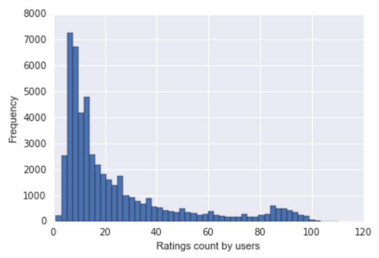
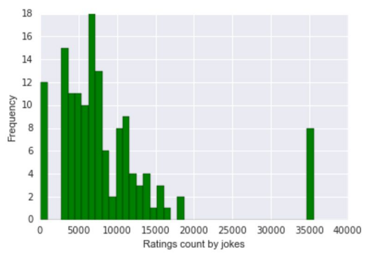
Jokes Recommendation Case Study

By: Jeffrey Li, Chris Seymour, Dawn Lu, Lihua Ma

I. Data Processing:

In order to assess how sparse our data is, we first looked at the distribution of rating counts by user and by joke. We found that about 200 users had fewer than 3 ratings, so pruned them from the training data to improve model quality. On the other hand, the minimum number of ratings of a joke was above 100, so we kept all jokes in the data.





Next, we wanted to parse the joke text to feature engineer side data. The joke text has random characters that makes the data unacceptable for analysis. To beautify our data, we stripped the random characters from the text while creating a separate column for ID.

II. **Recommender Models**: Since we were using explicit data, and we cared more about *ranking performance* over predicting rating accurately, we decided to go with the Rank Factorization Recommender Model.

Rank Factorization Recommendation:

RMSE on validation: 5.1

Factorization Recommender: We actually got a lower RMSE on the training set for Rank Factorization compared to Factorization Recommender(2.6 vs. 2.98, which is unexpected. But we're guessing this is because did not use the score metric provided to us.

Item Similarity Recommendation: *RMSE on validation:5.51*

Item Content Recommender: *RMSE on validation :5.53*

Regularization: *Improved RMSE by 0.6*

Cross-Validation: We used the cross-val score to determine which of the above models was the best with our data, and we went with the Rank Factorization Recommender.

Note: We used RMSE has our error metric, not the score provided to us in the example files.

III. Additional Features:

Additional 'User Rating Average' Feature:

Natural Language Processing: We decided to use NLP on the Joke Text data to find signal. Couple ideas we had:

1. **Find Word Counts**: To start NLP, we wanted to see if there was any signal within the word counts of each joke. We created a new column on the dataset called 'Word Counts.' **RMSE on validation: 5.1 --> RMSE on validation: 4.6**

Stuff We Didn't Get to:

2. **Find High-Frequency Words for different ratings:** Our idea was to examine each rating and find the highest term frequencies within each rating. We'd extract the highest-frequency words from the jokes for each rating, and use that term frequency to predict the rating for that joke.

3. **NMF Topic Modeling:** Create arbitrary topics for each joke text. Create new features for each topic, with a binary data-type, signalling either 1 or 0 if the NMF predicts a joke to be in a certain category. Use topic categories to predict user rating for joke.

FINAL RMSE SCORE (VALIDATION): 4.6

IV. Next Steps

- Deeper Dive into NLP created features
- Use the score provided to us