Optimizing Similar Item Recommendations in a Semi-structured Marketplace to Maximize Conversion

Yuri M. Brovman
eBay Inc., New York City, USA
ybrovman@ebay.com

Marie Jacob
eBay Inc., New York City, USA
marijacob@ebay.com

Natraj Srinivasan
eBay Inc., New York City, USA
nsrinivasan@ebay.com

Stephen Neola
eBay Inc., New York City, USA
sneola@ebay.com

Daniel Galron
eBay Inc., New York City, USA
dgalron@ebay.com

Ryan Snyder
eBay Inc., New York City, USA
rysnyder@ebay.com

ABSTRACT

This paper tackles the problem of recommendations in eBay's large semi-structured marketplace. eBay's variable inventory and lack of structured information about listings makes traditional collaborative filtering algorithms difficult to use. We discuss how to overcome these data limitations to produce high quality recommendations in real time with a combination of a customized scalable architecture as well as a widely applicable machine learned ranking model. A pointwise ranking approach is utilized to reduce the ranking problem to a binary classification problem optimized on past user purchase behavior. We present details of a sampling strategy and feature engineering that have been critical to achieve a lift in both purchase through rate (PTR) and revenue.

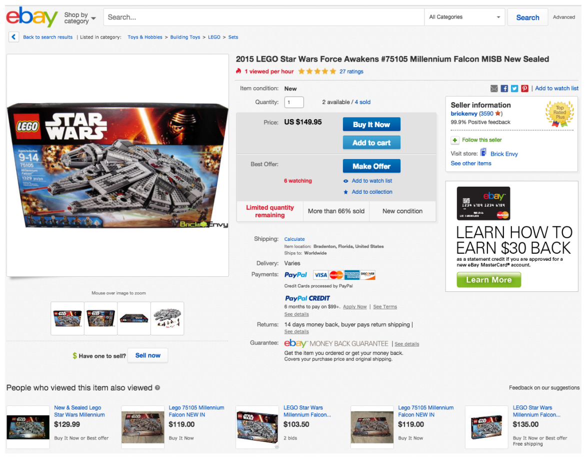
Keywords

e-commerce; recommender systems; machine learning; learning to rank

1. INTRODUCTION

Recommender systems in e-commerce have been extensively studied over the last few decades. Recommendations drive a considerable portion of site revenue, and ensure that users stay engaged with content for as long as possible. Unlike general marketplaces such as Amazon and Walmart, which offer warehouse products in a documented catalog, the eBay marketplace offers more diverse listings ranging anywhere from a new iPhone (a specific product with structured data attributes) to offhand antique items with no known characteristics. With over 800 million active listings at any given time as well as over 150 million active buyers, this semi-structured marketplace exhibits a heavy-tailed distribution of items – a large number of listings are unique, unidentifiable items that are popular among niche buyers. In addition, multiple item conditions and selling formats make serving relevant recommendations significantly harder.

RecSys '16 September 15-19, 2016, Boston , MA, USA
© 2016 Copyright held by the owner/author(s).
ACM ISBN 978-1-4503-4035-9/16/09.
DOI: http://dx.doi.org/10.1145/2959100.2959166



ebay
Shop by category
Search...
All Categories
Search
Advanced
Back to search results | Listed in category: Toys & Hobbies > Building Toys > LEGO > Sets

2015 LEGO Star Wars Force Awakens #75105 Millennium Falcon MISB New Sealed
1 viewed per hour ★★★★★ 27 ratings

Item condition: New
Quantity: 1 2 available / 4 sold
Price: US $149.95
Best Offer:
6 watching
Limited quantity remaining | More than 66% sold | New condition
Shipping: Calculate
Item location: Bradenton, Florida, United States
Ships to: Worldwide
Delivery: Varies
Payments: PayPal
Credit Cards processed by PayPal
PayPal CREDIT
6 months to pay on $99+. Apply Now | See Terms
See details
Returns: 14 days money back, buyer pays return shipping |
See details
Guarantee: ebay MONEY BACK GUARANTEE | See details
Get the item you ordered or get your money back.
Covers your purchase price and original shipping.

Seller information
brickenvy (3590 ★)
99.9% Positive feedback
Follow this seller
Visit store: Brick Envy
See other items

LEARN HOW TO EARN $30 BACK
as a statement credit if you are approved for a new eBay MasterCard® account.

Mouse over image to zoom
Have one to sell? Sell now

People who viewed this item also viewed
Feedback on our suggestions

New & Sealed Lego Star Wars Millennium $129.99 Buy It Now or Best offer
Lego 75105 Millennium Falcon NEW IN $119.00 Buy It Now
LEGO Star Wars Millennium Falcon... $103.50 2 bids
Lego 75105 Millennium Falcon NEW IN $119.00 Buy It Now
LEGO Star Wars Millennium Falcon... $135.00 Buy It Now or Best offer Free shipping

Figure 1: Item page showing similar item recommendations above the fold.

In this paper, we highlight several unique challenges in providing high quality item recommendations to users in real time that are specific to the scale and variety of this data. There is limited structured data coverage of items which makes it difficult to utilize specific item attributes. Additionally, many items tend to be short-lived – they surface on the site for one week and are never listed again. Traditional collaborative filtering algorithms [5, 10, 8] are not effective in this environment due to this volatility of inventory and limitation of structured data coverage.

While our recommendations are driven through a large number of channels, including desktop, mobile and emails, we discuss the problem setting of our most viewed placement – the eBay item page, shown in Figure 1. This page shows the details of a *seed* item, with five recommendations shown above the fold. The recommendations are generated based on both similarity between the seed item and the recommended item, and likelihood of purchase[1]. We use a two stage system where we first retrieve a subset of relevant items to the seed item, and then rank the subset of possible recommendations maximizing the chance of conversion. A comparable multi-stage approach where multiple sources were aggregated together to produce improved search results has

[1]Other placements on this site target more diverse recommendations. Past analysis showed that most users look for the same/similar product to the seed in this placement.

been implemented in LinkedIn [2]. The engineering details of the system architecture are discussed in Section 2.

The second stage of the system involves ranking the recommendations to decide which top 5 items to surface to the user. The topic of learning to rank has been widely studied in the field of Information Retrieval (IR) [9, 7] as it applies to ranking results based on a search query. More recently, machine learned ranking (MLR) techniques have been applied directly to recommender systems at Google [12], Netflix [11], and Amazon [4]. The pointwise learning to rank problem in the IR context can be reduced to a classification problem using $x_i = \{query, URL\}_i$ pairs as well as a binary or multi-class relevance label y_i as training input. Assigning the relevance labels to query-URL pairs is typically done with crowdsourcing methods which can be time consuming and cost prohibitive. In the context of recommender systems for this work, the classification problem uses $x_i = \{seed\ item, recommended\ item\}_i$ pairs and implicit user feedback (clicks/purchases) as the binary class label. This way, data collection is continuous and uses real-world conditions. The details of our ranking approach as well as experimental results are presented in Section 3.

We note that although our marketplace is unique, our ranking model is general enough for any domain employing recommender systems that uses a seed item, and can track click and purchase metrics. The model is widely applicable and is not limited to any one product category.

2. ARCHITECTURE

Our backend platform serves approximately 400 million requests every day and is optimized to have a response time of under 200 ms end-to-end. The backend architecture for the recommendation engine is divided into offline and online systems – the offline components indexes incoming eBay data, mines behavioral data and click logs, as well as trains a classifier on historical data to determine likelihood of purchase.

The overall recommendation process follows the paradigm of a search problem, which consists of two stages: **Recall**, which requires retrieving candidate items that might be similar to the given seed item, and **Ranking**, which sorts the candidates according to their probability of being purchased. An overview of the backend architecture can be seen in Figure 2. The input to the algorithm comes as an HTTP request to the merchandising backend (MBE) system with a given seed item. This initiates parallel calls to several services which return candidate recommendations that are similar in some way to the seed. The set of candidate recommendations are then ranked in real time. The output of the system is the top 5 ranked items, which are surfaced to the user.

2.1 Recall

With a rather sparse inventory of known products, eBay's internal product catalog serves only a small portion of similar item recommendations on the item page ($< 10\%$). This sparsity led to a variety of techniques in generating candidate recommendations for a given seed item.
Product catalog: Although the majority of items on eBay are not tagged with a specific product ID, the distribution of product coverage tends to vary significantly across different categories. Books, for instance, usually has a high percentage of items which are known products due to identifiers such as ISBNs. If the seed item is tagged with a particular

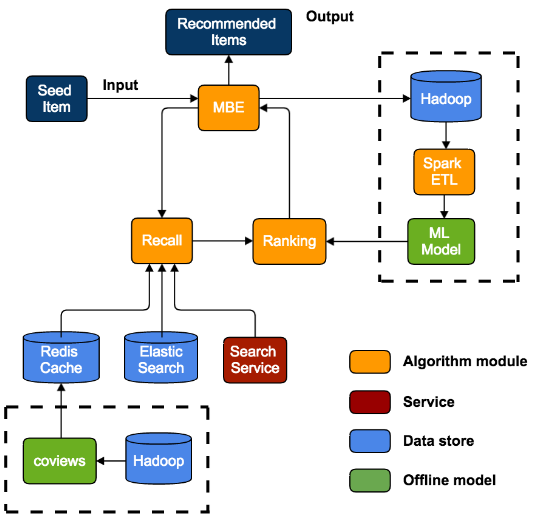


Figure 2: Architecture of the merchandising backend system. Offline components are marked by dashed lines.

product ID, we fetch other items with the same product ID from active inventory.
Coviewed items: A *coviewed* item pair is a pair of items which have been frequently viewed together in the same browsing session by multiple users. The coview behavioral model has been used across many recommender systems that use collaborative filtering. In our setting, we use a more rudimentary form of it and search for items that have been viewed together with the seed. Due to the volatile nature of the inventory, we follow a lambda architecture [6] to process session logs in realtime and batch. To ensure similarity, we also filter out coviewed items that may be from a different item category than the seed item.
Title similarity: Searching for items with a similar title yields a fairly large recall set. Title queries, however, posed an interesting challenge. Strict latency requirements on the item page prevent us from impacting the user experience, *i.e.*, there can be no noticeable delay when loading an item page with recommendations above the fold. In addition, unlike keyword searches, item titles tend to be much longer and hence, using eBay search was not a scalable option. To accommodate this type of search, we index all of eBay's items in an Elasticsearch database [3], sharded by its site (country) and category. As our problem setting includes a given seed item with a specific site and category, we can narrow our search focus to only items within the same domain. This kind of sharding allows search latencies to fit within our desired 100 ms range. We noticed that the quality of the returned recommendations was higher than our previous approach which used locality sensitive hashing, based on the item's title. Elasticsearch's indexing scheme uses TF/IDF based weightings in order to calculate relevance scores.

Coview and title-similarity based methods increase our total recall significantly and comprise up to 90% of recommendations.

2.2 Ranking

User impressions interactions (which include clicks and purchases) are logged to eBay's Hadoop cluster for analysis. Apache Spark [1] is used for Extract, Transform and Load

Table 1: Class label options for binary classifier and KL divergence for price feature.

negative class	positive class	KL divergence
non-clicked	clicked	0.01
non-purchased	purchased	0.13
clicked not purchased	purchased	0.07
non-clicked	purchased	0.15

(ETL) where the user logs are joined with item details and the data is subsampled. The trained offline MLR model parameters are passed to the runtime ranking application where predictions are made in real time.

3. RANKING MODEL

The pointwise learning to rank problem, in the context of serving item recommendations, is reduced to a binary classification problem where we rank the recommendations on the probability of being purchased (positive class), effectively maximizing conversion. The training and testing data sets are generated with features derived from {*seed item*, *recommended item*} pairs and binary class labels being {0=non-clicked, 1=purchased} from recommendations shown in the past. At runtime in production, the Recall stage produces candidates for possible {*seed item*, *recommended item*} pairs, and then in the Ranking stage the model assigns a probability score to each such pair, with the 5 highest scoring pairs resulting in recommendations to be shown to the user. The sampling strategy as well as the engineering of specific features are critical to the success of any classifier.

3.1 Sampling Strategy

We train a binary classifier on user browsing and purchase logs. Initially, we considered using "non-clicked" and "clicked" as the class labels for the model and rank the recommendations by the probability of click. To investigate the effectiveness of this strategy, we looked at the class separability of each feature. Figure 3a shows a histogram of the price feature score (described in the next section) for the non-clicked / clicked sampling strategy. User click patterns tend to be noisy as users browse items in a marketplace for a variety of reasons. This leads to the classes not separating well. We used the KL divergence to get a quantitative measure of the overlap of the probability distributions for the 2 classes. Table 1 shows several potential approaches to choose classes, as well as the resulting KL divergences for the price feature. Figure 3b shows the histogram for the non-clicked / purchased sampling strategy. We ran this type of analysis, looking at the KL divergence, for all of the features in our model to validate that the non- clicked / purchased strategy is optimal for class separation. This reflects user intention – non-clicked recommendations show absolutely no user interest while the purchased recommendations indicate complete user intention (conversion). Clicked recommendations are in the middle of this spectrum and cannot be used as effectively for classification.

Purchased recommendations occur an order of magnitude less often than clicked recommendations, which themselves occur in a relatively low portion of impressions. Due to this extreme class imbalance, we subsample the negative class to be balanced with the positive class in order to improve classifier performance.

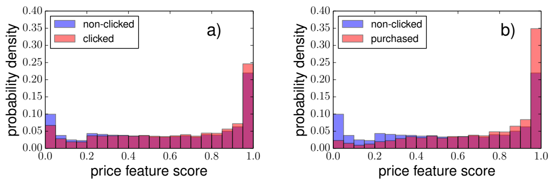


Figure 3: Histograms showing class separation of the price feature with the a) non-clicked / clicked and b) non-clicked / purchased sampling strategies.

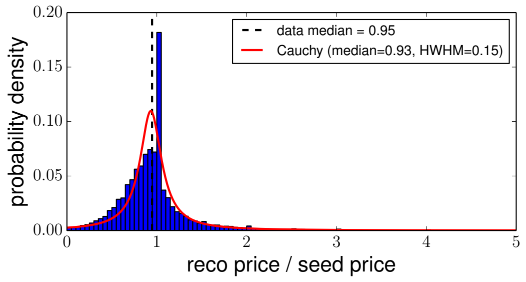


Figure 4: Histogram of price ratio and Cauchy fit.

3.2 Feature Engineering

There are two types of features in the model, *comparison* features and *item quality* features. Comparison features compare properties of the seed item to the recommendation candidate item. Some examples of this include comparing price, condition, format, and titles (using TF/IDF based methods). The item quality features are designed to ensure higher quality items are surfaced to the user. These features include popularity and seller feedback of the item seller, among others. We now discuss the details of an important *comparison* feature – the price feature.

Prices of identical items can vary by orders of magnitude because of condition or simply due to preference of the seller. Instead of directly comparing the seed item price, p_{seed}, to the recommended item price, p_{reco}, we instead model the ratio p_{reco}/p_{seed}. Analysis shows this ratio is centered near unity. Figure 4 shows a normalized histogram of the price ratio (blue bars) from past purchase events, where the user viewed the seed item and then purchased the recommended item. We modeled this data with the Cauchy distribution which has a probability density function of the following form:

$$f_{Cauchy}(x, x_0, \gamma) = \frac{1}{\pi\gamma\left[1 + \left(\frac{x - x_0}{\gamma}\right)^2\right]} \qquad (1)$$

where x_0 is the median and γ is the Half Width at Half Maximum. We looked at other distributions such as the Gaussian, Gamma, and Weibull, however the Cauchy had the best fit to the past purchase data. The price feature score is then generated from a normalized Cauchy distribution $s_{price} = f_{Cauchy}(x, x_0, \gamma) \cdot \pi\gamma$, which is 1 when $p_{reco} = p_{seed}$ and smoothly transitioning to 0 otherwise.

3.3 Baseline

The baseline ranking model that is used for comparison is a linear model with manually adjusted weights based on input from domain experts and human judgement of test output. Potential recommendations are ranked based on

Table 2: Classification and ranking metrics.

Classifier	AUC	NDCG@k=1	NDCG@k=5
baseline	0.67	0.336	0.684
Naive Bayes	0.74	0.341	0.682
Logistic Regression	0.77	0.353	0.689
Decision Tree	0.79	0.407	0.715
Random Forest	0.80	0.370	0.696
Gradient Boosting	0.81	0.370	0.697

highest global score, $G = \sum_{j=1}^{N} w_j \cdot s_j$, where w_j and s_j are the weight and feature score, respectively. We define normalizations on the weights $\sum_{j=1}^{N} w_j = 1$ and feature scores $0 \leq s_j \leq 1$ which results in a constraint on the global score $0 \leq G \leq 1$. We launched this model to live user traffic to validate effectiveness of the features.

3.4 Classification and Ranking Results

We evaluated the performance of several classifiers on the balanced dataset with the non-clicked / purchased class labels. The dataset contained 352,070 examples gathered over 10 days of user logs and was randomly split into training (60%) and validation (40%) sets. Table 2 shows the ROC AUC scores from the validation data set demonstrating improved performance over the baseline model (Section 3.3). The hyperparameters for the tree based classifiers ware optimized using grid search. The accuracy (0.70) as well as the positive class precision (0.70) and recall (0.70) for the logistic regression indicated reasonable classification performance.

To evaluate ranking performance, we took the raw unsampled recommendations (1,177,117 examples) from impressions that contained a purchase, excluding impressions used for training. To gauge the quality of the ranking, we used the normalized discounted cumulative gain truncated at rank k (NDCG@k) metric which is typically used for evaluating ranking performance [7, 4]. We defined the relevance function to be {non-clicked = 0, clicked = 0, purchased = 1}. The model was evaluated on 10 datasets with eBay user log data from different time frames and countries. Classification and ranking performance was comparable to the results shown in Table 2.

3.5 A/B Test Results

As typically done in the ad tech industry, we segment the model by product category and country. An individual classifier was trained in every major category as users tend to shop differently in different domains (for example, prices may be more important in clothing than in antiques).

We implemented a logistic regression model for our production platform due to its scalability and reasonable offline classification and ranking performance compared to the tree based classifiers. The country and category segmented MLR model was A/B tested in live production traffic with the control being the baseline model defined in Section 3.3. Table 3 shows positive lift in critical operational metrics: click through rate (CTR), purchase through rate (PTR), and revenue. The model was optimized to maximize conversion which can be seen in the increase in PTR (+6.6%) and revenue (+6.0%). While the A/B test results were correlated with the improvement in offline performance between the logistic regression classifier and the baseline model, we are investigating how well this holds for the other classifiers. This machine learned ranking model was launched to full production traffic on the eBay item page worldwide in 8 countries including US, UK, and Germany.

Table 3: Lift in A/B test operational metrics.

	CTR	PTR	Revenue
lift	+3.0%	+6.6%	+6.0%

4. CONCLUSIONS AND FUTURE WORK

In this work, we presented a highly scalable architecture which produces high quality similar item recommendations in a diverse semi-structured marketplace. We developed a widely applicable and interpretable pointwise machine learned ranking model trained on implicit eBay user shopping behavior. The model optimized recommendation rank based on probability of purchase. Although many of our marketplace characteristics are unique, the ranking model and sampling strategy is general enough for most domains which require ranking recommendations against a seed item. Our future work consists of adding new features, implementing better-performing offline classifiers for the runtime ranking model, as well as incorporating user personalization and segmentation.

5. ACKNOWLEDGMENTS

We are grateful for invaluable discussions with Tomer Meron, Benjamin Klein, Thomas Pinckney, and Uwe Mayer. We thank the reviewers for their valuable feedback.

6. ADDITIONAL AUTHORS

Paul Wang (eBay Inc., New York City, USA, email: pauwang@ebay.com)

7. REFERENCES

[1] Apache Spark, spark.apache.org/.
[2] D. Arya, V. Ha-Thuc, and S. Sinha. Personalized federated search at LinkedIn. In *CIKM*, 2015.
[3] Elasticsearch, www.elastic.co/products/elasticsearch.
[4] A. Freno et al. One-pass ranking models for low-latency product recommendations. In *SIGKDD*, 2015.
[5] D. Goldberg, D. Nichols, B. M. Oki, and D. Terry. Using collaborative filtering to weave an information tapestry. *Commun. ACM*, 35(12):61–70, Dec. 1992.
[6] Lambda Architecture, lambda-architecture.net/.
[7] P. Li, Q. Wu, and C. J. Burges. Mcrank: Learning to rank using multiple classification and gradient boosting. In *NIPS*, pages 897–904, 2007.
[8] G. Linden, B. Smith, and J. York. Amazon.com recommendations: Item-to-item collaborative filtering. *Internet Computing, IEEE*, 7(1):76–80, 2003.
[9] T.-Y. Liu. Learning to rank for information retrieval. *Foundations and Trends in Information Retrieval*, 3(3):225–331, 2009.
[10] B. Sarwar et al. Item-based collaborative filtering recommendation algorithms. In *WWW*, 2001.
[11] H. Steck. Gaussian ranking by matrix factorization. In *RecSys*, pages 115–122, 2015.
[12] J. Weston et al. Learning to rank recommendations with the k-order statistic loss. In *RecSys*, 2013.