DATA MINING: Jester- Movie Ratings Data Sets (Collaborative Filtering Dataset)

U23884224-Omphemetse Senna
University of Pretoria
Pretoria, South Africa

***Abstract*—This study investigates association rule mining techniques, specifically the Apriori and FP-Growth algorithms, for recommending jokes based on user ratings. The analysis focuses on universal jokes that have consistent evaluations, as well as all jokes in the dataset. Using raw and weighted ratings, the top five recommendations for each strategy were selected, providing information about user preferences and humorous co-occurrence patterns. Notably, Joke 19, Joke 5, Joke 20, Joke 7, and Joke 18 appeared as top picks for universal jokes, demonstrating their significant associative linkages. These findings are useful in developing tailored and engaging joke suggestion systems to improve user happiness.**

I. Introduction

Humor plays an important role in human connection and amusement, making it an appealing topic for recommendation systems. In this study, we employ association rule mining to examine user joke ratings and find the most popular and frequently occurring jokes. We use the Apriori and FP-Growth algorithms to investigate universal jokes with consistent ratings as well as the entire dataset, taking into account both raw and weighted ratings.

The primary purpose is to identify patterns in user preferences and make meaningful recommendations to increase engagement and happiness. Through this analysis, we aim to address critical questions:

- Which jokes have the strongest associations with others?
- What are the differences in recommendation patterns between universal jokes and all jokes?

This study examines the results in detail, providing insights into how to create effective joke recommendation systems for a wide range of audiences.

II. Literature Review

A. Introduction

Considering an increasing amount of digital data (content). Movie recommendation systems have become more essential for assisting viewers in discovering the appropriate content based on other viewer's views. The recommendation systems use data-driven algorithms to improve suggestions based on previous interactions, resulting in increased viewer satisfaction and engagement. This study takes a look at a hybrid movie recommendation model that combines both association rule mining (Apriori method) with a K-nearest neighbours (KNN) classification approach, with the MovieLens dataset serving as experimental validation. This hybrid approach addresses common recommendation system challenges, such as sparsity, scalability, and cold start issues, by improving the accuracy and relevance of recommendations for consumers looking for popular or top-rated films.

B. Methodology

The hybrid system is aimed at developing a list of movie recommendations. A hybrid score is used to calculate R_u^h for the viewer u. S_h^s is a combination of association rule support-confidence values $S_{apriori}$ and the distance-based similarity score S_{knn} obtained by KNN. The final recommendation score for a movie M is defined as:

$$S_h(m,u) = \alpha \cdot S_{apriori}(m,u) + \beta \cdot S_{knn}(m,u)$$

where α and β are weighting factors balancing the two approaches.

1) Apriori Support and Confidence: Based on a viewer's rated movies, Apriori defines association rules where frequent co-watched movies meet minimum support (S) and confidence (C). The support for a rule $X \Rightarrow Y$ is the fraction of transactions containing $X \cup Y$, while the confidence is the conditional probability of Y given X.

2) K-Nearest Neighbors Similarity: KNN measures similarity between movies using Euclidean distance. For a given movie m and user-rated movies M, the similarity score S_{knn} is calculated by averaging the similarity of the k closest movies to m.

C. Algorithms

1) Algorithm 1: Apriori Association Rule Mining: label*=0.

1) Input: Movie dataset D includes viewer ratings, minimum support S, and confidence C.
2) Process: Identify common itemsets in D based on S and construct association rules with confidence c.
3) Output: A list of frequently used movie itemsets and their corresponding confidence scores.

2) Algorithm 2: K-Nearest Neighbors (KNN): label*=0.

1) Input: User input includes movie m, dataset D, and parameter k.
2) Process: Calculate the Euclidean distance between each movie m and other movies in D. Rank the movies by distance and choose the top k as the closest neighbors.
3) Output: A list of k-nearest neighbors with similarity scores to m.

D. Experiment Design

The experimental design made use of the MovieLens dataset, which contains around 100,000 ratings from 943 viewers over 1,682 movies. The hybrid system was implemented using Python and C, with essential phases including: label*=0.

1) Data Preprocessing: Sorting and deleting duplicates to improve Apriori execution.
2) System Configuration: Setting k-values for KNN, minimum support, and confidence standards.
3) Recommendation Generation: Combined Apriori and KNN algorithms to create a hybrid recommendation list based on the final score S_h.

E. Analysis

To identify the ideal settings, the hybrid model was evaluated with different k, support, and confidence levels. For example, when the movie Star Wars (1977) (Movie ID = 50) was selected, the system yielded a high degree of matching between Apriori and KNN recommendations (66.6%). However, for lesser-known films such as Contact (1997), the matching percentage fell to 13.3%, highlighting the limitations of association rule mining with limited data.

F. Results

The hybrid approach produced more accurate and relevant recommendations than single-method models, with a significant degree of match at 66.6% for highly rated films. The Apriori-KNN combination efficiently identified both popular and genre-specific movie associations compared to the weighted score method, enhancing ranking accuracy, especially for films with a rich rating history.

G. Limitations

The system's accuracy is limited by data sparsity, as Apriori requires a minimum number of frequent itemsets and KNN requires a dense user-item matrix to perform effective similarity computations. The use of established support and confidence values may potentially slow down adaptation. Addressing these limitations through dynamic weighting or machine learning approaches could increase system robustness and scalability.

H. Conclusion

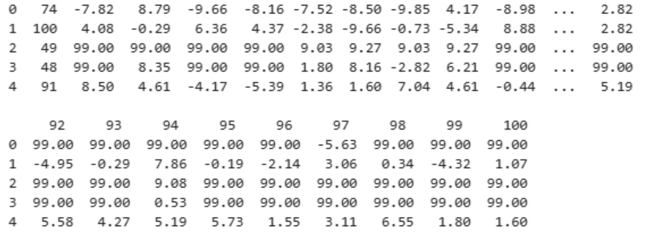
This methodical approach provides a detailed framework for our research, allowing us to expand on the ideas presented in this study for joke recommendation research. This models and algorithms can provide a solid foundation for developing and testing our recommendation system.

I. Methodology

III. Data Cleaning

A. Data-preprocessing

*1) **Structure Formatting**:* The initial dataset created interpretation challenges for us due to the absence of column headers. It was particularly difficult to distinguish between the columns reflecting the number of viewers who rated jokes and the columns representing jokes, as shown in Figure 1. Our data set has 101 columns, 100 for jokes rated and 1 for the number of viewers who provided ratings for the jokes.

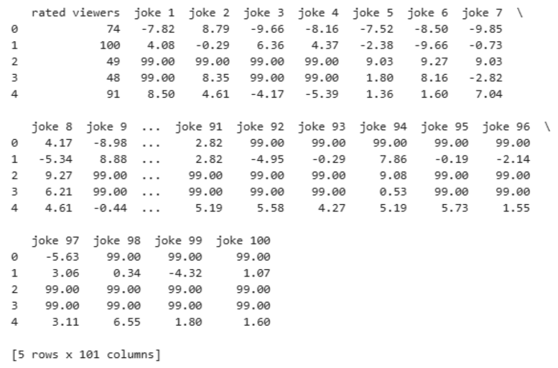


```
    0      1      2      3      4      5      6      7      8      9  ...    91  \
0  74  -7.82   8.79  -9.66  -8.16  -7.52  -8.50  -9.85   4.17  -8.98  ...   2.82
1 100   4.08  -0.29   6.36   4.37  -2.38  -9.66  -0.73  -5.34   8.88  ...   2.82
2  49  99.00  99.00  99.00  99.00   9.03   9.27   9.03   9.27  99.00  ...  99.00
3  48  99.00   8.35  99.00  99.00   1.80   8.16  -2.82   6.21  99.00  ...  99.00
4  91   8.50   4.61  -4.17  -5.39   1.36   1.60   7.04   4.61  -0.44  ...   5.19

      92     93     94     95     96     97     98     99    100
0  99.00  99.00  99.00  99.00  99.00  -5.63  99.00  99.00  99.00
1  -4.95  -0.29   7.86  -0.19  -2.14   3.06   0.34  -4.32   1.07
2  99.00  99.00   9.08  99.00  99.00  99.00  99.00  99.00  99.00
3  99.00  99.00   0.53  99.00  99.00  99.00  99.00  99.00  99.00
4   5.58   4.27   5.19   5.73   1.55   3.11   6.55   1.80   1.60
```

Fig. 1. Unstructured data

To improve the data table's readability, we renamed the first column 'Rated Viewers' and generated sequential column names, such as 'Joke 1', 'Joke 2', and so on, matching to the 100 joke columns. This set of descriptive names was assigned to the DataFrame columns, as shown in Figure 2.



```
   rated viewers  joke 1  joke 2  joke 3  joke 4  joke 5  joke 6  joke 7  \
0             74   -7.82    8.79   -9.66   -8.16   -7.52   -8.50   -9.85
1            100    4.08   -0.29    6.36    4.37   -2.38   -9.66   -0.73
2             49   99.00   99.00   99.00   99.00    9.03    9.27    9.03
3             48   99.00    8.35   99.00   99.00    1.80    8.16   -2.82
4             91    8.50    4.61   -4.17   -5.39    1.36    1.60    7.04

   joke 8  joke 9  ...  joke 91  joke 92  joke 93  joke 94  joke 95  joke 96  \
0    4.17   -8.98  ...     2.82    99.00    99.00    99.00    99.00    99.00
1   -5.34    8.88  ...     2.82    -4.95    -0.29     7.86    -0.19    -2.14
2    9.27   99.00  ...    99.00    99.00    99.00     9.08    99.00    99.00
3    6.21   99.00  ...    99.00    99.00    99.00     0.53    99.00    99.00
4    4.61   -0.44  ...     5.19     5.58     4.27     5.19     5.73     1.55

   joke 97  joke 98  joke 99  joke 100
0    -5.63    99.00    99.00     99.00
1     3.06     0.34    -4.32      1.07
2    99.00    99.00    99.00     99.00
3    99.00    99.00    99.00     99.00
4     3.11     6.55     1.80      1.60

[5 rows x 101 columns]
```

Fig. 2. structured data

2) ***Value Formatting:*** The value 99 had been intended to represent non-rated jokes; however, further investigation revealed that this value may have biased statistical computations such as mean and median because it would be seen as a high rating within the dataset. To address this issue, we chose to convert 99 instances in the joke columns (which total 100 columns) to 0, while keeping that a rating of zero means no rating.

We started by replacing all instances of 99 with NaN in all DataFrame columns. This update resulted in **68785** NaN in our dataset. Given that our initial dataset included **1835438** values, removing all NaN values would severely limit our capacity to discover recommendable jokes, leaving us with only **727200** values and a loss of **1108238** values because of the NaN entries. As a result, we chose to transform these NaN numbers to zero, preserving the quality of our dataset while ensuring that no important data was lost.

3) ***Data Division:*** We identified 10 jokes with high ratings and labelled them as "Universal Jokes." For the purposes of our analysis and comparison, we extracted and divided these Universal Jokes from the original dataset, resulting in two independent sets: one having the **Universal Jokes** and the other containing **all jokes**. This approach allows us to effectively evaluate and model the two groups in order to identify which jokes are highly rated utilising different algorithms.

B. Data Analysis

To obtain initial results from our data analysis we will use basic techniques such as plots, which will help us understand our direction. We first prepared our data in the following ways to ensure accurate outcomes.

- To extract the first column from our DataFrames, we assign it to `ratedViewersColumn`. This column typically contains information about viewers who rated the jokes.
- To select subsets of jokes, we exclude the first column (`rated viewers`) from both `jokesSubset` and `AllJokes` DataFrames. This results in `noviewers_universalJokes` and `noviewers_allJokes`, which contain only the ratings for universal and all jokes, respectively.
- Calculates the mean ratings, which is the average ratings for universal and all jokes.

1) *For Universal Jokes::* **Bar Chat:**

Figure 3 presents the top ten universal jokes. It shows the average ratings of the top 10 "universal jokes" from the universal dataset, with negative values representing lower ratings and positive values indicating higher ratings. Joke 5 has the highest average rating, with a small positive value, indicating that it is the most popular of the jokes analyzed. Joke 19 has a positive average rating; however, it is significantly lower than Joke 5. These two jokes have received the most positive ratings in this selection. Joke 16 has the lowest average rating (approximately -3.0), indicating that it is the least popular joke in this dataset.

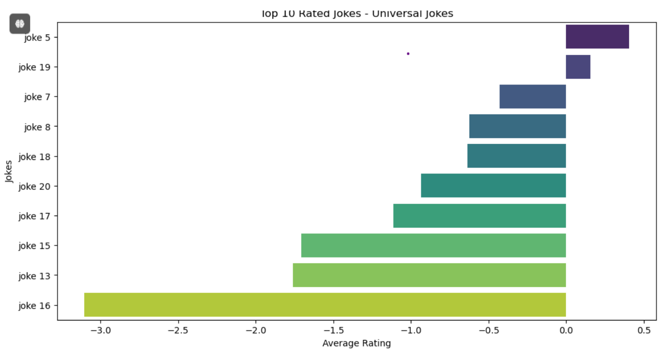


Fig. 3. Bar Chat

Other jokes with low ratings include Joke 13 and Joke 15, which have average ratings of less than -1.0, indicating a low level of user enjoyment. Jokes like Joke 7, Joke 8, and Joke 18 have average ratings closer to 0, indicating that viewers had mixed reactions or did not rate them at all. On average, neither of these jokes is strongly liked nor disliked. The data often shows more negative than positive scores, implying that many of these ”universal jokes” are poorly accepted on average.

Histogram

Figure 4 shows the distribution of ratings for all ”universal jokes” in the Universal Jokes datasets. We found a significant peak around -10.0, indicating a high amount of highly bad reviews. Another rise appears near the neutral (0) threshold, indicating that many jokes received average or neutral ratings. The distribution is slightly skewed to the right, with more low ratings (below zero) and fewer high ratings.

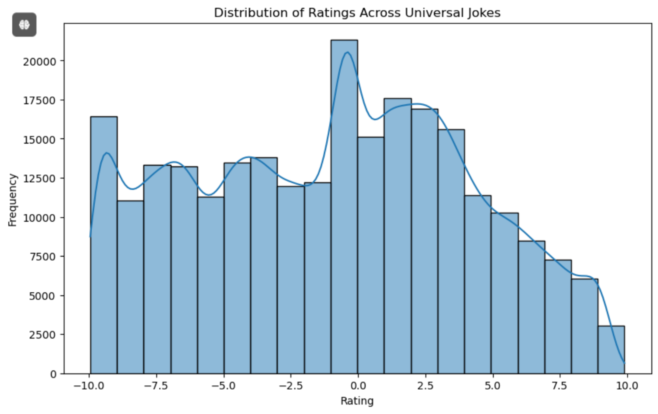


Fig. 4. Histogram

This suggests that the jokes are generally rated negatively or neutrally rather than positively. This distribution demonstrates that the majority of joke ratings are neutral or very low, with a few good ratings. This discovery suggests that determining a subset of positively rated jokes for recommendation purposes may be difficult, as most jokes appear to be assessed negatively or neutrally by viewers.

2) *FOR ALL THE JOKES::* **Bar chats**

Figure 5 shows the top 10 jokes based on the average ratings of all viewers who rated. Joke 50 is the most highly rated joke, with a much higher average rating than the rest. This means that it had the greatest positive impact on users. Joke 36 and Joke 27 follow closely behind, indicating that these jokes were also well received. The difference between the highest-rated joke (Joke 50) and the lowest among the top ten (Joke 54) is slight. This indicates a small range of viewer choices for the best jokes.

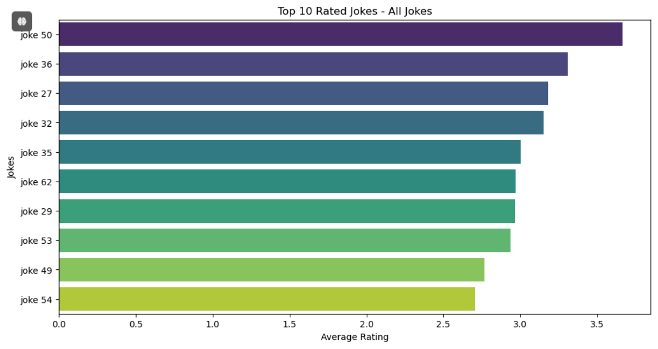


Fig. 5. Bar Chat

Histogram

Figure 6 shows the distribution of ratings across all jokes. The results show that a significant majority of the ratings are clustered near 0, forming a sharp peak. This shows that most viewers rates jokes neutrally or avoid rating them and providing strong positive/negative feedback. Ratings are distributed rather evenly across the rest of the range, however the frequency of significant ratings (both -10 and 10) is considerably reduced. The distribution is uneven, with a minor skew to negative evaluations (left of zero).

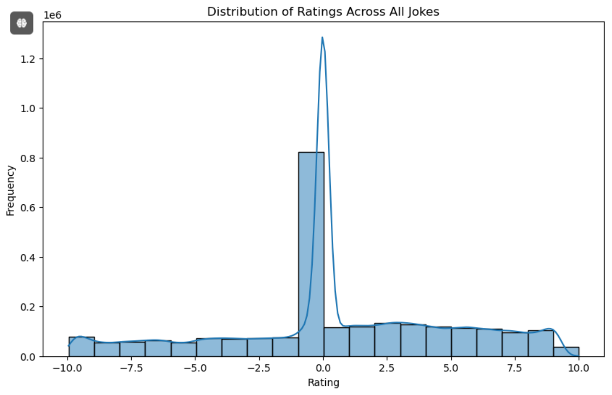


Fig. 6. Unstructured data

IV. Methodology

We decided to use the Apriori Algorithm and FP-Growth to identify our top jokes. Apriori is effective at detecting correlations between items and, in our example (Top jokes), allowing analysts to identify trends in viewer preferences. By applying it to jokes, one can determine which jokes are commonly appreciated together or by similar audiences, offering information about what defines a ”top” joke.

The FP-Growth algorithm will assist us by scanning the dataset and identifying popular jokes and their support counts. It then creates the FP-tree by inserting transactions in compressed format and sorting frequent jokes by frequency.

Dealing with both datasets (Universal and All Jokes), we will continue to use this approach to determine which joke is rated from both datasets. Before we used this algorithm, we discovered that the number of viewers who rated the jokes can also make a significant difference when it comes to selecting the best jokes. Some jokes can get less low rates, less high rates, or any scenario that could lead to the decision.

To take an advanced approach to the issue of rate variance (where some films may obtain high ratings with fewer users while others may receive lower ratings with more users), we can utilize the formula:

$$WeightedRating = \frac{R \cdot v + C \cdot m}{v + m}$$

Where: R = Average rating of the joke
v = Number of ratings for the joke
C = Mean rating across all jokes
m = Minimum number of ratings required to consider a joke in the recommendation (e.g., the 70th percentile of the number of ratings).

With this formula, we aim to address the issue where ratings may be biased by the number of viewers that evaluated each joke, resulting in an unfair comparison. A joke rated by a small number of viewers may appear to be comparable to one evaluated by many, but the bigger sample should be more confident.

As a result, we decided to address this issue through the usage of Weighted Average ratings. This ensures that jokes with a low number of ratings (high variation) receive less weight than ones with many ratings.

Preparation before Algorithm:

Apriori algorithms commonly use binary values (1 and 0) to represent the presence or absence of objects in transaction data sets. In the context of ratings, a number of 1 indicates that a specific item (for example, a joke) is rated, whereas a value of 0 indicates that it is not rated or is rated less than zero.

Raw Rating to Binary Conversion:

- **Jokes subset raw binary:** For each DataFrame `noviewers_universalJokes`, the `applymap` function is combined with a lambda function to determine whether each rating value x is larger than zero. If true, it assigns a value of 1 (showing that the joke was rated); else, it assigns a value of 0 (meaning that the joke was not rated or was rated very low).
- **All jokes in raw binary format:** A similar conversion is performed on another DataFrame, `noviewers_allJokes`.

Weighted Ratings for Binary Conversion:

- The DataFrame `jokes_subset_weighted_binary` undergoes the same transformation, indicating that it may have weighted ratings.
- The transformation for `all_jokes_weighted_binary` likewise applies to `all_jokes_weighted`.

V. Results

Approach	**Top 5 Rules**
Apriori - Raw Ratings (Universal Jokes)	[Joke 20, Joke 18 $\Rightarrow$ Joke 19 (Support: 0.20, Confidence: 0.79, Lift: 1.48), Joke 19, Joke 18 $\Rightarrow$ Joke 20 (Support: 0.20, Confidence: 0.64, Lift: 1.47), Joke 7, Joke 19 $\Rightarrow$ Joke 20 (Support: 0.20, Confidence: 0.64, Lift: 1.47), Joke 5, Joke 19 $\Rightarrow$ Joke 20 (Support: 0.22, Confidence: 0.63, Lift: 1.44), Joke 7, Joke 20 $\Rightarrow$ Joke 19 (Support: 0.20, Confidence: 0.75, Lift: 1.40)]
Apriori - Weighted Ratings (Universal Jokes)	[Joke 5 $\Rightarrow$ Joke 19 (Support: 1.00, Confidence: 1.00, Lift: 1.00), Joke 19 $\Rightarrow$ Joke 5 (Support: 1.00, Confidence: 1.00, Lift: 1.00)]
FP-Growth - Raw Ratings (Universal Jokes)	[Joke 20, Joke 18 $\Rightarrow$ Joke 19 (Support: 0.20, Confidence: 0.79, Lift: 1.48), Joke 19, Joke 18 $\Rightarrow$ Joke 20 (Support: 0.20, Confidence: 0.64, Lift: 1.47), Joke 7, Joke 19 $\Rightarrow$ Joke 20 (Support: 0.20, Confidence: 0.64, Lift: 1.47), Joke 5, Joke 19 $\Rightarrow$ Joke 20 (Support: 0.22, Confidence: 0.63, Lift: 1.44), Joke 7, Joke 20 $\Rightarrow$ Joke 19 (Support: 0.20, Confidence: 0.75, Lift: 1.40)]
FP-Growth - Weighted Ratings (Universal Jokes)	[Joke 5 $\Rightarrow$ Joke 19 (Support: 1.00, Confidence: 1.00, Lift: 1.00), Joke 19 $\Rightarrow$ Joke 5 (Support: 1.00, Confidence: 1.00, Lift: 1.00)]
Apriori - Raw Ratings (All Jokes)	[Joke 49, Joke 32 $\Rightarrow$ Joke 62, Joke 50 (Support: 0.50, Confidence: 0.79, Lift: 1.19), Joke 62, Joke 50 $\Rightarrow$ Joke 49, Joke 32 (Support: 0.50, Confidence: 0.76, Lift: 1.19), Joke 49, Joke 36 $\Rightarrow$ Joke 62, Joke 50 (Support: 0.50, Confidence: 0.79, Lift: 1.19), Joke 62, Joke 50 $\Rightarrow$ Joke 49, Joke 36 (Support: 0.50, Confidence: 0.75, Lift: 1.19), Joke 35, Joke 50 $\Rightarrow$ Joke 29, Joke 32 (Support: 0.51, Confidence: 0.75, Lift: 1.19)]
FP-Growth - Raw Ratings (All Jokes)	[Joke 49, Joke 32 $\Rightarrow$ Joke 62, Joke 50 (Support: 0.50, Confidence: 0.79, Lift: 1.19), Joke 62, Joke 50 $\Rightarrow$ Joke 49, Joke 32 (Support: 0.50, Confidence: 0.76, Lift: 1.19), Joke 49, Joke 36 $\Rightarrow$ Joke 62, Joke 50 (Support: 0.50, Confidence: 0.79, Lift: 1.19), Joke 62, Joke 50 $\Rightarrow$ Joke 49, Joke 36 (Support: 0.50, Confidence: 0.75, Lift: 1.19), Joke 35, Joke 50 $\Rightarrow$ Joke 29, Joke 32 (Support: 0.51, Confidence: 0.75, Lift: 1.19)]

VI. Conclusion

Based on the findings, the top five recommended jokes for universal appeal using the Apriori algorithm with raw ratings are Joke 19, Joke 5, Joke 20, Joke 7, and Joke 18. These jokes were found to have strong associative links with one another, as indicated by their high lift and confidence scores.

Observations:

1) Joke 19 was a popular joke, appearing in numerous rules with high lift values. This suggests that it has a strong connection to other jokes, making it a central recommendation.
2) Joke 5 displayed universal appeal by repeatedly linking to other popular jokes, indicating that it is favored among a wide range of people.
3) Joke 20, Joke 7, and Joke 18 were also frequently linked, establishing a cluster of jokes that appear to co-occur in user preferences. This emphasizes their complementary nature in humor styles or subjects.

References