CMPE 256 PROJECT

Joke Recommendation System

Team Group - JRS

By:-

Amruta Dhondage(011416210)

Navneet Jain(011419291)

Surbhi Jain(011428040)

Project Link :- https://github.com/surgoku/CMPE-256_project_joke_recommendation

Ch.1 Introduction

Motivation:

The motivation to build recommendation system on real life problem came from course work as significant portion of the course has been around Recommendation System. As we have gone through several examples, programming assignment around Recommendation System, we were interested to work on Recommendation System on bigger scope. We wanted to take a real problem where we could apply the knowledge we learn from the class. We tried to find several datasets, however majority of the available datasets are already solved problem and apply standard matrix factorization techniques. Therefore, we picked Jester data, which has millions of ratings 1M. However, joke items are limited (150). We planned to extend this dataset by extending the joke level information by providing labelling and grouping users together based on labelling. We then tried alternate algorithms to make the recommendations better and evaluated based on accuracy. We also extend evaluation on how item level features or commonly known as meta-level information about items/users can significantly improve the Recommendation System.

For Joke recommendation system based on user's preferences we obtained the data from UC Berkeley's Jester Portal. Jester has a collection of hundreds of jokes within millions of ratings provided by hundreds and thousands of users. We aim to a model which understand user's preference, likes/dislikes to recommend the most engaging jokes for the user. We further aim to analyze the categorization of jokes into multiple domains. For recommendation purpose, we will try to find the nearest domain and then predict the jokes from the closest domain for a given user.

Secondly, we can categorize the jokes into buckets and use the user ratings data to place every user in a particular category of jokes. Each user can like more than one category of jokes. By this we can find a target audience for a new joke in the market. The approach will be, place the new joke in one of the buckets and the send it to the users in that bucket.

Objectives:

1. Build a recommendation system which looks for user's preferences using the following techniques:
    a. Ranking
    b. Rating
    c. Item
2. Perform categorization of jokes. Mapping each item to the corresponding category and use that as features while performing the matrix factorization.
3. Classifying user's choice of jokes into likes and dislikes based on ratings.
4. Mapping joke category to users forming a bucket. This will give the target audience for each new joke in the market. Like this new joke will only be sent to the audience who are most likely to read them in comparison to sending to every user.
5. By building the Recommendation System, we evaluate the performance of the Recommendation System by optimizing on RMSE (full-form). The Recommendation System eventually can be used to either predict the ratings for new/unkown user-joke pair or get the preference ranking of joke for each user.
6. Perform a comprehensive evaluation of Joke Recommendation System for the given dataset.
7. Evaluate the performance of Recommendation System with the following item level information:
    a. Joke categories
    b.Glove Embeddings of Joke: Obtaining the vector representation of each joke and adding that as item level information.

Ch.2 System Design & Implementation details

Algorithm considered:

To get the best results we considered multiple algorithms for each system developed as below:

Joke Recommendation System:

1. Ranking Factorization:

A RankingFactorizationRecommender learns latent factors for each user and item and uses them to rank recommended items according to the likelihood of observing those (user, item) pairs. This model cannot be constructed directly. Instead,use graphlab.recommender.ranking_factorization_recommender.create() to create an instance of this model. RankingFactorizationRecommender contains a number of options that tailor to a variety of datasets and evaluation metrics, making this one of the most powerful models in the GraphLab Create recommender toolkit.
[Ref.https://turi.com/products/create/docs/generated/graphlab.recommender.ranking_factorization_recommender.RankingFactorizationRecommender.html]

2. Rating Factorization:

Create a FactorizationRecommender that learns latent factors for each user and item and uses them to make rating predictions. This includes both standard matrix factorization as well as factorization machines models (in the situation where side data is available for users and/or items).
[Ref.https://turi.com/products/create/docs/generated/graphlab.recommender.factorization_recommender.create.html]

3. Item Based Factorization:

This model first computes the similarity between items using the observations of users who have interacted with both items. Given a similarity between item i and j, S(i,j), it scores an item j for user u using a weighted average of the user's previous observations Iu.
[Ref.https://turi.com/products/create/docs/generated/graphlab.recommender.item_similarity_recommender.ItemSimilarityRecommender.html]
Train all these three models to obtain the best model. All of these models take user-item rating matrix as input.

Joke Categorization:

The problem of joke categorization has been posted as a classification problem with all the joke category are classes while the features are derived from jokes:
Glove vectors:-
GloVe is an unsupervised learning algorithm for obtaining vector representations for words. Training is performed on aggregated global word-word co-occurrence statistics from a corpus, and the resulting representations showcase interesting linear substructures of the word vector space:

1.Gradient Boosting
2. Naive Bayes
3. SVM
4. Logistic Regression
5. Nearest Neighbors using Centroid
6. K-nearest Neighbors
7. Ensemble of all above classifiers

Joke-User Bucketing:

Results from recommendations and categorization are fed to bucketing system. Model is trained for users likes/dislikes based on ratings. This trained model is tested on following algorithms:

1. Nearest Neighbors
2. K-means clustering

3. Naive Bayes
4. Random Forest

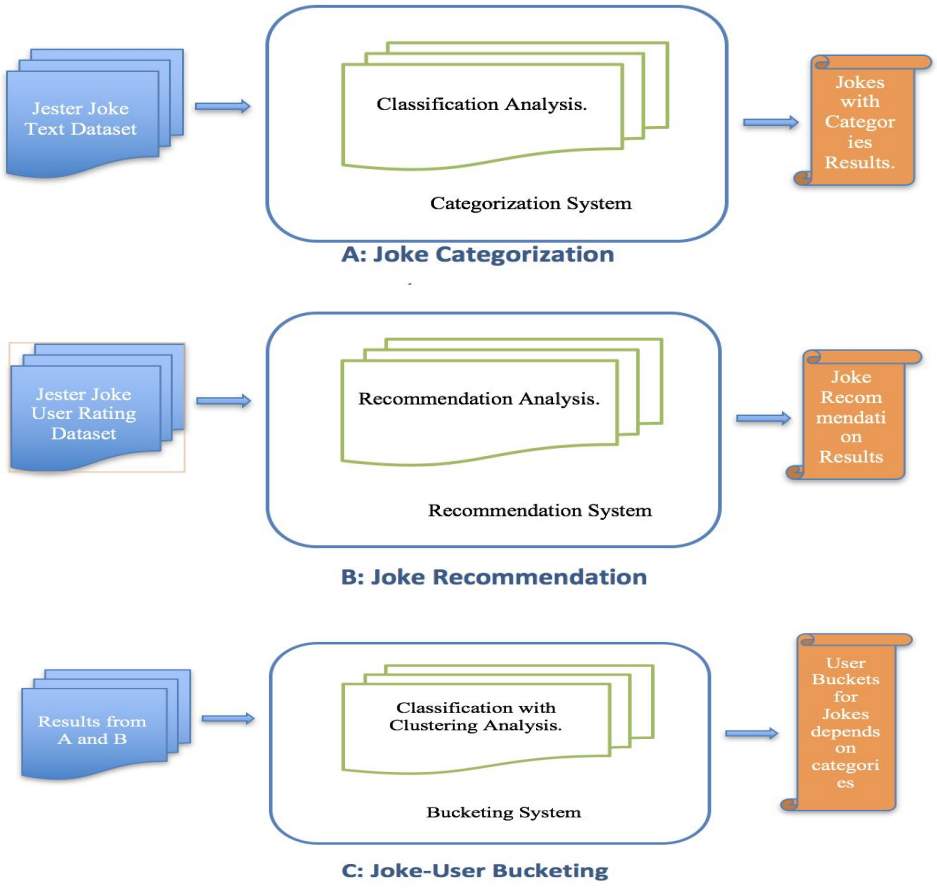
Based on likes and joke categorization, buckets are formed for joke_category->users.
Using this we found the audience for specific joke category.

Technologies & Tools:

1. Python with Jupyter Notebook
2. Python's libraries – scipy, numpy, panda, sklearn and Google News Vector Negative(word to vector generation, ref. https://drive.google.com/file/d/0B7XkCwpI5KDYNlNUTTlSS21pQmM/edit)
3. GloVe for vector representation of words.
4. Github for our repository, regular commits and collaboration.

[https://github.com/surgoku/CMPE-256_project_joke_recommendation]

System Design :



System A, B and C shows the Joke Recommendation System working. Each system works on the algorithms mentioned in earlier sections. Data is preprocessed and collected into needed formats for each system. Results are compared and evaluated for accuracy.

Ch.3 Experiments

Dataset:

- Dataset is taken from Jester (UC Berkley) http://eigentaste.berkeley.edu/dataset/
- jester_dataset_1_1.zip: (3.9MB) Data from 24,983 users who have rated 36or more jokes, a matrix with dimensions 24983 X 101
- Ratings are real values ranging from -10.00 to +10.00 (the value "99" corresponds to "null" = "not rated").
- The first column gives the number of jokes rated by that user. The next 100 columns give the ratings for jokes 01 - 100.
- The text of the jokes can be downloaded here:

jester_dataset_1_joke_texts.zip (92KB) / jester_dataset_2.zip (7.7MB)

Data Processing:

We worked on two datasets User-Joke ratings and Joke Text. Before filling data into algorithm, we needed to process them in appropriate format.

1. **User-Joke Rating**:

From User-Joke rating dataset we formatted data into userid-jokeid-rating matrix. This dataset is then fed to recommendation system

2. **Joke Text Processing**:

Joke text dataset was raw text dataset. We processed it into joke_id and joke_text structure. Joke_text had many unwanted words and characters. We processed the text for following conditions to get the clean joke text.

- Remove special characters such as html tags and obtain only words
- Remove stop words

3. **Joke-User Bucketing**:

We needed to have categorization for joke and user likes and dislikes for specific category. Data is processed for having a userid-jokecategory-jokeid-rating structure.

Methodology followed:

Recommendation System:

Train all below three models to obtain the best model. All of these models take user-item rating matrix

1. Ranking Factorization
2. Rating Factorization
3. Item Based Factorization

Steps:

1. Split the data in train and validation using Graphlab library
2. Define all these models (here in: recommendation_modules method)
3. Train these models for various latent factors: Factors required in factorization
4. Optimize for RMSE
5. Plot and evaluate various models to obtain which model and factor performed the best.

Joke Categorization:

This is the Code for predicting the category of joke from the following categories.

We have pre-labelled the jokes data (150 jokes) into the mentioned classes. Following is the distribution of the categories:

Category	Percentage
1. Animals	5.33
2. Technology	12.66
3. Doctor	4.67
4. Man	7.33
5. Politics	10
6. Relationship	8.68
7. Religion	0.67
8. School	0.67
9. Food	5.33
10. Others	38.66

Approach for Joke Categorization:

Features for Jokes:

1. Obtain key words for each joke
2. Obtain Glove vectors (similar to Word2Vec) from pre-trained based on Wikipedia data of 300 dimension for each word and then averaging out for the entire joke.
3. These averaged out 300-dimensional Glove vector for joke is used as feature for classification

Training:

The problem is posed as a multi-class classification problem with 10 classes. Since the data is of small size, only 10% of the jokes are used for testing while
remaining 90% jokes are used for training. Models were trained using the algorithms mentioned above.
Evaluation metrics is generated for accuracy.

Joke-User Bucketing:

Steps:

1. Obtain the Joke categorized dataset
2. Retrieve jokes for the specified category
3. Classify/cluster user likes and dislikes for the specific jokes under specified category.
4. Show users liking the jokes belonging to specified category.

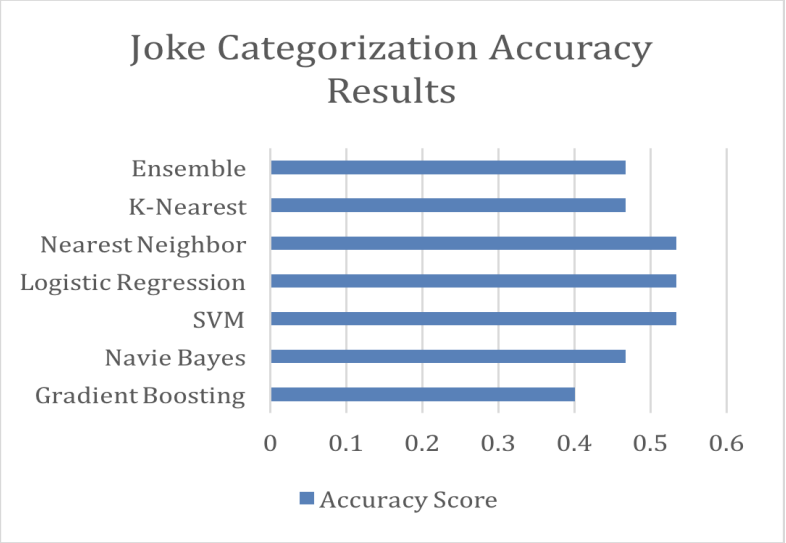
Training:

Dataset is divided into 90% training and 10% testing data.

1. User likes and dislikes are decided from ratings.
2. Model is trained for user likes and dislikes.
3. Classification in 0:dislike, 1 :like is done using algorithms mentioned.
4. From the results and categorization output, joke-category to user bucketing is done.

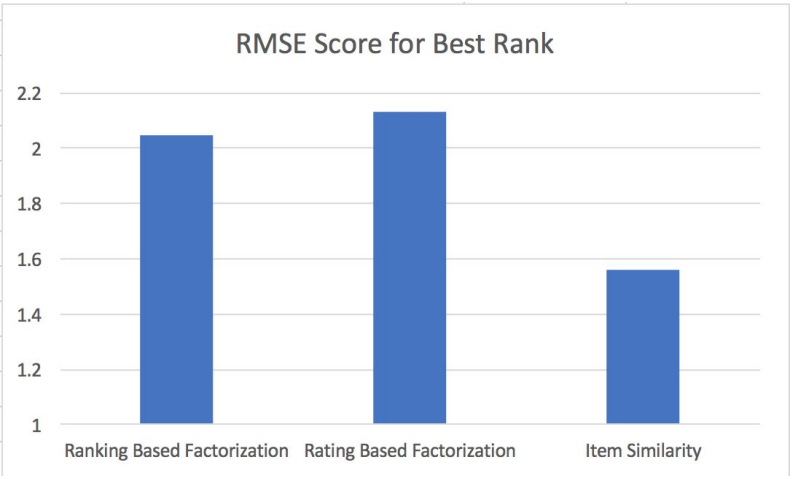
Evaluation:

Evaluation of Joke Categorization Accuracy Results



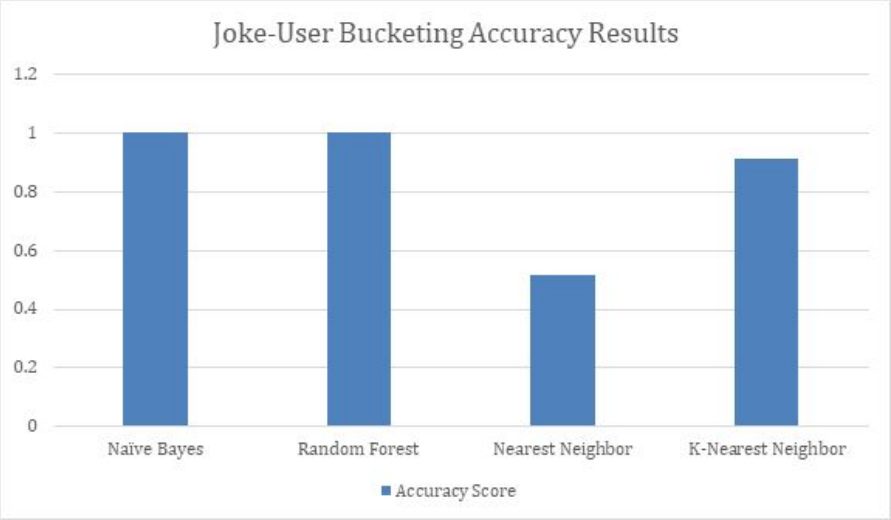
Even though Nearest Neighbors and LR gives best performance, however they often bias towards "other" class. On qualitative analysis, Naive Bayes and Gradient boosting does better generally. KNN does better when there are enough samples for a category, however it is biased towards other class which is a popular class making the accuracy to be high.

Evaluation of Joke Recommendation Accuracy Results



Among the three adopted approaches, Item Similarity gives the best result on RMSE on test data. Therefore, we further improved Item Similarity based recommendation with joke level features such as joke category and Joke Glove vectors.

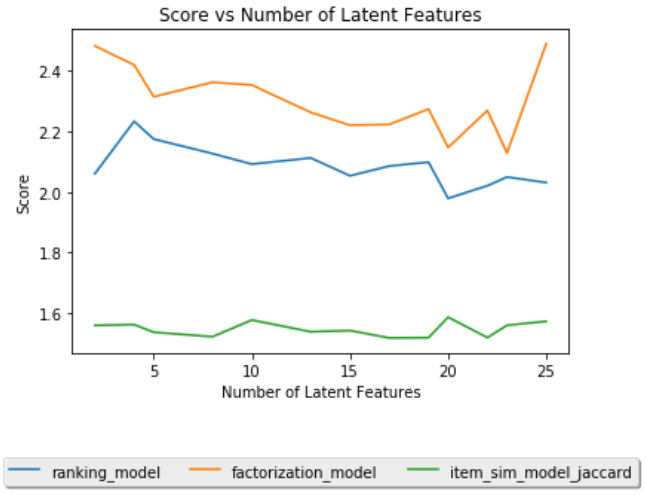
Evaluation of Joke-User Bucketing



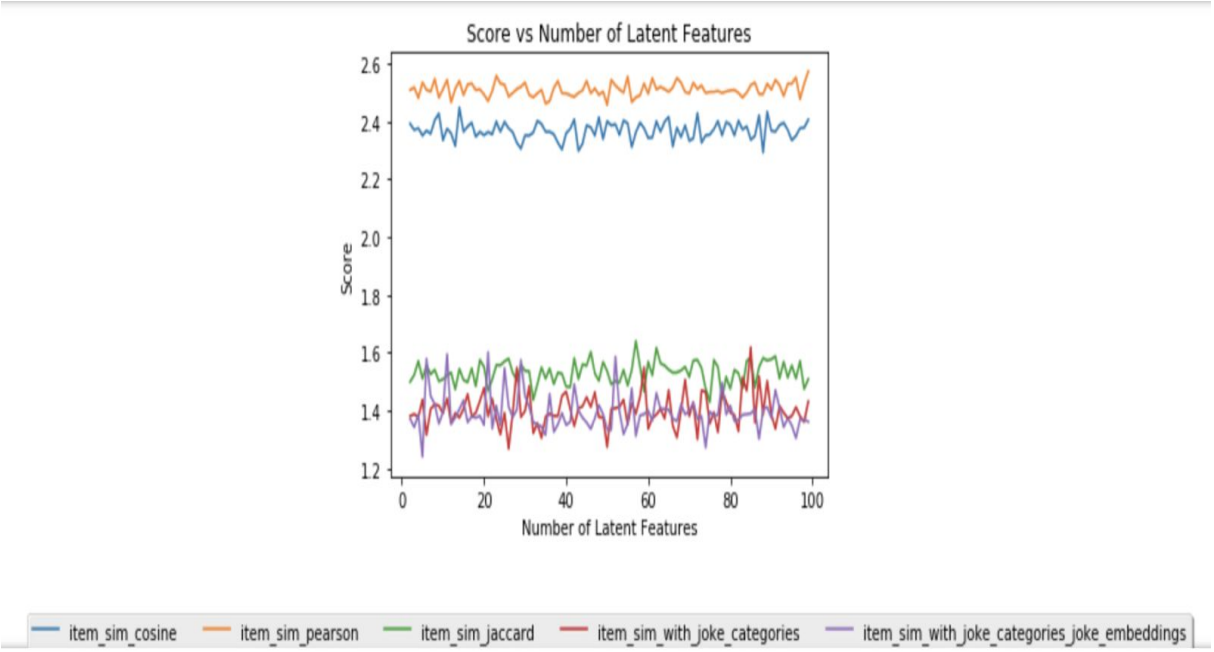
Naïve Bayes and Random Forest gave the accurate results for likes/dislikes prediction. KNN results were close to accuracy. Nearest Neighbor did not perform well. Finally further bucketing is performed based on best model i.e Naïve Bayes.

Analysis Results:

As we analyze joke categorization, recommendation and user bucketing for joke category, we observed that:-



From the figure mentioned above, it is clear that Item Similarity gives least RMSE. Therefore, we further improved Item Similarity based recommendation with joke level features such as joke category and Joke Glove vectors.



We tried Item similarity based recommendation in five settings:

1. Item Similarity using Pearson
2. Item Similarity using Cosine
3. Item Similarity using Jaccard
4. Item Similarity using Jaccard and Joke Categories
5. Item Similarity using Jaccard and Joke Categories and Joke level Glove Features.

We observed that as we add more item level information, the model further improves and does much better compared to the baseline models. Hence, the model with Joke categories and Joke level Glove Features gave us the best result.

Naïve Bayes
Total Number of Users for bucket: politics 5161
Total Number of Users for bucket: animal 3159
Total Number of Users for bucket: doctor 3036
Total Number of Users for bucket: others 25020
KNN
Total Number of Users for bucket: animal 3311
Total Number of Users for bucket: politics 5503
Total Number of Users for bucket: doctor 3197
Total Number of Users for bucket: others 26425

From the Joke-User Bucketing system, we analyzed that though KNN is closer to accuracy it has different set of users and count. Naïve Bays and Random Forest has same accuracy, both model predicts the same and has similar bucketing.

Ch.4 Discussion & Conclusions

We build a Joke Recommendation system using Jester dataset, which has over 1 million ratings by around 80,000 users on 150 jokes. We explored this dataset and performed a thorough analysis on various kinds of recommendation systems for this dataset. Furthermore, we performed an extensive evaluation of various techniques in conventional recommendation system. We tried the following techniques: 1. Rating Based Matrix Factorization (MF) 2. Ranking Based MF 3. Item Similarity based MF. We observed that Item Similarity based RS performed the best. We furthermore expanded this analysis by manually labelling each joke into a category such as: Politics, School, Technology, etc. We used this item level information as features and further expanded our RS. We observed that the results were significantly improved by adding this information. With this motivation, we further on added Joke level features such as: Joke level Glove Embedding vectors (Joke Vector Representation), which we use as features. This setting gave us the best result. We furthermore, built a model to predict the category of a given joke in the predefined 10 categories used in the RS. With just 150 samples of Jokes, we got around 50% accuracy in a cross-validation setting. We therefore can use this model to predict the category of a new joke, which can be used to recommend new jokes to the users. On the same lines, we did user-joke-category bucketing, wherein we obtained the likes and dislikes for each user, i.e. the favourite joke category for each user. This can also be used to recommend new jokes in each category to the users.

Difficulties faced: It was hard to obtain joke level features initially, since there are only 150 jokes and it is hard to derive word level features such as n-gram.

Things that worked: To address the difficulty mentioned above, we used GloVe word vectors, which worked.

Things that didn't work well: SVM Algorithm for Joke-User bucketing halted the system.

Future Work: We propose using item level information in Ranking and Rating based Recommendation System to see if it improves there as well just the way it improved item similarity based RS.

Ch.5 Project Plan

Task	Assignee	Justification (if any)
Project Topic/ Dataset discussion	All	
Project analysis	All	
Joke Categorization	Surbhi/Navneet	
Joke recommendation	Surbhi/Navneet	
Joke-User bucketing	Amruta	
Project report/presentation	All	