is," "and" that add little meaning.
- **Purpose**: Removed to focus on significant words in text analysis.

Filtering Process:

- **Identify**: Use lists such as 'Default English stopwords' and 'MySQL Stopwords'.
- **Remove**: Exclude these words during preprocessing.

Benefits:

- **Improved Analysis**: Emphasizes key terms and relevance.
- **Enhanced Efficiency**: Reduces data size and speeds up processing.

SCAN TO SEE THE PART-3