**Final Project Proposal – Jithendra Seneviratne**

This project hopes to use the both the [Beer Recommender Data](https://snap.stanford.edu/data/web-BeerAdvocate.html) as well as the [Jester](http://eigentaste.berkeley.edu/about.html) dataset to run various models and evaluate their performance. We will use an AWS EC2 instance to gain the processing power needed to manipulate very large sparse matrices. We’ll also use Apache Spark to compare efficiency of distributed systems.

**Objectives**

* Use cloud computing for efficiency
* Use different models in the Surprise package to decide ideal algorithm (using RMSE)
* Evaluate models again on different levels of sparsity
* Look at performance metrics other than RMSE
* Compare model performance with Apache Spark implementation and compare against efficiency of distributed systems.

1. **Cloud Computing: AWS Machine Specs**

The machine is a T2 Extra Large machine with fore cores and 16GB of RAM.

1. **Using the Surprise Package to Evaluate Models**

This exercise uses the Surprise Package to evaluate different models and then run grid-search cross validation to get the best model parameters.

**Steps:**

1. *Create Reader and Load Dataframe into Readable Format*
2. *Compare Models using a Three-Fold Cross Validation Score*
3. *Pick Best Model and Compare Against Baseline Predictor*
4. *Use on Holdout Testset*
5. *Look at Best Predictions and Worst Predictions on Testset*
6. *Run Grid Search on Tuning Parameters*
7. *Fit Model again on Best Hyper-Parameters*
8. *Get Predictions for a Given User*
9. **Managing Sparsity**

All models don’t perform the same given sparsity of data. For example, we know that SVD based approaches perform better when sparsity is lower; i.e., when information present is greater, SVD performs better than the baseline method, otherwise, maybe not. We can see from the histograms below that the levels of sparsity are different. It’ll be interesting to see which models perform better on each dataset given the different levels of sparsity.

Jester Dataset Histogram Beer Dataset Histogram

### /var/folders/yp/75nvl29s2kv8v3p49wf4165c0000gn/T/com.microsoft.Word/Content.MSO/EDE458CA.tmp/var/folders/yp/75nvl29s2kv8v3p49wf4165c0000gn/T/com.microsoft.Word/Content.MSO/52CE365C.tmp

### Number of Reviews. Number of Reviews

### Metrics Other Than RMSE

### We'll explore ideas such as Personalization and Coverage below. Documentation for the [Recmetrics package](https://towardsdatascience.com/evaluation-metrics-for-recommender-systems-df56c6611093) suggest that maximizing personalization and coverage is desirable.

#### **Coverage**

#### This is the % of recommendations the model can make on the testset.

#### **Personalization**

#### This is the level of personalized recommendations the algorithm spits back. This might be of particular value to users, as unique and accurate recommendations are better than simply accurate recommendations.

#### **Tuning Hyper-Parameter**

#### We'll use the minimum number of ratings the user has provided as parameter and retrieve RMSE, Coverage and Personalization for different filtered models

### Metrics Other Than RMSE

### Finally, we’ll look at the Pyspark implementation of ALS to compare distributed computing to regular local implementation.