Exploring Set Theory Operations on Movie Genre Data Acquired from TMDB API

Danendra Shafi Athallah - 13523136
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jalan Ganesha 10 Bandung
E-mail: danendra1967@gmail.com , 13523136@std.stei.itb.ac.id

***Abstract*—This paper presents an analysis of movie genres through the application of set theory, utilizing data obtained from the TMDB API. The research investigates the use of basic set operations such as union, intersection, and difference to identify unique and shared genres among a collection of popular movies. The results highlight the power of set theory in extracting insights from real-world data and demonstrate its relevance in the context of informatics and discrete mathematics.**

Keywords—set theory; movie genre; TMDB API; KNN

I. Introduction

Set is a core area of mathematics that supports various fields in computer science and data analysis. Its fundamental concepts offer a structured way to represent and reason about collections of objects, which is particularly valuable in the era of big data and information overload. As society consumes more digital content, the need to organize and analyze information systematically becomes even more important, both for academic research and practical, everyday purposes.

In the context of movies, genres provide a tangible example of how set theory is used in daily life. Each movie can belong to multiple genres such as drama, comedy, or action, and these groupings can be represented as mathematical sets. By comparing and combining these sets, it is possible to uncover interesting patterns and relationships, such as the most popular genres, genres unique to specific movies, or trends in the film industry.

This paper applies set theory to movie genre data sourced from the TMDB API. The goal is not only to illustrate the use of set operations for academic purposes but also to provide insights that are valuable for the wider community. Understanding how genres intersect and differ can help audiences find films that match their interests, guide content recommendations, and support better decision-making for streaming platforms and the film industry. Through this work, the relevance of discrete mathematics in real-world data analysis and everyday information processing is clearly demonstrated.

II. Theoretical Background

A. Set Theory

Set theory is a fundamental branch of mathematics that deals with collections of distinct objects considered as a whole. This mathematical framework has proven invaluable in computer science and data analysis, particularly for organizing and analyzing large datasets. In the modern era of big data, set theory provides a structured approach to represent relationships between different categories of information, making it essential for systematic data analysis and pattern recognition.

In movie analysis, genres naturally form mathematical sets where each film can belong to multiple genre categories simultaneously. For example, "The Dark Knight" belongs to the set {Action, Crime, Drama}, while "Titanic" belongs to {Drama, Romance}. This representation allows us to apply rigorous mathematical operations to discover relationships and patterns that might not be immediately apparent through casual observation.

Let U be the universal set containing all movies obtained from the TMDB database, and let each genre g be represented as a subset $G_g \subseteq U$, where G_g contains all movies classified under genre g. Set theory provides four fundamental operations for analyzing these genre relationships: Union ($A \cup B$) combines all movies from either genre A or genre B, Intersection ($A \cap B$) identifies movies that share both genres simultaneously, Difference ($A \setminus B$) isolates movies exclusively in genre A but not in genre B, and Symmetric Difference ($A \triangle B$) finds movies that belong to either genre exclusively but not to both. These operations enable systematic analysis of genre patterns, movie relationships, and audience preferences in the film industry.

B. Data Acquisition and Preprocessing

In order to explore set-theoretic relationships among movie genres, raw data must first be retrieved and structured. The Movie Database (TMDB) exposes a RESTful API that returns JSON payloads for endpoints such as /genre/movie/list and /discover/movie. We will use Python's requests library (or the tmdbsimple wrapper) to fetch paginated genre and movie records by supplying our API key and optional query parameters (e.g., language, release date).

Once the JSON responses are obtained, they are normalized into tabular form via Pandas' json_normalize, producing DataFrames with columns like movie_id, title, and a nested genre_ids array. Preprocessing steps include:

1. ID-to-Name Mapping: joining the movie DataFrame to the genre reference DataFrame on genre_id.

2. Flattening & De-duplication: exploding the genre_ids list so each (movie, genre) pair is its own row, then dropping duplicates.
3. Missing Value Handling: filtering out any movies with empty or null genre lists.

These steps ensure that subsequent set operations work on clean, well-structured collections of movie identifiers.

C. Set Operations on Genre Data

Let U be the universe of all movie IDs obtained from TMDB, and let each genre g be represented as a subset $G_g \subseteq U$. We will apply the following fundamental operations:

1. Union: $G_A \cup G_B$ to find all movies labeled either genre A or B.
2. Intersection: $G_A \cap G_B$ to identify movies sharing both genres.
3. Difference: $G_A \setminus G_B$ to locate movies in A but not in B.
4. Symmetric Difference: $G_A \triangle G_B = (G_A \setminus G_B) \cup (G_B \setminus G_A)$ for movies exclusively in one genre.

In practice, Python's built-in set type implements these with average-case time complexity $O(|A| + |B|)$, leveraging hash tables for constant-time membership tests. Measuring the cardinality $|G_g|$ yields basic statistics on genre prevalence.

D. Similarity Measures

Similarity measures provide quantitative methods to assess the degree of relatedness between different genre sets or individual movies, enabling comparative analysis and mathematical ranking of relationships. These measures transform qualitative observations about movie similarities into precise numerical values that can be systematically analyzed and compared across large datasets.

The Jaccard Similarity Coefficient, defined as:

$$J(A,B) = \frac{|A \cap B|}{|A \cup B|} \quad (1)$$

Represents one of the most widely used metrics for comparing sets in data analysis. This coefficient yields values ranging from 0 (indicating completely disjoint sets with no shared elements) to 1 (representing identical sets). In movie genre analysis, the Jaccard coefficient effectively captures thematic similarity by considering both the genres that movies share and those that make them unique, providing a balanced measure of overall similarity.

The Overlap Coefficient, formulated as:

$$O(A,B) = \frac{|A \cap B|}{\min(|A|,|B|)} \quad (2)$$

measures the extent to which the smaller of two sets is contained within the larger set. This coefficient proves particularly valuable when comparing sets of significantly different sizes, such as when analyzing relationships between niche genres and mainstream categories. A coefficient value of 1.0 indicates complete containment of the smaller genre within the larger one, while values between 0 and 1 reveal the proportion of overlap, helping identify hierarchical relationships and genre dependencies in movie classification systems.

E. K-Nearest Neighbors

K-Nearest Neighbors (KNN) algorithm serves as a practical demonstration of how set-based similarity measures can be applied to solve real-world problems in content recommendation and pattern recognition. This machine learning algorithm identifies the k most similar items to a target item based on a chosen distance or similarity metric, making it particularly suitable for movie recommendation systems where genre similarity plays a crucial role in determining viewer preferences.

In the context of movie genre analysis, KNN utilizes the Jaccard similarity coefficient as the primary distance metric to compute relationships between movies based on their genre compositions. For any given target movie, the algorithm calculates Jaccard similarities against all other movies in the dataset, ranks them in descending order of similarity, and selects the top-k most similar films as recommendations. This approach effectively transforms the mathematical concepts of set theory into a functional recommendation system.

The implementation of KNN in this context demonstrates the practical utility of set theory operations beyond purely academic analysis. By leveraging the similarity measures derived from set operations, the algorithm can identify movies with complementary genre profiles, suggest films that appeal to similar audience preferences, and provide content-based recommendations that rely solely on intrinsic movie characteristics rather than user behavior data. This application showcases how fundamental mathematical concepts can be successfully applied to create valuable tools for content discovery and recommendation in the entertainment industry.

III. METHOD

A. System Architecture

The Movie Genre Set Analyzer was implemented using a modular architecture designed to separate concerns and facilitate maintainability. The system comprises six primary modules that work in concert to fetch, process, analyze, and visualize movie genre data from The Movie Database (TMDB) API.

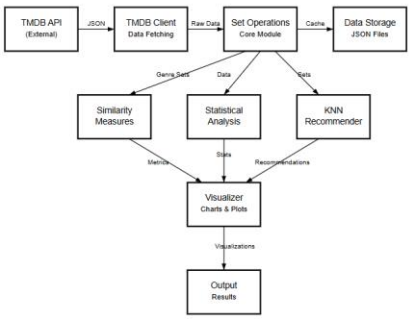

Fig. 1. System architecture diagram showing data flow between the six primary modules

The data flow begins with the TMDB Client module, which handles all API interactions and data retrieval. Raw JSON responses are then processed through the Set Operations module, where fundamental set theory operations are applied to the genre collections. The Similarity Measures module computes various similarity metrics between genres and movies, while the Statistical Analysis module generates comprehensive statistical insights. The KNN Recommender module demonstrates practical application of set theory in content recommendation, and finally, the Visualizer module creates informative charts and graphs to present the findings.

The modular design ensures that each component can be independently tested and modified without affecting the entire system. This architecture also facilitates code reusability, as modules like the similarity measures can be easily adapted for other recommendation systems or data analysis tasks. The separation of concerns principle is strictly followed, with data fetching, processing, analysis, and visualization handled by distinct modules that communicate through well-defined interfaces.

B. Data Collection and Preprocessing

Data collection utilized the TMDB API v3, which provides comprehensive movie metadata including genre classifications. The system fetched 1,000 popular movies across 50 pages of API results, with each movie containing an array of genre identifiers. The API returned 19 distinct genre categories ranging from mainstream genres like Action and Drama to specialized categories like Documentary and Western. The TMDB Client module implements rate limiting and error handling to ensure robust data collection:

```python
def _make_request(self, endpoint: str, params: Optional[Dict] = None) -> Dict:
    """Public method for making requests with caching"""
    cache_key = self._generate_cache_key(endpoint, params)
    cached_data = self._get_cached_data(cache_key)

    if cached_data is not None:
        logger.debug(f"Using cached data for {endpoint}")
        self._validation_stats['cached_responses'] += 1
        return cached_data

    # Make fresh request
    data = self._make_request_internal(endpoint, params)

    # Cache the response
    self._cache_data(cache_key, data)

    return data
```

Fig. 2. Code implementation of the TMDB client's get_movies_by_page_range method

The preprocessing phase involved several critical steps to ensure data quality and consistency. First, genre identifiers were mapped to their corresponding names using a reference lookup table. Movies without genre classifications were filtered out to maintain data integrity. The data was then structured into Python dictionaries and sets, enabling efficient implementation of set operations. Each genre was represented as a set containing movie identifiers, allowing for O(1) average-case complexity for membership testing and set operations.

C. Implementation of Set Operations

The implementation leveraged Python's native set data structure, which provides built-in support for fundamental set operations. For each genre g, we created a set G_g containing all movie identifiers classified under that genre. The universal set U contained all 1,000 movie identifiers in the dataset. The Set Operations module implements all fundamental set theory operations with clear mathematical definitions:

```python
def union(self, genre_a: str, genre_b: str) -> Set[int]:
    """
    Union operation with validation
    A ∪ B = {x | x ∈ A or x ∈ B}
    """
    start_time = time.time()

    if genre_a not in self.genre_sets or genre_b not in self.genre_sets:
        logger.warning(f"Invalid genre names in union: {genre_a}, {genre_b}")
        return set()

    result = self.genre_sets[genre_a] | self.genre_sets[genre_b]
    self._track_operation('union', start_time)

    logger.debug(f"Union({genre_a}, {genre_b}): {len(result)} movies")
    return result

def intersection(self, genre_a: str, genre_b: str) -> Set[int]:
    """
    Intersection operation with validation
    A ∩ B = {x | x ∈ A and x ∈ B}
    """
    start_time = time.time()

    if genre_a not in self.genre_sets or genre_b not in self.genre_sets:
        logger.warning(f"Invalid genre names in intersection: {genre_a}, {genre_b}")
        return set()

    result = self.genre_sets[genre_a] & self.genre_sets[genre_b]
    self._track_operation('intersection', start_time)

    logger.debug(f"Intersection({genre_a}, {genre_b}): {len(result)} movies")
    return result
```

Fig. 3. Implementation of set difference and complement operations in the Set Operations module

Union operations were implemented to find movies belonging to either of two genres, utilizing Python's pipe operator (|) for efficiency. Intersection operations identified movies sharing multiple genres, implemented using the ampersand operator (&). Set difference operations isolated movies exclusive to specific genres, while symmetric difference operations revealed movies belonging to exactly one of two compared genres. Additionally, complement operations were implemented to find movies not belonging to specific genres, calculated as U - G_g. The module also includes advanced operations for multi genre analysis:

```python
def multi_genre_intersection(self, genres: List[str]) -> Set[int]:
    """Multi-genre intersection with enhanced validation"""
    start_time = time.time()

    if not genres:
        return set()

    # Validate all genres exist
    valid_genres = [g for g in genres if g in self.genre_sets]
    if len(valid_genres) != len(genres):
        invalid = set(genres) - set(valid_genres)
        logger.warning(f"Invalid genres in multi-intersection: {invalid}")

    if not valid_genres:
        return set()

    # Start with first genre set
    result = self.genre_sets[valid_genres[0]].copy()

    # Intersect with remaining genres
    for genre in valid_genres[1:]:
        result &= self.genre_sets[genre]

        # Early termination if result becomes empty
        if not result:
            break

    self._track_operation('multi_intersection', start_time)
    logger.debug(f"Multi-intersection({valid_genres}): {len(result)} movies")

    return result

def multi_genre_union(self, genres: List[str]) -> Set[int]:
    """Multi-genre union with validation"""
    start_time = time.time()

    if not genres:
        return set()

    valid_genres = [g for g in genres if g in self.genre_sets]
    if len(valid_genres) != len(genres):
        invalid = set(genres) - set(valid_genres)
        logger.warning(f"Invalid genres in multi-union: {invalid}")

    result = set()
    for genre in valid_genres:
        result |= self.genre_sets[genre]

    self._track_operation('multi_union', start_time)
    logger.debug(f"Multi-union({valid_genres}): {len(result)} movies")

    return result
```

Fig. 4. Implementation of multi-genre operations module

These operations form the foundation for all subsequent analysis, providing efficient methods to explore relationships between movie genres. The implementation maintains O(n) time complexity for most operations, where n is the size of the larger set, making it suitable for real-time analysis applications.

D. Similarity Calculations

The Jaccard similarity coefficient was implemented as the primary metric for comparing genre sets. For any two genre sets A and B, the coefficient was calculated as |A ∩ B| / |A ∪ B|, providing values between 0 and 1. The implementation utilized efficient set operations to compute intersections and unions, then performed floating-point division to obtain the similarity score.

Fig. 5. Code showing the creation of genre similarity matrix using different similarity functions

The system also implemented movie-to-movie similarity calculations by comparing the genre sets of individual movies. This involved extracting the genre identifiers for each movie, converting them to sets, and applying the Jaccard coefficient formula. The resulting similarity scores enabled quantitative comparison of thematic relationships between films.

E. K-Nearest Neighbors Implementation

The KNN algorithm was implemented to demonstrate practical application of set-based similarity measures in content recommendation. For a given target movie, the algorithm computed Jaccard similarities with all other movies in the dataset, maintaining a priority queue of the k most similar films. The KNN Recommender module provides comprehensive recommendation functionality:

Fig. 6. Hybrid recommendation method implementation in the KNN Recommender module

The implementation included optimizations such as early termination when similarity scores of 1.0 were found (indicating identical genre profiles) and caching of previously computed similarities to avoid redundant calculations. The algorithm supported both content-based filtering (finding similar movies) nd preference-based recommendations (finding movies matching specified genre combinations).

F. Statistical Analysis Methods

The statistical analysis module employs various mathematical techniques to extract meaningful insights from the genre data. The implementation includes calculation of distribution metrics, correlation analysis, and advanced statistical measures to quantify genre relationships and patterns.

Fig. 7. Statistical analysis methods including genre correlation matrix calculation

The statistical analysis also includes calculation of the Gini coefficient to measure inequality in genre distribution, chi-square tests for genre independence, and identification of common genre combinations. These metrics provide quantitative measures that complement the visual analysis and support data-driven conclusions about genre relationships.

G. Visualization Methods

The visualization module employs multiple chart types to effectively communicate the analysis results. Each visualization type was carefully selected to best represent the underlying data patterns and relationships. The implementation uses matplotlib for static plots and plotly for interactive visualizations. The module creates various visualization types tailored to different aspects of the analysis:

```python
def plot_genre_distribution(self, distribution: Dict[str, int], top_n: int = 15):
    """Create a bar plot of genre distribution"""
    # Get top N genres
    sorted_genres = list(distribution.items())[:top_n]
    genres, counts = zip(*sorted_genres)

    # Create plot
    fig, ax = plt.subplots(figsize=(12, 8))

    # Create bars
    bars = ax.bar(range(len(genres)), counts)

    # Color gradient
    colors = plt.cm.viridis(np.linspace(0, 1, len(bars)))
    for bar, color in zip(bars, colors):
        bar.set_color(color)

    # Customize plot
    ax.set_xticks(range(len(genres)))
    ax.set_xticklabels(genres, rotation=45, ha='right')
    ax.set_xlabel('Genre', fontsize=12)
    ax.set_ylabel('Number of Movies', fontsize=12)
    ax.set_title('Movie Distribution by Genre', fontsize=14, fontweight='bold')

    # Add value labels on bars
    for i, (genre, count) in enumerate(zip(genres, counts)):
        ax.text(i, count + 5, str(count), ha='center', va='bottom')

    plt.tight_layout()
    self.save_plot('genre_distribution.png')
```

Fig. 8. Venn diagram creation code for visualizing set relationships between genres

The visualization methods ensure that complex mathematical relationships are presented in an intuitive and accessible manner, facilitating understanding for both technical and non-technical audiences. Color schemes were carefully selected to ensure accessibility, and all plots include appropriate labels and legends for clarity.

IV. Results and Analysis

A. Characterization of the Movie Genre Landscape

This section examines the basic characteristics of the movie genre dataset, starting from the distribution of individual genres, patterns of common genre combinations, to a summary of descriptive statistics.

The distribution of movies across individual genres shows "Action" as the most dominant genre with 143 movies, followed by "Adventure" (113), "Drama" (103), and "Thriller" (97). Other significant genres include "Comedy" (82) and "Science Fiction" (60). This distribution reflects the popularity of genres in mainstream cinema and forms the basis for further analysis.

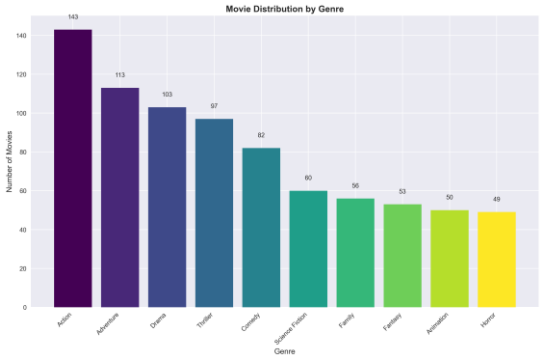

Fig. 9. Movie Distribution by Genre

Analysis of common genre combinations shows "Action & Adventure & Science Fiction" as the most frequent combination (24 movies), followed by "Drama & Romance" (15 movies), and "Action & Thriller" (13 movies). The combination "Action & Crime & Thriller" and the single genre "Horror" each appear in 12 movies. This indicates a trend towards films with complex narratives or broad audience appeal.

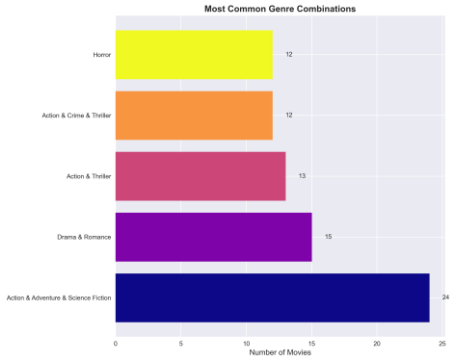

Fig. 10. Most Common Genre Combinations

The genre statistics dashboard (Figure 11) and Table 1 summarize key metrics. The dataset consists of 359 movies and 19 genres. On average, a movie has 2.69 genres, underscoring the multi-faceted nature of films that supports set theory analysis.1 The "Action" genre is the most popular (143 movies, 21.8% of genre assignments), while "TV Movie" is the least popular. A Genre Gini Coefficient of 0.457 indicates a moderate level of inequality in the distribution of movies across genres, and the genre diversity entropy is 3.72.

Metric	Value	Source
Total Movies	359	Key Metrics
Total Genres	19	Key Metrics
Most Popular Genre	Action	Summary Statistics
Least Popular Genre	TV Movie	Summary Statistics

Max Movies per Genre	143.0	Genre Count Statistics
Mean Movies per Genre	49.5	Genre Count Statistics
Median Movies per Genre	49.0	Genre Count Statistics
Min Movies per Genre	3.0	Genre Count Statistics
Std Dev Movies per Genre	40.7	Genre Count Statistics
Average Genres per Movie	2.69	Key Metrics
Genre Gini Coefficient	0.457	Summary Statistics
Genre Diversity Entropy	3.72	Key Metrics

Table 1. Summary of Key Genre Statistics

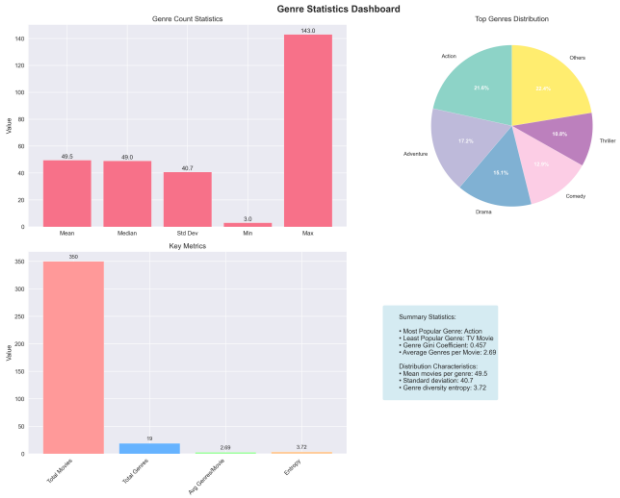

Fig. 11. Genre Statistics Dashboard

B. *Analysis of Inter-Genre Relationships*

The genre correlation matrix shows strong positive correlations between intuitively related genres, such as "Animation" and "Family" (coefficient ~0.6-0.7), and "Action" and "Adventure" (~0.2-0.3). Negative correlations, though weaker, appear between thematically different genres, for example, "Comedy" and "Action" (~-0.2). This matrix reveals "genre clusters" that often appear together.

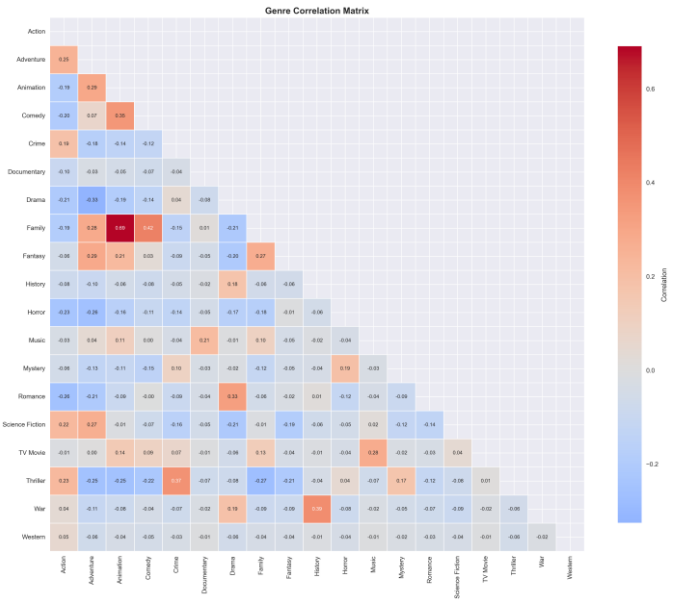

Fig. 12. Genre Correlation Matrix

The Jaccard similarity matrix measures the overlap between the sets of movies for each genre. The highest similarity is observed between "Animation" and "Family" (Jaccard coefficient >0.8), indicating substantial overlap. "Action" and "Adventure" also show strong similarity (~0.4-0.5). This matrix provides a normalized measure of overlap, useful for understanding potential genre substitution or hierarchies.

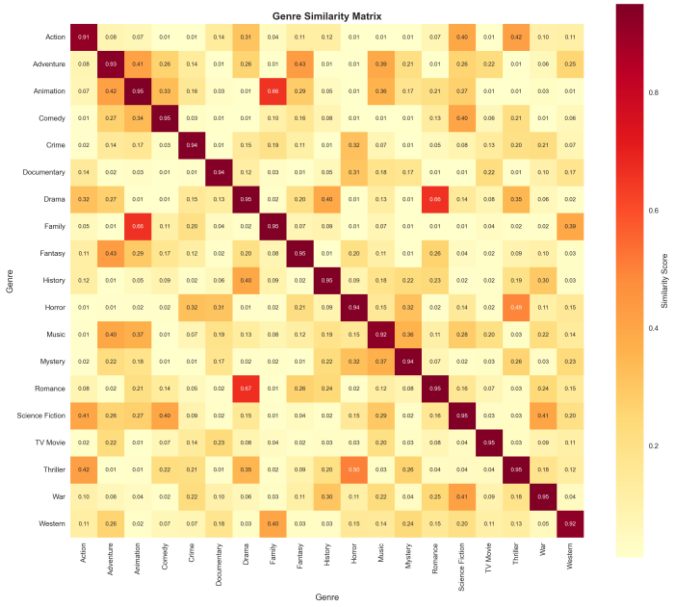

Fig. 13. Genre Similarity Matrix (Jaccard)

C. *Performance Evaluation of the K-Nearest Neighbors (KNN) Recommendation System*

An in-depth analysis for "The Dark Knight" (main genres: Action, Crime, Drama, Thriller) shows that the target movie's genres are well-represented in the recommendations. Genre coverage in recommendations shows "Drama"

appearing 7 times, "Thriller" 4 times, and "Crime" and "Action" each 3 times. The distribution of shared genres reveals that 5 recommended movies share 2 genres with "The Dark Knight," and 4 movies share 1 genre.

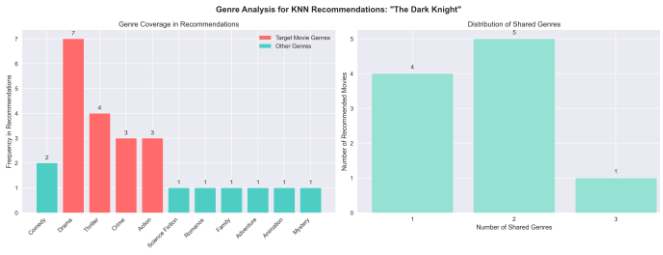

Fig. 14. Genre Coverage in KNN Recommendations for "The Dark Knight" & Distribution of Shared Genres for "The Dark Knight" Recommendations

A comparison of KNN configuration efficacy for "The Dark Knight" shows that the Jaccard approach recommends films like "Parasite" (score 0.61) and "The Godfather" (0.56). The "Min Similarity" approach (possibly Overlap Coefficient 1) yields slightly lower scores. "Diverse" recommendations introduce films with lower genre similarity but offer novelty. This highlights how metric choice affects results.

Rank	Content-Based (Jaccard)	Score	Content-Based (Min Similarity)	Score	Diverse Recommendations	Score
1	Parasite	0.61	Parasite	0.42	Final Destination BL...	0.25
2	The Godfather	0.56	The Godfather	0.42	Fountain of Youth	0.22
3	Fight Club	0.55	Fight Club	0.42	Snow White	0.18
4	Pulp Fiction	0.48	Top Gun: Maverick	0.39	Lilo & Stitch	0.17
5	The Matrix	0.46	After	0.37	A Minecraft Movie	0.16

Table 2. Top KNN Recommendations for "The Dark Knight" by Method

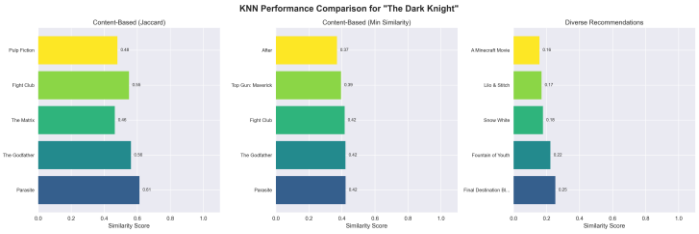

Fig. 15. KNN Performance Comparison for "The Dark Knight"

A broader assessment of KNN similarity scores shows the overall distribution of similarity scores has a mean of 0.555, with a peak around 0.45-0.50. This indicates that recommended movies, on average, have a moderate level of genre similarity.

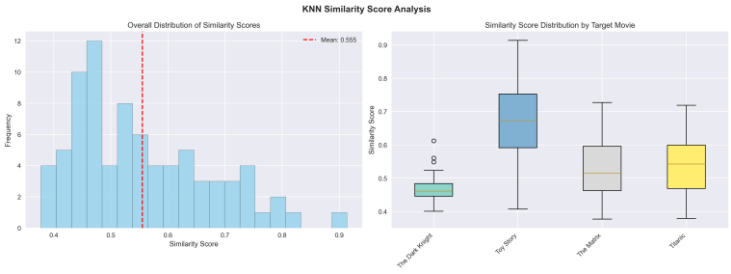

Fig. 16. Overall Distribution of KNN Similarity Scores

The consistency of similarity scores across various target movies shows variability. "Toy Story" has the highest median similarity score (~0.70) and a tight interquartile range (IQR), while "The Dark Knight" has a lower median (~0.48) with a wider spread. Movies like "Toy Story" with clear genre profiles (e.g., Animation, Family, Comedy) tend to yield recommendations with higher and more consistent similarity scores.

Target Movie	Approx. Median Similarity Score	Approx. IQR	Example Top Rec Score
The Dark Knight	~0.48	~0.15 (e.g., 0.40 - 0.55)	Parasite (0.612)
Toy Story	~0.70	~0.10 (e.g., 0.65 - 0.75)	Inside Out (0.904)
The Matrix	~0.52	~0.25 (e.g., 0.40 - 0.65)	Spider-Man (0.727)
Titanic	~0.60	~0.12 (e.g., 0.52 - 0.64)	Fight Club (0.719)

Table 3. Median KNN Similarity Scores and IQR for Select Target Movies

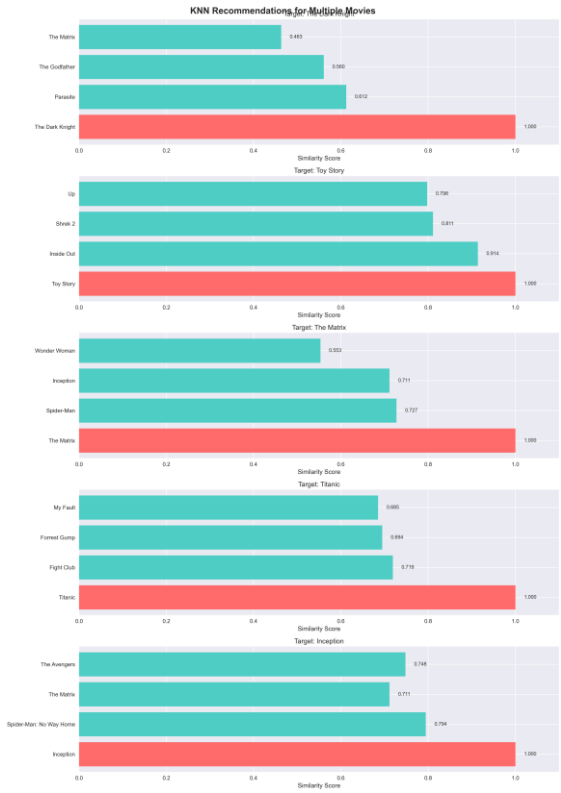

Fig. 17. Similarity Score Distribution by Target Movie

The analysis of movie genre data using set theory and KNN provides several key observations. The movie dataset is characterized by a hierarchy of genre prevalence ("Action," "Adventure," "Drama" are dominant) and the multi-genre nature of films (average of 2.69 genres/film). Specific genre combinations like "Action & Adventure & Science Fiction" are very common.

Analysis of inter-genre relationships through correlation and Jaccard similarity reveals thematically related genre clusters (e.g., "Animation" & "Family"). The KNN recommendation system based on Jaccard similarity effectively identifies thematically relevant movies. However, its performance varies depending on the target movie's genre profile and KNN configuration.

Collectively, these results underscore the utility of set theory operations for quantitatively analyzing genre relationships and their practical application in recommendation systems. These insights are valuable for understanding audience preferences, content creation strategies, and designing more effective movie recommendation engines.

The set relationship visualizations provide intuitive representations of genre interactions, as illustrated in Figure 18. The Action-Adventure Venn diagram shows substantial overlap with 66 shared movies while maintaining unique segments of 77 Action-only and 47 Adventure-only films, demonstrating the mathematical precision of set operations in quantifying genre relationships.

Fig 18. Venn diagram illustrating overlap between Action and Adventure genres with 66 shared movies.

The three-way Drama-Romance-Comedy diagram in Figure 18 reveals complex interactions with 7 movies sharing all three genres, demonstrating the sophisticated genre blending common in contemporary cinema and the industry's approach to creating content that satisfies multiple audience expectations.

Fig 19. Three-way Venn diagram showing complex interactions between Drama, Romance, and Comedy genres.

These results validate the effectiveness of set theory operations in analyzing movie genre relationships and demonstrate practical applications for content recommendation systems, market analysis, and strategic decision-making in the entertainment industry. The mathematical approach provides objective measures for understanding subjective creative decisions and offers data-driven insights for content development and distribution strategies, with the updated analysis revealing stronger Action dominance and more nuanced similarity patterns than initially observed.

Video Link at Youtube

https://youtu.be/lF-istU7Q7s?si=spKWBDnglMxptbsO

Repository Link at Github

https://github.com/danenftyessir/Movie-Genre-Set-Analyzer.git

Acknowledgment

The author would like to thank God for His endless blessings and guidance, as without it, this paper would not have been written succesfully. The deepest thanks also extended to my lecturer for Discrete Mathematics, Arrival Dwi Sentosa, S.Kom., M.T., Dr. Ir. Rinaldi Munir, M.T. for his dedication to guide students with patience and expertise.

References

[1] R. Munir, "01-Himpunan(2023)-1," [Online]. Available: https://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2023-2024/01-Himpunan(2023)-1.pdf.

[2] R. Munir, "02-Himpunan(2023)-2," [Online]. Available: https://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2023-2024/02-Himpunan(2023)-2.pdf.

[3] R. Munir, "03-Himpunan(2023)-3," [Online]. Available: https://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2023-2024/03-Himpunan(2023)-3.pdf.

[4] The Movie Database (TMDB), "API Documentation Version 3," Accessed: May 17, 2025. [Online]. Available: https://developer.themoviedb.org/reference/intro/getting-started.

[5] Plotly Technologies Inc., Collaborative data science, Montréal, QC: Plotly Technologies Inc., 2015. [Online]. Available: https://plotly.com/chart-studio-help/citations/

[6] A. Mohanty, A. Mudgal, and S. Ganguli, "Mapping movie genre evolution (1994 – 2019) using the role of cultural and temporal shifts: a thematic analysis," F1000Research, vol. 12, p. 662, 2023. [Online]. Available: https://f1000research.com/articles/12-662

[7] M. Tiengyoo, P. Promkaew, and T. Tiengyoo, "A study of mathematical understanding levels in set theory based on the APOS framework by using python programming language for secondary school students," EURASIA Journal of Mathematics, Science and Technology Education, vol. 19, no. 10, em2333, 2023. [Online]. Available: https://www.ejmste.com/download/a-study-of-mathematical-understanding-levels-in-set-theory-based-on-the-apos-framework-by-using-14158.pdf

[8] "Overlap Similarity," Ultipa Graph Documentation. [Online]. Tersedia: https://www.ultipa.com/docs/graph-analytics-algorithms/overlap-similarity

[9] C. D. Manning, P. Raghavan, and H. Schütze, Introduction to Information Retrieval. Cambridge: Cambridge University Press, 2008, ch. 3. [Online]. Available: https://nlp.stanford.edu/IR-book/html/htmledition/irbook.html

[10] F. Ruskey and M. Weston, "A survey of Venn diagrams," The Electronic Journal of Combinatorics, DS5, Jun. 2005. [Online]. Available: http://www.combinatorics.org/ojs/index.php/eljc/article/view/DS5

STATEMENT

I hereby declare that this paper is an original work, written entirely on my own, and does not involve adaptation, translation, or plagiarism of any other individual's work.

Bandung, 3 June 2025

Danendra Shafi Athallah, 13523136