Jokes Recommendation System: A Survey

Richa Muktibodh[1], Siddhant[1], Ameya Kumar[1], Sourav Chakraborty[1], and Chayan Tank[1]

[1]Department of Computer Science Engineering
Indraprastha Institute of Information Technology
New Delhi, India, 110020

Abstract

We are developing a system for predicting joke ratings, where we categorize jokes into existing classes. Once a joke is assigned to a specific class, we utilize the mean rating that users have given to that class to score the incoming joke. We have evaluated various classification and regression models and compared their accuracy scores in this context.

1 Introduction

A recommendation system is a type of software or algorithm designed to suggest items, products, or content to users based on their past behaviors, preferences, and other relevant data. It provides users with personalized and relevant recommendations, enhancing their experience by helping them discover new items they might be interested in. There is a multitude of research ongoing behind creating more efficient RS, which can be used in various domains like multimedia, social media, news, and products in various online websites. In this paper, we are interested in creating RS for jokes, where we recommend jokes to a user based on his past joke ratings. The below section discusses various approaches for building a recommendation system.

1.1 Popular methods to build recommendation system

Inferencing from paper by Isinkaye et al.(2015) [1]. RS can be mainly be classified into two broad classes i.e. Content-based and collaborative filtering-based systems. Content-based systems are systems where a profile of items liked or purchased by the user is made. Based on the item profile, a user profile is generated based on their past interactions or explicitly stated preferences. The user profile is then matched against a catalog of items using some similarity metrics to make recommendations. Often, finding the appropriate features needed to build the item profile is very hard. Also, people might have multiple interests, but this system limits the recommendation to only what the user has previously rated/liked/purchased.

The collaborative filtering-based systems rely on collective choices made by a group of users. First, it identifies users who have similar preferences or behavior patterns based on their interactions with items. To find the similarity, Cosine similarity, Pearson correlation coefficients are used. One of the challenges faced by these systems is finding the k most similar users(or items). To mitigate this, we can use clustering or dimensionality reduction techniques. Also, the user/item matrix is sparse which makes it hard to find the users who have rated/liked/purchased the same item for which we are making recommendations. We can also implement two or more RS and combine them. These are typically called Hybrid models.

Preprint. Under review.

1.2 Literature survey

Eigentaste, developed by Ken Goldberg et al.(2001) [2] used a two-phase approach, dividing its operations into offline and online stages. In the offline phase, it employs PCA to reduce dimensionality efficiently and organizes users into clusters within this lower-dimensional space. In the online phase, it leverages eigenvectors to map new users into clusters and utilizes a lookup table to suggest relevant items, ensuring that its runtime remains unaffected by the database's user count.But it uses a gauge set of size K to form the user rating matrix where all users have rated all items. As some users might not rate all items in the catalog, forming a gauge set causes a loss of information for other users.

Iterative PCA developed by Dohyun Kim et al.(2005) [3] filled the missing ratings in the user-item matrix by an appropriate method (instead of selecting a gauge set) and then they combine collaborative filtering and PCA to enhance recommendation quality. They apply PCA to reduce data dimensionality and extract latent factors that capture user-item interactions. To optimize the accuracy of recommendations, they introduce an iterative process to refine PCA-based recommendations.

The hybrid recommendation system algorithm introduced by Iqbal et al. in 2022, as described in [4], adopts a two-phase approach, which divides its operations into offline and online stages. During the offline stage, the algorithm employs Affinity Propagation clustering to process the similarity matrix generated from the dataset. One notable advantage of this algorithm, in comparison to other clustering methods, is its ability to automatically determine the number of clusters in the dataset, removing the need for user-provided cluster numbers. In the context of this algorithm, unrated jokes by a specific user are normalized to have a rating of 0. Once the user-item matrix is established, pairwise similarities between users are computed using the negative Euclidean distance, and the resulting similarity matrix is then utilized as input for the Affinity Propagation algorithm to create clusters. In the online stage, the algorithm considers both the cluster's population and its similarity to the active user to suggest

In the paper introduced by Miyahara et al.(2000) [5], they discuss about performing collaborative filtering based on a simple Bayesian classifier algorithm. They have converted the problem into a classification task where the jokes fall into either classes "like" or "dislike" based on some predefined threshold, they have labeled "like" class as 1 and "dislike" as 0. The null values are handled by placing 0 in both the like and dislike cells on the data matrix. After this transformed data matrix the task was simply to classify a given joke as liked (1) or disliked (0) based on what the implemented supervised classification model predicts. (Bayesian classifier in case of [5]), It is shown that it performs well in the classification tasks despite of its simplicity. They have evaluated these algorithms using a database of movie recommendations and joke recommendations. It is interesting to note that their results show that the proposed Bayesian approaches significantly outperforms a correlation-based collaborative filtering algorithm.

2 Exploratory data analysis

2.1 Dataset

The dataset we have taken for this RS task is the "Jester 4.1 million" dataset. The dataset essentially contains 4 files, 1) Jester_dataset_1_joke_texts (Contains 100 HTML files for 100 joke texts with joke ID's associated with them), 2) Jester_dataset_1_1.zip (Ratings Data from around 25k users who have rated 36 or more jokes), 3) Jester_dataset_1_2.zip (Data from around 23.5k users who have rated 36 or more jokes), 4) Jester_dataset_1_3.zip (Data from around 25k users who have rated between 15 and 35 jokes)

In the ratings datasets, the rows represent user ratings of the 100 jokes. The first column gives the number of jokes rated by each user (associated with row). The next 100 columns give the ratings for jokes 1 to 100. There are essentially 73k+ user ratings if we account for all three ratings datasets, The ratings are numerical values which represent how funny the users find each joke. The scale ranges from -10 (not funny) to +10 (very funny), with 0 representing a neutral rating (the value "99" corresponds to "null" = "not rated"). The Jester dataset is quite sparse as many users have not rated all the jokes.

2.2 Pre-processing of ratings dataset

Firstly, we combine all the three ratings datasets to make one dataset of around 73k+ rows and 101 columns. The null Values in the dataset (set as "99") are being set as 0.

Taking inferences from the approaches on the Eigentaste algorithms [2][6][7] , We perform:

Normalization: Before applying unsupervised clustering algorithms (like Eigentaste), the data needs to be normalized. Normalization typically involves scaling the ratings to a common range to account for variations in users' rating behaviors. We have used the following Formula for mean normalisation:

$$\mathbf{Xi - X\_mean} \tag{1}$$

Where Xi is the user rating and X_mean is the mean of that all ratings in that row.

Similarity Matrix Calculation: Once the data is normalized, a similarity matrix is calculated. This matrix quantifies the similarity between users based on their rating patterns. For example, users who have similar preferences will have higher similarity scores, and those with different preferences will have lower scores. The unsupervised clustering algorithm, such as Eigentaste, uses this similarity matrix to group users with similar preferences. The algorithm can then make recommendations based on the preferences of similar users. We have used following formula for Cosine Similarity:

$$\cos(\mathbf{a},\mathbf{b}) = \frac{\mathbf{ab}}{\|\mathbf{a}\|\|\mathbf{b}\|} = \frac{\sum_{i=1}^{n}\mathbf{a}_i\mathbf{b}_i}{\sqrt{\sum_{i=1}^{n}(\mathbf{a}_i)^2}\sqrt{\sum_{i=1}^{n}(\mathbf{b}_i)^2}} \tag{2}$$

Where a is row vector for one user and b is row vector for another user.

Principle Component analysis: After the similarity matrix calculation we can apply PCA to the cosine similarity matrix. PCA will identify the principal components, which are linear combinations of the original dimensions that capture the most variance in the data. Experimentally we have Found that first 30 Components cover almost 85% of the data. These components will help us in our clustering task ahead.

2.3 Jokes dataset analysis

The dataset comprises of 100 jokes in HTML format. Along with that there are three CSV files with the ratings of these jokes by 73,421 distinct users. In instances where a user has not rated a particular joke, the default rating of 99 is assigned to the cell.

3 Our Methodology

3.1 Pre-processing pipeline

For processing the data, we have used two popular NLP libraries called spacy and NLTK. Each joke undergoes sequential preprocessing through the following pipeline to generate a processed version.

Lowercasing: The initial step involves converting the entire joke text to lowercase. This normalization ensures uniformity and consistency in the subsequent processing steps.

Abbreviation Expansion: Following lowercase conversion, abbreviations within the jokes are expanded. For instance, if the joke contains "Q:", it is replaced with "Question", and if "A:" is present, it is substituted with "Answer". This step aims to provide clarity and context to abbreviated elements.

Forward Slash Removal: Forward slashes are removed from the text. This action is taken to facilitate the transformation of contractions and handle words like "aren't" becoming "are not" and "isn't" becoming "is not". Forward slashes preceding single inverted quotes are eliminated in this process.

Contractions Replacement: A predefined list of contractions is maintained, and each contraction in the joke text is replaced with its expanded form. This step ensures that words like "don't" are converted to "do not" and other contractions are appropriately substituted.

Special Character Removal: Subsequent to abbreviation expansion and contraction replacement, any remaining special characters are removed from the text. This step contributes to cleaning the data by eliminating non-alphanumeric characters.

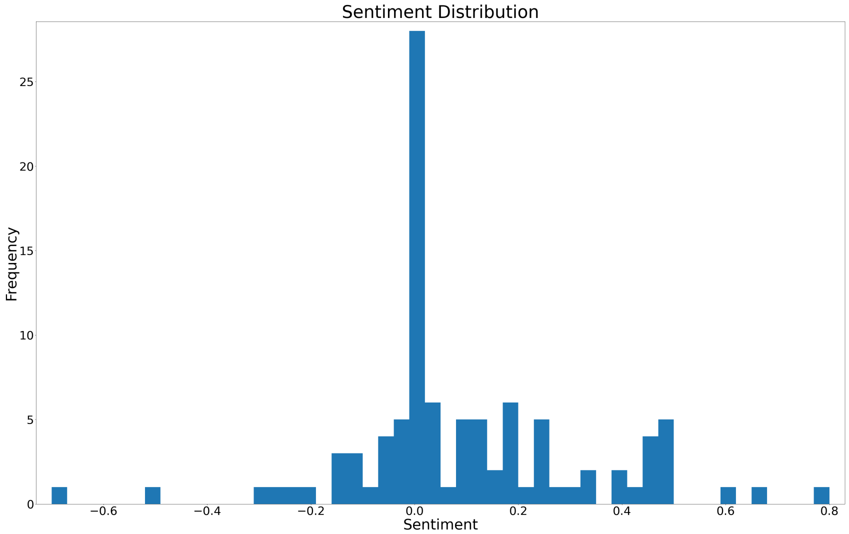

Figure 1: Sentiment Analysis

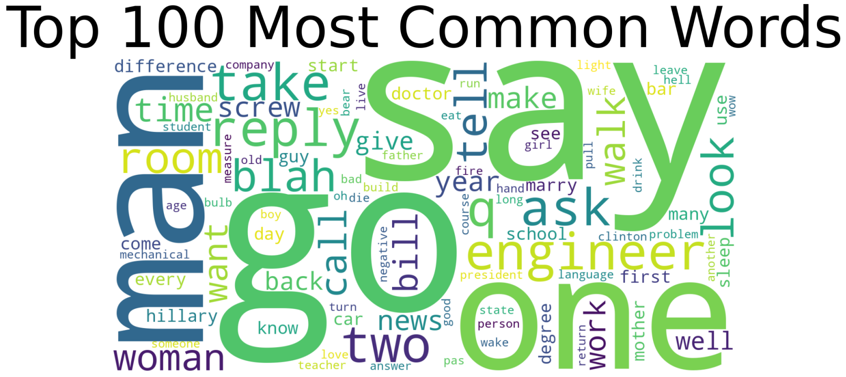

Figure 2: Word Cloud

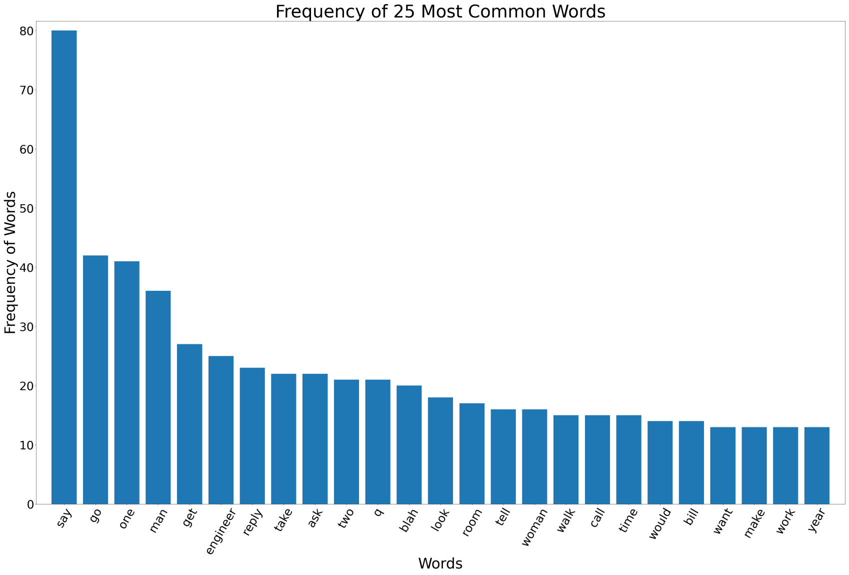

Figure 3: Frequencies of most common words

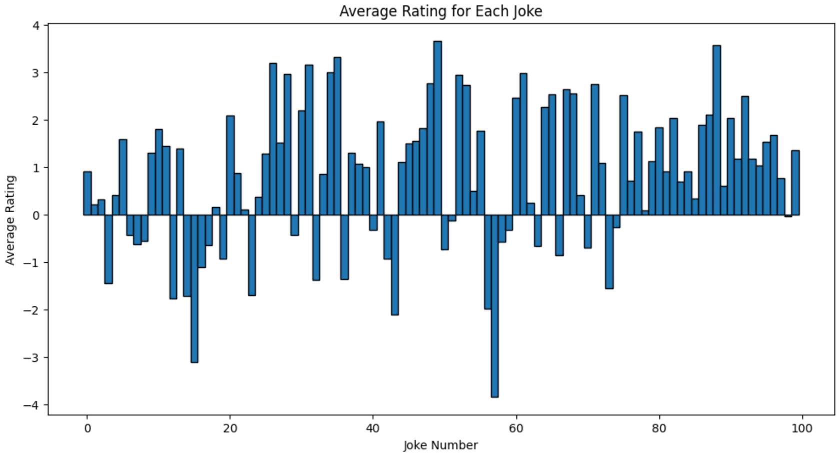

Figure 4: Average Rating for each Joke

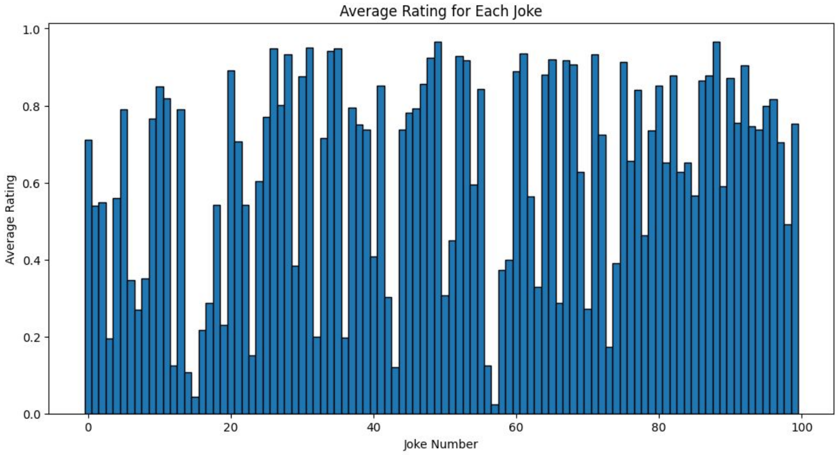

Figure 5: Scaled mean ratings

Stop Word Removal: Common stop words, which typically do not carry significant meaning, are removed from the text. This enhances the efficiency of downstream processes by focusing on more informative words.

Lemmatization: The text undergoes lemmatization, a process where words are reduced to their base or root form. This step helps in standardizing the representation of words, reducing inflections to a common base form. For lemmatization we have used spacy library.

3.2 Featurization

The final processed string is then sent for vectorization. Vectorization transforms the textual data into numerical vectors, making it suitable for machine learning algorithms to analyze the text. For featurization, a combination of n-grams and tf-idf vectorization is employed. During training, a technique called windowing is utilized, with each window having an overlap of 3 words. Specifically, sentences are segmented into sets of 8 words, and multiple overlapping windows are created from a given text or joke. Each set of 8 words, with a 3-word overlap, is then associated with the ID of the corresponding joke.

Example of windowing with a 3-word overlap:

Let’s consider the following example:-

joke id 1: "Why did the chicken join a band? Because it had the drumsticks."

joke id 2: "What did one ocean say to the other ocean? Nothing, they just waved."

The following data will be generated, which will be used for training purposes:

JokeId, Joke

1,' Why did the chicken join a band?'

1,' join a band? Because it had the'

1,' it had the drumsticks'

2,' What did one ocean say to the'

2,' say to the other ocean? Nothing, they'

2,' Nothing, they just waved.'

For the classification techniques, we have maintained a map of jokeId and the mean rating of the joke provided by the users. As the mean rating is very low, we have used a sigmoid function to scale the ratings up to 1. The best joke will have a score of 1, and the worst joke will have a score of 0.

4 Score Comparison For Various Models and Techniques

4.1 Regression Techniques

The methodology mentioned above outlines a comprehensive approach to solving the problem using different regression models to predict average ratings on jokes data. 'train_test_split' from sklearn was used to split data using an 80-20 ratio, setting 'random_state' parameter to 42 to ensure the reproducibility of the results.

A KNN regressor model is initialized with n_neighbors as three and the metric as Euclidean, which were found by running a grid search over a specified parameter grid for KNN based on the negative mean squared error across a 5-fold cross-validation. After evaluation, the R-squared metric was found to be 0.8085, and the Mean Squared Error (MSE) was 0.4096. A linear regression model was fitted on the training data and evaluated on the test data using the R-squared metric and MSE, and their values were found to be 0.8167 and 0.3920, respectively. SVR model with a linear and rbf kernel was trained, and their performance was evaluated using the R-squared metric and MSE. Linear SVR performed better than rbf SVR with R square and MSE values 0.7169 and 0.6054, whereas the same metric values for rbf kernel were 0.5077 and 1.053.

4.2 Classification Techniques

Tackling this problem as a classification task entails assigning each joke from the dataset to represent a class which is associated with the sigmoid scaled mean rating of that joke. The 'train_test_split' from sklearn was used to split data using an 80-20 ratio, setting 'random_state' parameter to 42 to ensure the reproducibility of the results.

The model pipeline incorporates TFidf with N-grams (both unigrams and trigrams) to transform the input data into feature vectors. Joke IDs serve as the class labels for each joke, These labels are then mapped with their ratings. These feature vectors are then utilized by various models to perform classification. Specifically, the Support Vector Classifier (SVC) with a linear kernel is employed, achieving an accuracy score of 0.88. Additionally, alternative models have been explored for this classification task, including Random Forest (accuracy: 0.8177), KNN (accuracy: 0.782), Naive Bayes (accuracy: 0.495), and Decision Tree (accuracy: 0.488).

The comparative performance of these models is summarized in the table below:

5 Conclusion

In the regression models, both KNN Regressor and Linear Regression models performed well, with Linear Regression having slightly higher R^2 value and lower MSE. SVR, while still performing reasonably well, has a lower R^2 and a higher MSE compared to the other two.

Regression	Score (R^2)
KNN	0.808
Linear Regression	0.816
SVR	0.716

Classification	Accuracy
SVC	0.882
Random Forest	0.817
KNN	0.782
Naive Bayes	0.495
Decision Tree	0.488

Table 1: Model Performance

Among the classification models, Support Vector Classifier outperformed other models in terms of both accuracy and F1 score making it the best model for the task of predicting joke ratings.

Given, the comprehensive evaluation of multiple regression and classification models, the SVC model with a linear kernel and n-grams from 1 to 3 is the optimal model.

References

[1] Isinkaye, Folasade Olubusola, Yetunde O. Folajimi, and Bolande Adefowoke Ojokoh, 2015. “Recommendation systems: Principles, methods and evaluation." *Egyptian informatics journal 16, no. 3 (2015): 261-273.*, https://doi.org/10.1016/j.eij.2015.06.005

[2] Ken Goldberg, Theresa Roeder, Dhruv Gupta, Chris Perkins , 2001. “Eigentaste: A Constant Time Collaborative Filtering Algorithm", *Information Retrieval 4, 133–151*, https://doi.org/10.1023/A:1011419012209

[3] Dohyun Kim, Bong-Jin Yum (2005), “Collaborative filtering based on iterative principal component analysis", *Expert Systems with Applications, Volume 28, Issue 4,Pages 823-830*, https://doi.org/10.1016/j.eswa.2004.12.037

[4] Iqbal Qasim, Mujtaba Awan, Sikandar Ali, Shumaila Khan, Mogeeb A. A. Mosleh, Ahmed Alsanad, Hizbullah Khattak, Mahmood Alam (2022) ,“Affinity Propagation-Based Hybrid Personalized Recommender System", *Complexity, vol. 2022, Article ID 6958596, 12 pages*,https://doi.org/10.1155/2022/6958596

[5] Miyahara. Koji, and Michael J. Pazzani (2000), “Collaborative filtering with the simple bayesian classifier", *Pacific Rim International conference on artificial intelligence, pp. 679-689. Berlin*, https://doi.org/10.1007/3-540-44533-1_68

[6] Nathanson, Tavi. (2009) “Algorithms, Models and Systems for Eigentaste-Based Collaborative Filtering and Visualization". *Master’s thesis, EECS Department, University of California, Berkeley.*, https://www2.eecs.berkeley.edu/Pubs/TechRpts/2009/EECS-2009-85.pdf

[7] Tavi,Nathanson, Ephrat Bitton, and Ken Goldberg (2007) “ Eigentaste 5.0: constant-time adaptability in a recommender system using item clustering." *In Proceedings of the 2007 ACM conference on Recommender systems, pp. 149-152.*, https://doi.org/10.1145/1297231.1297258