UC San Diego

JACOBS SCHOOL OF ENGINEERING

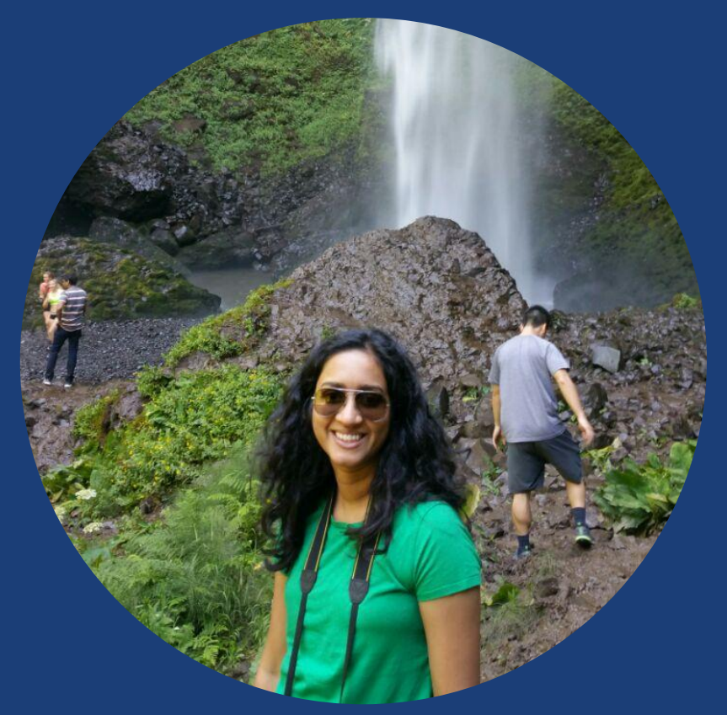
Sharknado

Advanced Recommender Systems

Advisors: Julian McAuley, Ilkay Altintas

Team:

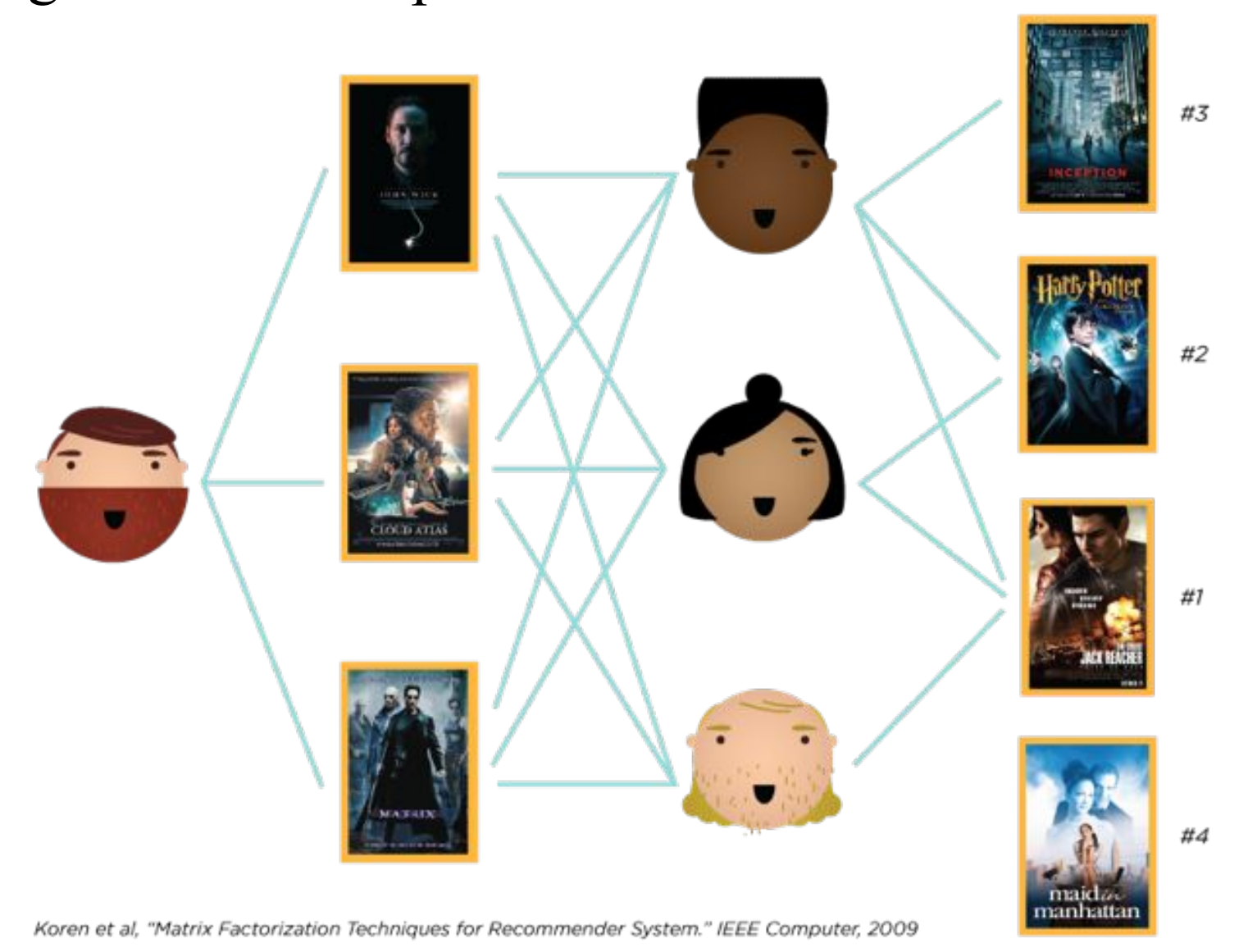
Alex Egg, Peyman Hesami, Deepthi Mysore Nagaraj, Julius Remigio



Masters of Advanced Study in Data Science and Engineering

Introduction

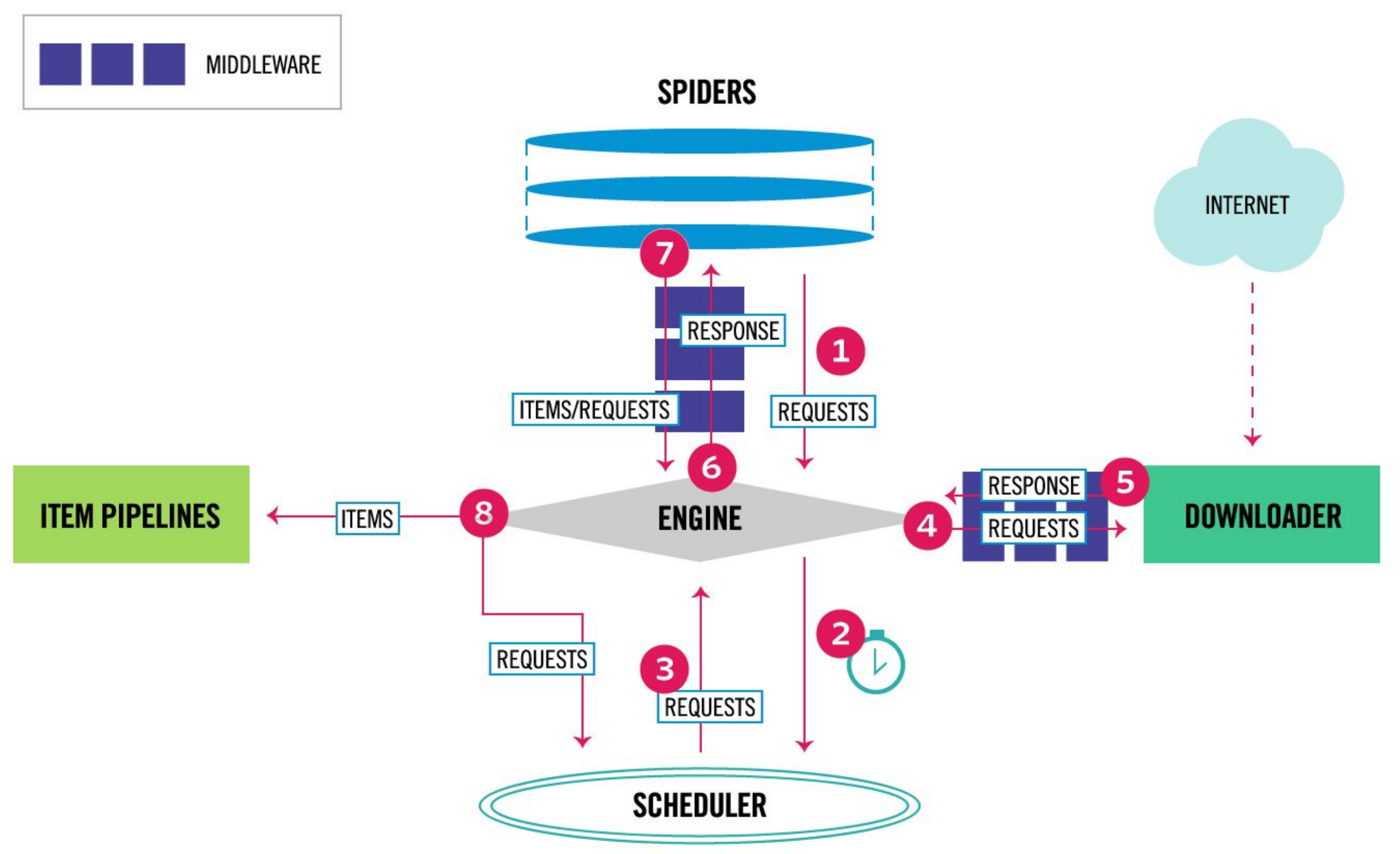
There are two traditional methods in recommender systems: Content-based and Collaborative Filtering Systems. Each have tradeoffs: content-based systems suffer the Pigeon Hole Problem, which Collaborative Filtering System suffer the cold start problem. While Collaborative Filtering systems generally perform better in practice, the cold start problem remains.. Through a hybrid combination of Content-based systems and Collaborative Filtering we propose to build a recommender w/ the performance characteristics of Collaborative filtering while mitigating the cold start problem.



Koren et al. "Matrix Factorization Techniques for Recommender System" IEEE Computer, 2009

Data Collection

Amazon data was collected using a specially designed web scraper. Over 142.8 million product reviews were scraped including their associated product pages and images.



Amazon has a stringent anti-scraping policy and employs CAPTCHAs to discourage automated data harvesting. Dynamic User Agent strings were employed in combination with traffic data from w3schools.com was used to model page requests that reflect real-world traffic distributions. In addition, session isolation and clever cookie handling techniques were employed to impersonate a brand new Amazon user with each page request. Also, all traffic was routed through a pool of proxies to for an additional layer of anonymization.

Data Preparation

Different non visual features were included in the model. Namely:

1. Price
2. Brand
3. Product features

Price has a huge impact on the sales of a product. To capture this influence in the model, price was quantized into 10 buckets.

Brand affinity of a customer is an important factor that influences his purchases. Also, customers who like a particular brand tend to like other similar brands. So 1992 brands were included as one-hot encoded binary vector.

Product features are some important non-visual features extracted from the product title field (as bigrams). Most frequently used 4525 bigrams were used and included as one-hot encoded binary vector.

Modeling

Model ranking of user's preference (rating) to products can be done as:

$$x_{u,i} = \alpha + \beta_u + \beta_i + \gamma_u \cdot \gamma_i$$

Where:

α -- popularity of item, β_u -- user tendency to rate things above the mean β_i -- item tendency to receive higher ratings than others, $\gamma_u \cdot \gamma_i$ -- 'compatibility' between the user u and the item i

This can be optimized using Bayesian Personalized Ranking (BPR) method. This model can also be extended to include visual (VBPR) and non-visual features (NVBPR) or a hybrid of both (HBPR) as:

$$x_{u,i} = \alpha + \beta_u + \beta_i + \gamma_u \cdot \gamma_i + \theta_u^T(\mathbf{E}\, f_i) + \beta'^T f_i$$

Where:

$\theta_u^T(\mathbf{E}\, f_i)$ -- 'compatibility' between the user u and the visual features of item i and $\beta'^T f_i$ -- users' overall opinion toward the visual appearance of item i

Performance Evaluations

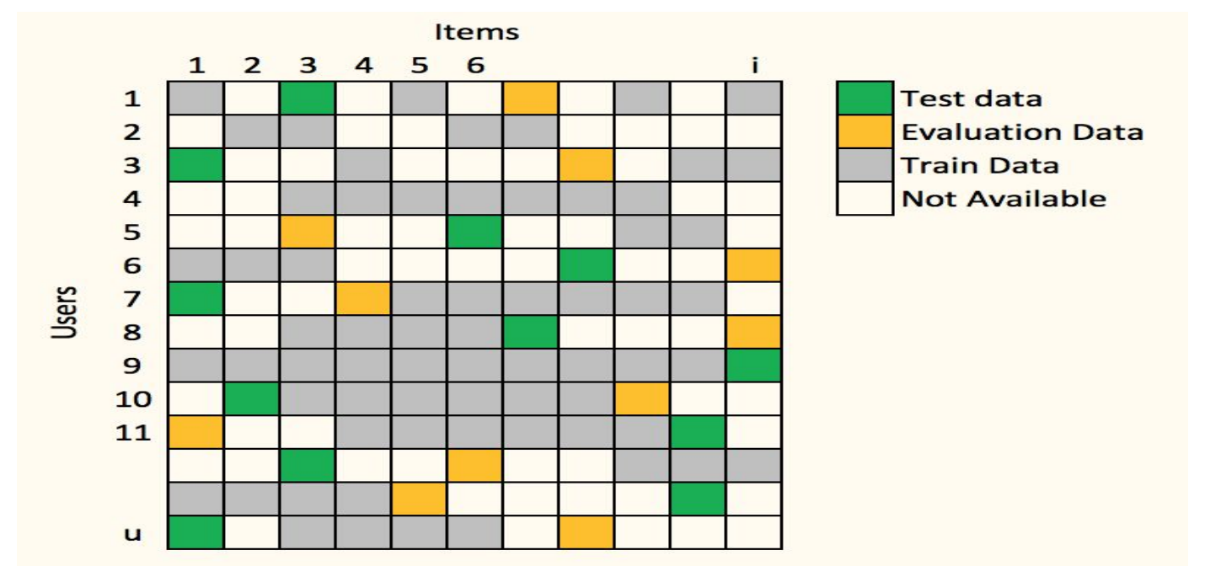
We use the Area under the curve to evaluate our models:

$$AUC = \frac{1}{|\mathcal{U}|} \sum_u \frac{1}{|E(u)|} \sum_{(i,j) \in E(u)} \delta(\widehat{x}_{u,i} > \widehat{x}_{u,j})$$

Where:

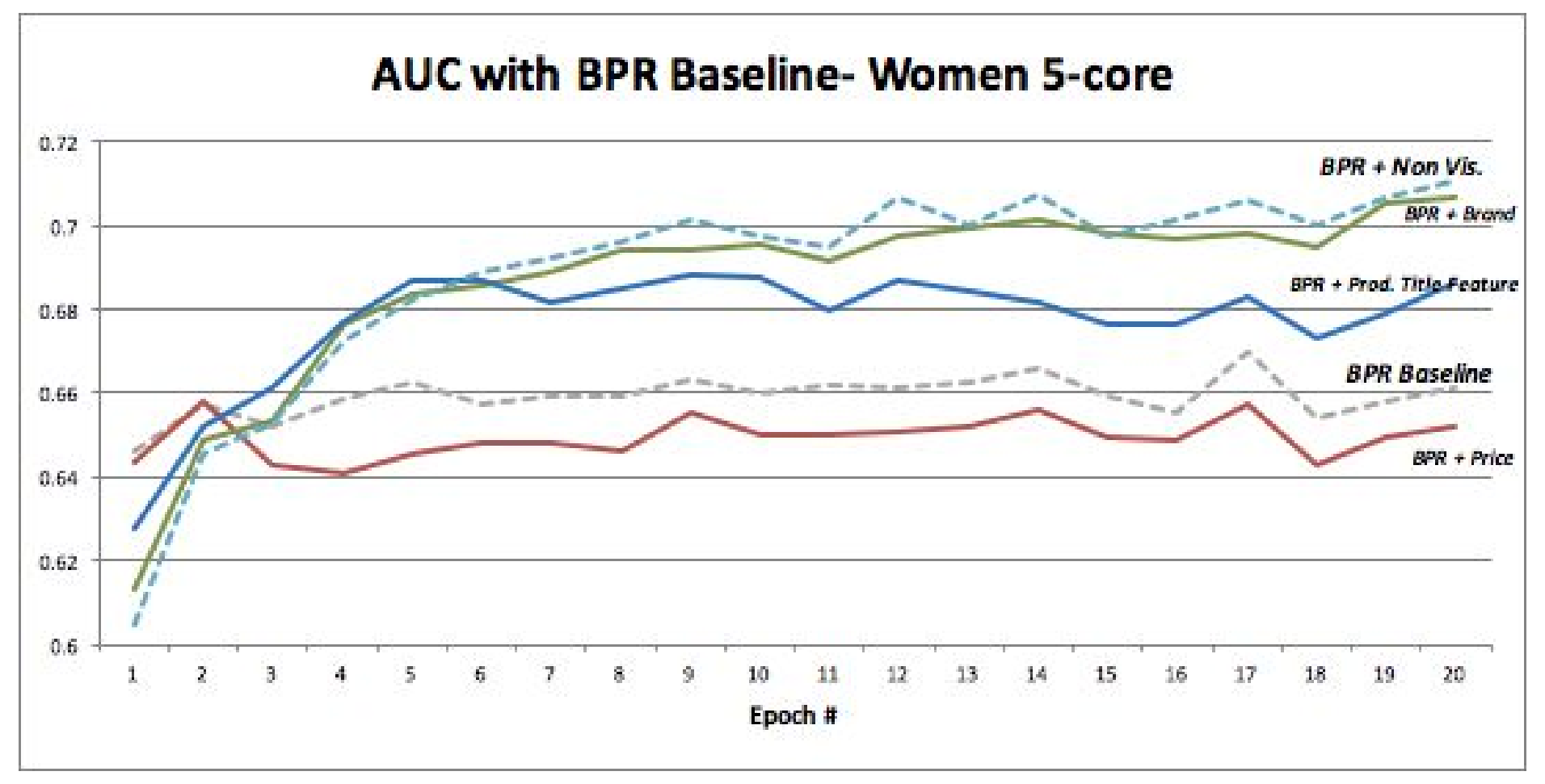
$$E(u) = \{(i,j) | (u,i) \in \mathcal{T}_u \wedge (u,j) \notin (\mathcal{P}_u \cup \mathcal{V}_u \cup \mathcal{T}_u)\}$$

The train/test/eval split is illustrated as below:

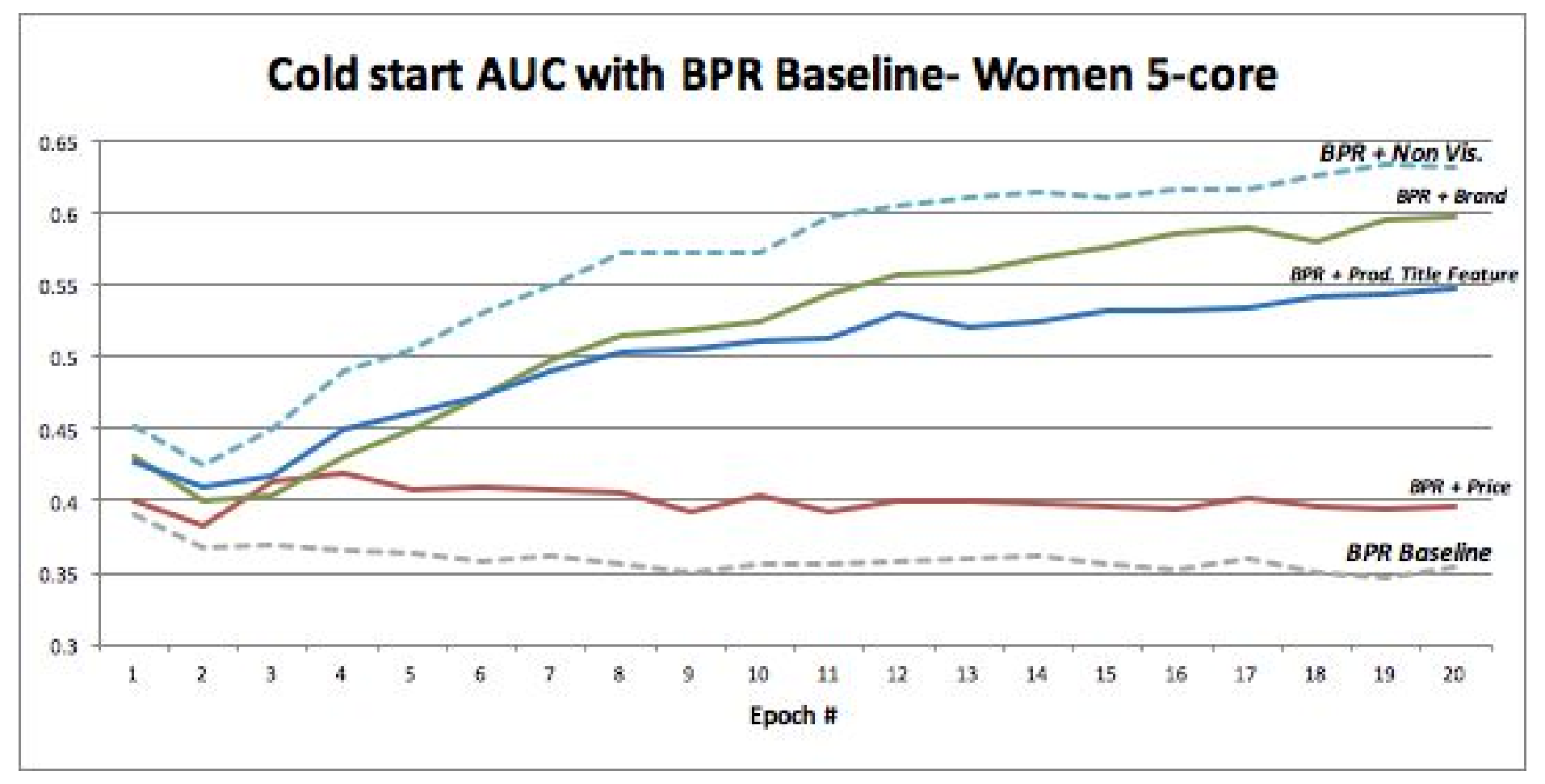


Findings

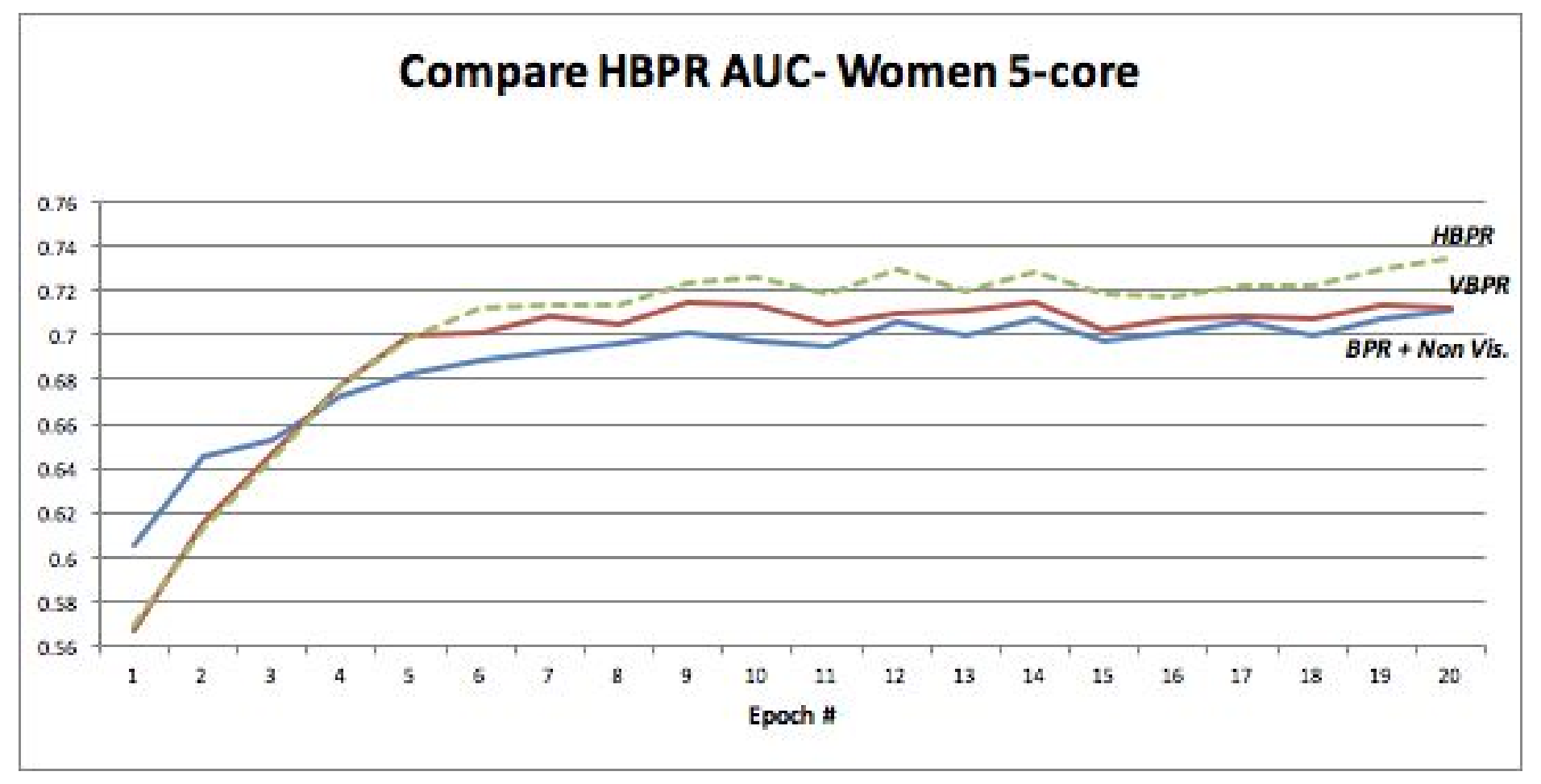
We ran our model on Amazon Women dataset and observed around 10% improvement in NVBPR AUC (compared to baseline BPR):

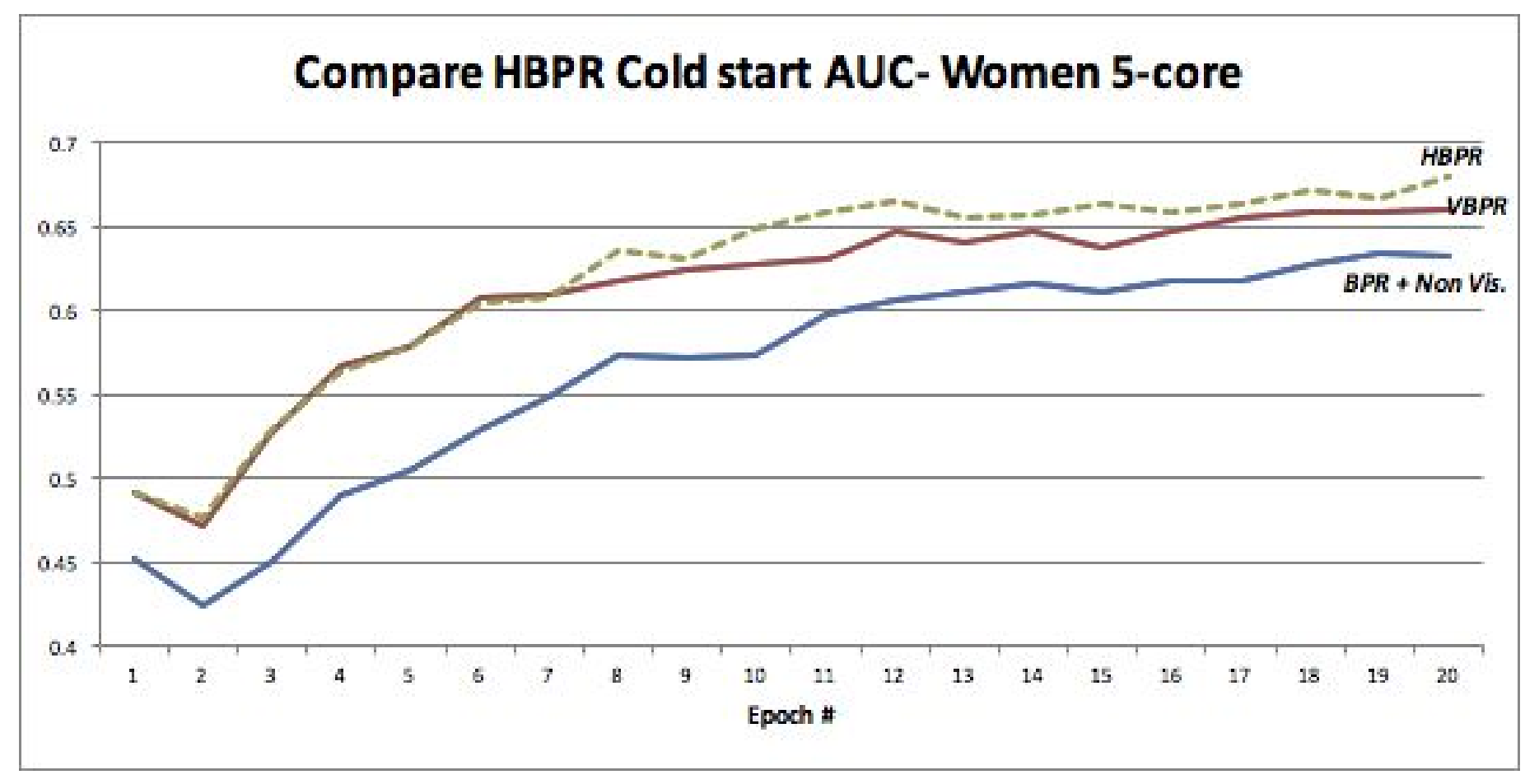


The NVBPR also addresses the cold start issue as it can be seen from the AUC curves:



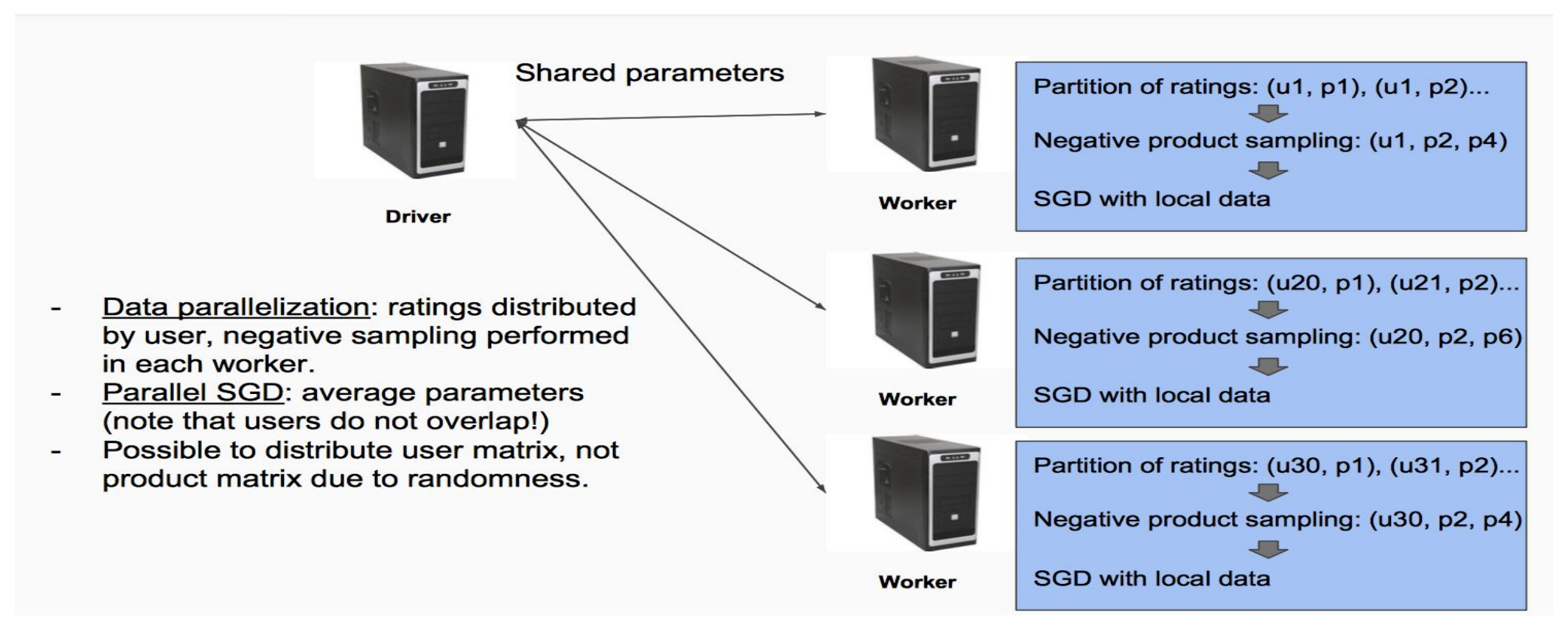
We ran our HBPR model (VBPR+NVBPR) on Amazon women dataset see a persistent gain compared to the VBPR for all items as well as cold items as illustrated in the below AUC curves:





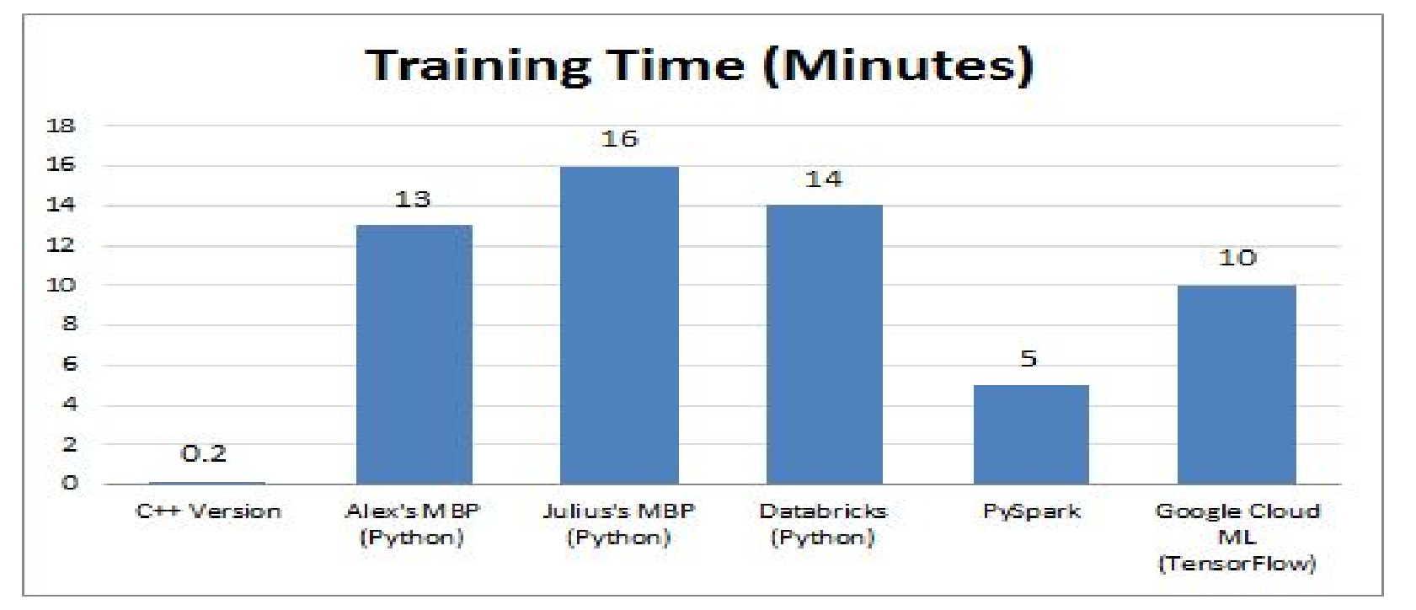
Scalability

Our first attempt at implementing the baseline model (BPR) was in pure Python which was not scalable due to long training times. We then examined Spark using the following architecture:


Shared parameters
Driver
Worker
Partition of ratings: (u1, p1), (u1, p2)...
Negative product sampling: (u1, p2, p4)
SGD with local data
Partition of ratings: (u20, p1), (u21, p2)...
Negative product sampling: (u20, p2, p6)
Partition of ratings: (u30, p1), (u31, p2)...
Negative product sampling: (u30, p2, p4)

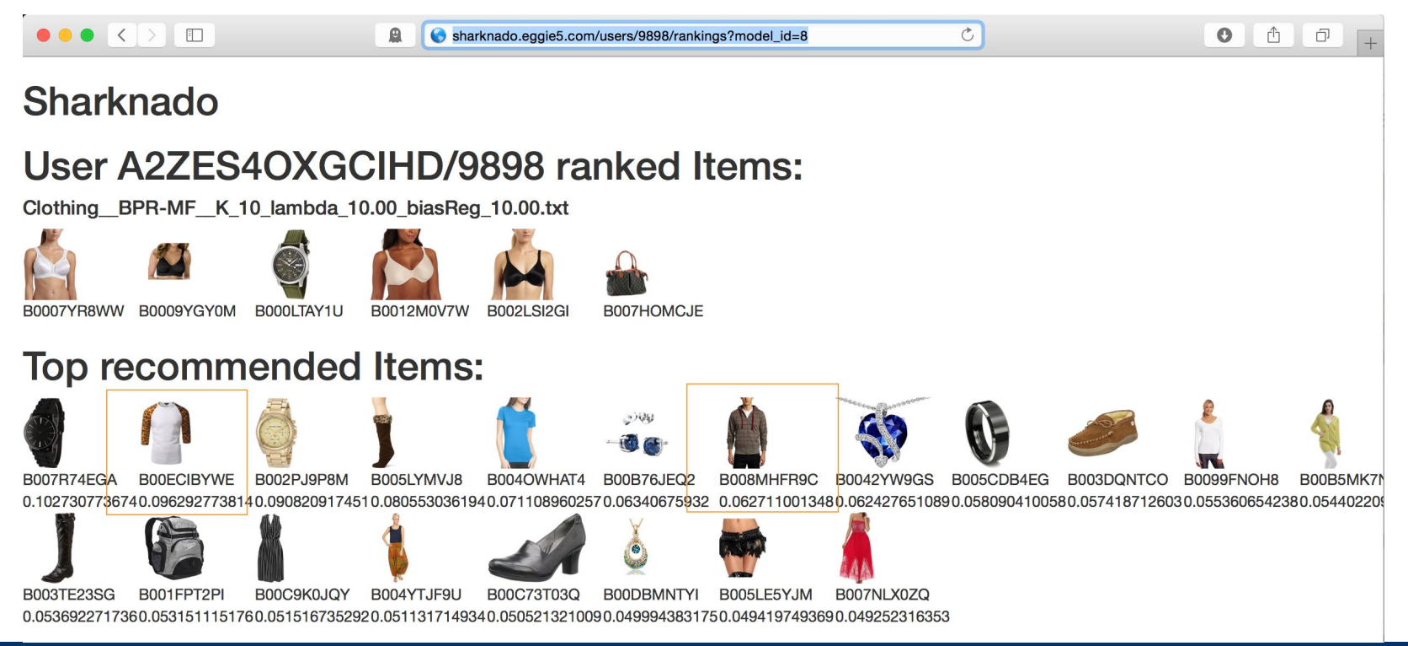
- Data parallelization: ratings distributed by user, negative sampling performed in each worker.
- Parallel SGD: average parameters (note that users do not overlap!)
- Possible to distribute user matrix, not product matrix due to randomness.

The Spark was very fast but the complexity of the implementation was high. We then examined Theano and TensorFlow and based on training time and the complexity of the implementation we picked TensorFlow as our platform. Here are the training times for different platforms:



Final Product

A web based tool was created to demonstrate the ability of our Advanced BPR model. It visually displays product recommendations for a given user based on visual and non visual preferences.



sharknado.eggie5.com/users/9898/rankings?model_id=8

Sharknado

User A2ZES4OXGCIHD/9898 ranked Items:

Clothing__BPR-MF__K_10_lambda_10.00_biasReg_10.00.txt

B0007YR8WW B0009YGY0M B000LTAY1U B0012M0V7W B002LSI2GI B007HOMCJE

Top recommended Items:

B007R74EGA 0.102730773674
B00ECIBYWE 0.0962927738 14
B002PJ9P8M 0.090820917451
B005LYMVJ8 0.080553036194
B004OWHAT4 0.071108960257
B00B76JEQ2 0.06340675932
B008MHFR9C 0.062711001348
B0042YW9GS 0.062427651089
B005CDB4EG 0.058090410058
B003DQNTCO 0.057418712603
B0099FNOH8 0.055360654238
B00B5MK7N 0.05440220

B003TE23SG 0.053692271736
B001FPT2PI 0.053151115176
B00C9K0JQY 0.051516735292
B004YTJF9U 0.051131714934
B00C73T03Q 0.050521321009
B00DBMNTYI 0.049994383175
B005LE5YJM 0.049419749369
B007NLX0ZQ 0.049252316353

Conclusion and Future Work

Conclusions:

- Visual aspects of items bias users' opinion toward them in some categories
- Non-visual aspects of items bias users' opinion toward them in other categories
- Combination of these two can help build a performant and generic recommender system
- Scalability is crucial in recommender systems

Future Work:

- Tuning the current model
  - # of quantized levels of price
  - # of elements to use from product description
  - Use purchase frequency of a brand for a user
- Incorporate item grouping (clustering similar items)
- Temporal dynamics to capture drifting price/fashion tastes over time