Group: Zhifan, Vineet, Chris VD, Stefan Decker

Initial impressions of the data / initial issues

After doing some exploratory analysis on the user-joke rating dataset, we made some initial observations of our data:

	user_id	joke_id	rating
counts	1218325	1218325	1218325
uniques	50692	141	641
missing	0	0	0
missing_perc	0%	0%	0%
types	numeric	numeric	numeric

Table 1: Summary Table of Dataset

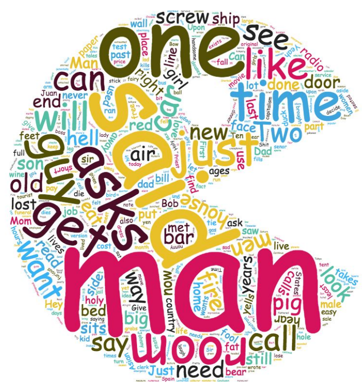

Figure 1: Word Cloud of 150 Jokes

There are way more users than jokes in the dataset
There are ~1.2M ratings, ~50K users, and only ~150 jokes in total (Table 1). Initially this seems great as overfitting (probably) won't be a big issue.

Obfuscated scoring metrics
Because our final test scoring metric only values the ranking of the recommended jokes, and not the absolute accuracy of the predicted values, we chose to use a model that is optimized to be accurate only on the top rated jokes for each user. This means RMSE as a metric is not very meaningful to us, and we need to use a different metric for our training test. The GraphLab recommender has recall score attributes which are supposed to be the recall on the top rank jokes, but we do not know how it is calculated, so there is a bit of black box going on here.

Validation and the cold start problem
Cross validation is one of the only ways to test our model's performance. Unfortunately, when you split the data into a test and train set every time you test the model it is working on a cold start problem.

Humor is Subjective
It's very hard to quantify the nature of humor, people can have subjective opinions on whether or not a specific joke is good and even experts and professionals cannot explain why a joke is funny.

Methods

Feature engineering

To improve our recommendation model we engineered the length of the jokes as side feature. We realized that the jokes were truncated, so this feature was not accurate. However, this did reduce the recall of our model slightly. We assume that this is due to overfitting.

Data pruning

We took out users with only a few ratings which reduced our model score. It turned out, that the less users we took out, the better our model did so we didn't use data pruning in our final model.

Model selection

We tested 4 different ranking models with the following results (on a test/train split):

	RMSE	**Recall (10 jokes)**
Popularity_recommender	4.92	0.21
Factorization_recommender	4.61	0.16
Item_similarity_recommender	5.49	0.42
Ranking_factorization_recommender	6.94	0.43

Table 2 Model Performance Comparison

The popularity and factorization recommenders made a good job predicting the user ratings but the ranking factorization recommender outperformed it by identifying the top ten jokes which is what we are interested in.

Results

Our model achieved a 52.74% recall rate with the validation set. Unfortunately, with the issues stated above we aren't sure how accurate this metric is so we aren't sure what to make of it. What we do know is that the RMSE (6.96) was totally useless to us given the goals / methodology of our model so we didn't use it as a metric.

Next steps

Given more time, there are several directions we could taken to improve our model. First, we could actually spend time researching how graphlab calculates its recall metric and if it appropriate to use in this instance. We could also spend more time researching other useful metrics to rate recommenders. If we find no metrics that explain success better than RMSE, we could have spent more time creating appropriate validation sets to perform cross-validation without running into the 'cold-start' problem inherent in testing recommender systems. Additionally, we could have spent more time performing sentiment analysis on the jokes themselves. We could have run tf-idf or other NLP methods to create topic groupings and other feature columns that could have given our model better ranking performance. We also didn't utilize any domain knowledge to improve our model predictions or incorporate item-item relationships (are jokes sarcastic? etc.)