A Content-based Recommender System for E-commerce Offers and Coupons

Yandi Xia
Rakuten Institute of Technology
Boston, Massachusetts - USA 02110
yandi.xia@rakuten.com

Giuseppe Di Fabbrizio
Rakuten Institute of Technology
Boston, Massachusetts - USA 02110
difabbrizio@gmail.com

Shikhar Vaibhav
Ebates
San Francisco, California - USA 94100
svaibhav@ebates.com

Ankur Datta
Rakuten Institute of Technology
Boston, Massachusetts - USA 02110
ankur.datta@rakuten.com

ABSTRACT

Reward-based e-commerce companies expose thousands of online offers and coupons every day. Customers who signed up for online coupon services either receive a daily digest email with selected offers or select specific offers on the company website front-page. High-quality online discounts are selected and delivered through these two means by applying a manual process that involves a team of experts who are responsible for evaluating recency, product popularity, retailer trends, and other business-related criteria. Such a process is costly, time-consuming, and not customized on users' preferences or shopping history. In this work, we propose a content-based recommender system that streamlines the coupon selection process and personalizes the recommendation to improve the click-through rate and, ultimately, the conversion rates. When compared to the popularity-based baseline, our content-based recommender system improves F-measures from 0.21 to 0.85 and increases the estimated click-through rate from 1.20% to 7.80%. The experimental system is currently scheduled for A/B testing with real customers.

CCS CONCEPTS

• **Computing methodologies** → *Classification and regression trees*;
• **Applied computing** → **Online shopping**;

KEYWORDS

Recommender Systems; Personalization; E-commerce offers and coupons

ACM Reference format:
Yandi Xia, Giuseppe Di Fabbrizio, Shikhar Vaibhav, and Ankur Datta. 2017. A Content-based Recommender System for E-commerce Offers and Coupons. In *Proceedings of SIGIR eCom 2017, Tokyo, Japan, August 2017,* 7 pages.

In: J. Degenhardt, S. Kallumadi, M. de Rijke, L. Si, A. Trotman, Y. Xu (eds.): Proceedings of *the SIGIR 2017 eCom workshop, August 2017, Tokyo, Japan*, published at http://ceur-ws.org

1 INTRODUCTION

Reward-based e-commerce services are a fast-growing online sector that provides cashback to subscribers by leveraging discounted prices from a broad network of affiliated companies.

Typically, a consumer would use the company website to search for specific products, retailers or potentially click on one of the prominent published offers. Shoppers are then redirected to the websites of the respective retailers. A visit to the merchant's website generated by this redirection is defined as *shopping trip*.

At the same time, subscribers receive *daily digest email* with the most popular discounts. Figure 1 shows a fragment of a daily digest email featuring a list of current offers and coupons. Both website and email contents are manually curated by experts focusing on delivering the highest quality offers to customers.

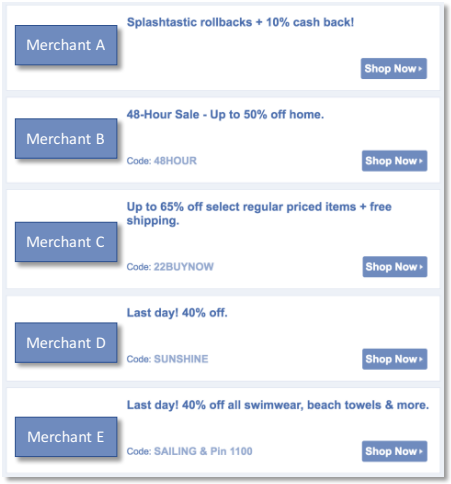

Figure 1: Example of email campaign message including offers and coupons.

This manual process is laborious and does not scale well to millions of online customers who are receiving the same offer recommendations regardless of their previous browsing or purchase history. A recommender system would be able to capture the customers' preferences by optimizing the coupon selection based on the previous history and the similarity across users and / or items.

However, compared to traditional recommender systems in other domains, such as movies [6, 10] and music [16], online offers differ for a number of reasons: 1) Coupons are *highly volatile* items only valid in a limited time span and removed after their expiration date; 2) Customers using coupons are affected by *high turnover*; 3) Users' population is *skewed* between a minority of heavy users and a majority of occasional or light users; and finally, 4) Online offers are *rarely rated* like usually happens for movie and music domains.

Consequently, the online discount domain is more prone to the *cold-start problem* [4], where there is insufficient click history for a new item or, conversely, there is not enough history for a new subscriber. Matrix factorization methods are less robust to the cold-start problem [4] and lack the capability to capture the number of features necessary to model preferences of high volatile items and large customer populations.

For these reasons, we adopted a *content-based filtering* (CBF) approach [17] where we exploit coupons and customers' attributes to build a stochastic classification model that predicts the posterior probability for a coupon to be clicked by the customer. We rely on users' *implicit feedback* which indirectly express users' preferences through behavior like clicking on specific offers in a ranked list. This is typically a more noisy signal compared to *explicit feedback* such as ratings and reviews, but it is also abundant and with consistent bias across user's click-through history [15, 23].

However, click information is more challenging to use since it does not directly reflect the users' satisfaction neither provides clues about items that are irrelevant (*negative feedback*) [18] which is fundamental to build an effective discriminative model. In this paper, we illustrate how to utilize noisy implicit feedback to build a CFB model for offers and coupons recommendations. The main contributions are the following:

(1) We describe a method to generate negative samples from the implicit feedback data;
(2) We perform extensive experiments to compare different learning models;
(3) We demonstrate that our approach is effective with respect to the baseline and the upper-bound baseline.

2 MODELING APPROACHES

We formalize the coupon recommendation problem as a *content-based filtering* [17] with implicit feedback. Let M and N indicate the number of users and coupons, respectively. For each shopping trip interaction included in the matrix $\mathbf{Y} \in \mathbb{R}^{M \times N}$, a user u_i is exposed to a number of coupons L and selects a specific coupon c_j generating the implicit feedback $y_{u_i,c_j} = 1$, and $y_{u_i,c_k} = 0$ for all the other items in the list of length L, where $k = \{0, 1, \ldots, L-1\}$ and $k \neq j$. Formally, predicting a user's shopping trip for an online offer requires to estimate the function $\hat{y}_{u_i,c_j} = f(u_i, c_j|\Theta)$, where $\hat{y}_{u_i,c_j}$ is the predicted score for the online interaction (u_i, c_j); Θ represents the model parameters to learn, and f is the mapping function between the parameters and the predicted scores. The learning can be done by applying a machine learning classification algorithm that optimizes an object function such as logistic loss or cross-entropy loss [12].

However, taking a closer look at the implicit feedback used for modeling, $y_{u_i,c_k} = 0$ does not necessarily mean that the user u_i dislikes the offers y_{u_i,c_k}, but merely indicates the choice preference in this round of interactions. The user may go back to the offer list and select another item which is different from c_j. This makes the prediction modeling challenging since user's implicit feedback is noisy and there is a lack of negative feedback. In our case, we only considered the positive feedback and modeled the shopping trip predictions as *one-class classification* (OCC) problem [21], where we only use the positive samples and artificially derive the negative samples from the user's shopping history.

To handle the classification task, we experimented with two non-parametric learning methods that are resilient to noisy features, robust to outliers, and capable of handling missing features (but not necessarily robust to noisy labels): 1) Random forests [5] [12, Chapter 15]; and 2) Gradient boosted trees [9].

2.1 Generating negative samples

When relying on the user's implicit feedback, an accepted convention is that any higher ranked item that was not clicked on is likely less relevant than the clicked ones [20]. In an industrial setting, it is often the case that neither the coupon rank or the coupon list exposed to users are available. In this case, it is necessary to derive *"negative shopping trips"* (i.e., not relevant coupons) by some other means. For the negative training and testing sets, we followed the procedure described below:

For each shopping trip, we extracted the triple: user, coupon, and click time;

- We retrieved the user's shopping trip history and select all the coupons clicked by them up to the current click time. We call this data set A;
- Given the considered shopping trip click time, we retrieved all the non-expired coupons at that time. We call this data set S;
- We removed all the clicked coupons by the user from the non-expired coupons set and obtained the data set $U = S - A$;
- We uniformly sampled 2, 4, 8, and 9 coupons from U, and build from each of the sample set a list of negative shopping trips that would simulate the missing negative sample in our data set.

Note that we sampled different sizes of negative sets for evaluation purposes, but we only reported the results for negative sets of size 9 since it is the closest to the real system conditions where the user is exposed to a list of 10 offers and selects one of them.

2.2 Random forests

Random forest (RF) models [5][12, Chapter 15] use bagging techniques to overcome decision trees's low-bias and high variance problem. A RF model averages the results of several noisy and relatively unbiased decision trees trained on different parts of the same training set. The general method to create a RF model follows the steps below [12, Chapter 15]:

For $b = 1, \ldots, B$, where B is the total number of bagged tree:

(1) Draw N samples from the training set and build a tree T_b by recursively apply the steps below for each terminal node until reaching the minimum node size n_{min}:

(a) Randomly select m features from the total p features, where $m \leq p$;
(b) Pick the best node split among the m features;
(c) Split the selected node into two children nodes;

(2) Save the obtained ensemble tree $\{T_b\}_1^B$

To determine the quality of the node split, either Gini impurity or cross-entropy measures can be used for the candidate split [12, Chapter 9]. For a classification task, the prediction is the majority vote among the results from the tree ensemble. In other terms, if $\hat{C}_b(x)$ is the class prediction for the b_{th} random-forest (rf) tree, then the predicted class is: $\hat{C}_{rf}^B(x) = majority\ vote\{\hat{C}_b(x)\}_1^B$.

2.3 Gradient boosted trees

Gradient boosted trees (GBTs) [9] optimize a loss functional: $\mathcal{L} = E_y[L(y, F(\mathbf{x})|\mathbf{X})]$ where $F(\mathbf{x})$ can be a mathematically difficult to characterize function, such as a decision tree $f(\mathbf{x})$ over $\mathbf{X}$. The optimal value of the function is expressed as $F^\star(\mathbf{x}) = \sum_{m=0}^{M} f_m(\mathbf{x}, \mathbf{a}, \mathbf{w})$, where $f_0(\mathbf{x}, \mathbf{a}, \mathbf{w})$ is the initial guess and $\{f_m(\mathbf{x}, \mathbf{a}, \mathbf{w})\}_{m=1}^{M}$ are *additive boosts* on $\mathbf{x}$ defined by the optimization method. The parameter $\mathbf{a}_m$ of $f_m(\mathbf{x}, \mathbf{a}, \mathbf{w})$ denotes split points of predictor variables and $\mathbf{w}_m$ denotes the boosting weights on the leaf nodes of the decision trees corresponding to the partitioned training set $\mathbf{X}_j$ for region j. To compute $F^\star(\mathbf{x})$, we need to calculate, for each boosting round m,

$$\{\mathbf{a}_m, \mathbf{w}_m\} = \arg\min_{\mathbf{a},\mathbf{w}} \sum_{i=1}^{N} L(y_i, F_m(\mathbf{x}_i)) \tag{1}$$

with $F_m(\mathbf{x}) = F_{m-1}(\mathbf{x}) + f_m(\mathbf{x}, \mathbf{a}_m, \mathbf{w}_m)$. This expression is indicative of a gradient descent step:

$$F_m(\mathbf{x}) = F_{m-1}(\mathbf{x}) + \rho_m\left(-g_m(\mathbf{x}_i)\right) \tag{2}$$

where ρ_m is the step length and $\left[\frac{\partial L(y,F(\mathbf{x}))}{\partial F(\mathbf{x})}\right]_{F(\mathbf{x}_i)=F_{m-1}(\mathbf{x}_i)} = g_m(\mathbf{x}_i)$ being the search direction. To solve $\mathbf{a}_m$ and $\mathbf{w}_m$, we make the basis functions $f_m(\mathbf{x}_i; \mathbf{a}, \mathbf{w})$ correlate most to $-g_m(\mathbf{x}_i)$, where the gradients are defined over the *training data* distribution. In particular, using Taylor series expansion, we can get closed form solutions for $\mathbf{a}_m$ and $\mathbf{w}_m$ – see [7] for details. It can be shown that

$$\mathbf{a}_m = \arg\min_{\mathbf{a}} \sum_{i=1}^{N} \left(-g_m(\mathbf{x}_i) - \rho_m f_m(\mathbf{x}_i, \mathbf{a}, \mathbf{w}_m)\right)^2 \tag{3}$$

and

$$\rho_m = \arg\min_{\rho} \sum_{i=1}^{N} L(y_i, F_{m-1}(\mathbf{x}_i) + \rho f_m(\mathbf{x}_i; \mathbf{a}_m, \mathbf{w}_m)) \tag{4}$$

which yields,

$$F_m(\mathbf{x}) = F_{m-1}(\mathbf{x}) + \rho_m f_m(\mathbf{x}, \mathbf{a}_m, \mathbf{w}_m) \tag{5}$$

Each boosting round updates the weights $w_{m,j}$ on the leaves and helps create a new tree in the next iteration. The optimal selection of decision tree parameters is based on optimizing the $f_m(\mathbf{x}, \mathbf{a}, \mathbf{w})$ using a logistic loss. For GBTs, each decision tree is resistant to imbalance and outliers [12], and $F(\mathbf{x})$ can approximate arbitrarily complex decision boundaries.

2.4 Popularity-based baseline

To evaluate the contributions of our coupon recommender system, it is necessary to establish a baseline measure that correlates with the current recommender system performance. However, the current coupon recommendation process cannot be evaluated offline since recommendations are done by manually selecting popular retailers and offers. The resulting list of coupons is then exposed to all the users by either daily email or website front-page visits. All users see the same list of coupons regardless of their shopping trip history or preferences. A direct comparison with such a system would require exposing both recommendation techniques through A/B testing with real customers.

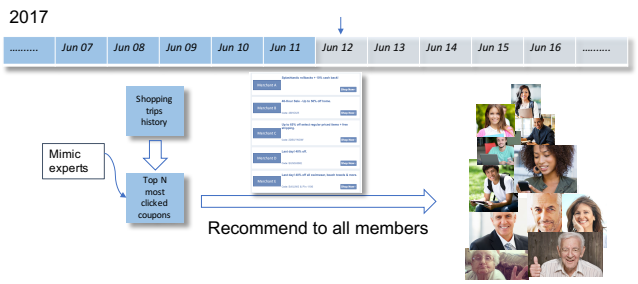

Figure 2: Popularity-based baseline (PBB) coupon selection obtained by considering all the valid coupons prior the shopping trip date.

As an alternative approach, we propose an offline baseline evaluation based on popular coupons, where the popularity is derived from shopping trips click-through rates. In addition to popularity criteria, there could be other business reasons for offer selection in the current setting, but for baseline and simulation purposes we have limited the selection to the popularity criteria only.

The assumption is that user-defined popularity correlates with the current recommender system and it is likely to represent an upper-bound estimation of the performance. At a high-level, we give two baselines which select the most popular coupons based only on the past click data (PBB) or only based on the future click data (God-View PBB), which represents a strong upper bound.

As illustrated in Figure 2, we first divide users' shopping trips into training and testing partitions with a 90% and 10% ratio, respectively. For each shopping trip in the shopping trip test set, we performed the following steps:

- We extracted the shopping trip click date;
- From the shopping trip train set, we extracted all the shopping trips prior the click date with a valid coupon (i.e., unexpired coupons at the time of the click date);
- We sorted the obtained list of coupons by click-through rate and selected the top N popular coupons;
- If the coupon in the test shopping trip is in the top N most popular coupons (with $N = 10$), then there would be a match and the prediction is correct; otherwise the prediction is wrong.

The general idea is that the customized popularity list simulates the expert selection by identifying the best offer for each of the considered day in the test set. We also define a stronger baseline where we have access to future click-through information after the shopping trip under evaluation. By predicting the future, we can build a baseline which is the upper-bound of the popularity-based baselines when recommending the same list of offers to the whole user population. We call this baseline *God-View PBB* (GVPBB).

2.5 Run-time

Figure 3 shows the simplified architecture of the run-time system. A daily list of thousands of candidate coupons is submitted for selection to the recommender system. For each member subscribing to the reward-based e-commerce company, each pair of member / offer is evaluated by the recommender system and scored for affinity. The resulting score list is used to rank the predictions which are then truncated to the top 10 candidates and used either as personalized email campaigns or personalized front page offers.

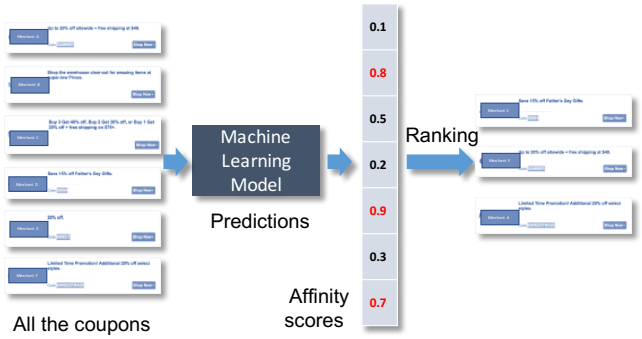

Figure 3: Run-time coupons selection and ranking.

3 DATA

We used data collected from customers directly clicking on the published offers, *e-shopping* at the merchants' website and consequently adding offers, or auto-applying offers through browser's button extensions.

The data includes historical information about customers' click-throughs (i.e., *shopping trips*) to visit affiliated retailer websites. The data was sampled across a full year of traffic and anonymized for privacy requirements. Table 1 summarizes the distributions of the various data sets including active members generating shopping trips, shopping trip counts, clicked coupons, and retailers that issued the offer.

Coupons are characterized by a textual descriptions (e.g., *"25% off Tees Shorts Swim for Women Men Boys and Girls Extra 30% off Sale Styles and Extra 50% off Final Sale Styles"*.) and a date range defining their validity (e.g., from *12/18/2016* to *12/27/2016*). Optionally, coupons are classified by three categories: 1) percentage discount; 2) money discount; and / or 3) free shipping offer for specific brands or products. Additionally, they can directly refer to a specific product or retailer. Retailers' information includes a business name (e.g., *Stila Cosmetics*) and an optional category (e.g., *Health & Beauty*)

Table 1: Shopping trips data distribution across different data sets.

Data set	Members	Shopping trips	Coupons	Retailers
Train	448,055	1,037,030	15,276	1,528
Test	49,789	114,300	7594	1,192
Train Negative	358,444	11,666,592	9,705	1,453
Test Negative	39,831	1,285,875	8,190	1,413

Table 2: Data counts and stats for head, torso, and tail sets of the shopping trips distribution.

Data set	Members	Shopping trips	%	Min	Max	Avg
Head	25,149	317,623	28.7%	8	1,569	15.79
Torso	132,337	441,533	39.8%	3	7	4.17
Tail	333,686	349,262	31.5%	1	2	1.31

The details about the negative data sets in Table 1 are explained in the previous Section 2.1.

The number of shopping trips per customers follow the typical power law probability distribution [1] where the *heavy-tailed* data still holds a substantial amount of probability information. Figure 4 shows the probability density function (PDF) $p(X)$ in blue, the complementary cumulative density function (CCDF) $p(X \geq x)$ in red, and the corresponding power law distribution fit with parameters $x_{min} = 1$ and $\alpha = 3.27$.

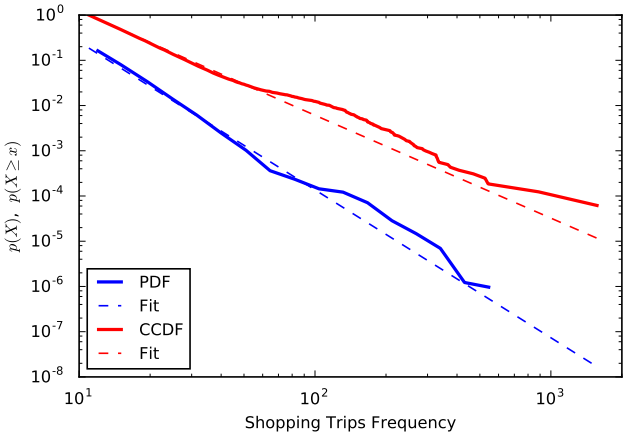

Figure 4: Probability density function ($p(X)$, blue) and complementary cumulative density function ($p(X \geq x)$, red) with power law distribution fit for $x_{min} = 1$ and $\alpha = 3.27$

Based on the shopping trips frequency (see Table 2), we split the data in three segments: 1) The *Head* comprising the most active members with more than 8 shopping trips; 2) the *Torso* including the members with a click rate between 2 and 7; and 3) the *Tail* for the less active members with 1 or 2 clicks.

This data partition will be used in the Section 4.2 to evaluate the user-based cold-start problem.

4 EXPERIMENTS

To build and evaluate random forests, XGBoost, and the baseline models, we partitioned the data by members using 90% of the population for training and 10% for testing and derived the associated shopping trips sets. Partitioning by members rather than by shopping trips would increase the chances to capture the model generalization capabilities, since the shopping trips history of unseen test users will not be biasing the training data. For the negative samples, as explained in Section 2.1, we select 9 negative shopping trips for each positive sample.

To capture the users / coupons affinity, we experimented with a number of features involving specific users' and coupons' attributes, as well as shopping history features. The following list summarize them:

- User-based features
  - Devices used
  - OS systems
  - Email domain
- Coupon-based features
  - Text description
  - Originating retailer
  - Retailer's category
  - Discount info (free shipping, dollar off, percent off)
  - Cashback info
- Shopping trip history-based features
  - Times the user has visited this coupon before
  - Recent visited retailers
  - Recent discount information
  - Recent cash back information
  - Recent coupon keywords (from descriptions)
  - Recent retailer's categories
  - Recent luxury purchases

In the data pre-processing stage, we encoded text as bag of word frequencies and members' shopping behavior histories as bag of frequencies, and categorical features as one-hot vector representation. All the other features are either binary or numeric values. Since both RF and XGBoost models are insensitive to monotonic feature transformations, we avoided numeric standardization and normalization. We carefully handled the normalization of missing information and replaced missing text with the label unk and both categorical and numerical features with 0s. After pre-processing, the resulting number of features is 45,329. For the XGBoost model, we ran a partial grid-search optimization to set the most important hyper-parameters, the maximum tree depth and the number of estimators, to 15 and 1,000, respectively. We used the scikit-learn [19] implementation of Random Forest Tree and XGBoost.

4.1 Results

Table 3 shows the experimental results for the machine learning models and the baselines. We evaluated performance by truncating the ranked prediction list at 10 items. RFC model could predict click-through for 96.80% of the tested coupons (accuracy), while the XGBoost model followed with 95.85% correct click predictions, but with a substantial higher recall of 0.83. This pushed the XGBoost predicted click-through rate to 8.30%. Both PBB and GVPBB baselines suffer of low recall since the popularity lists, once selected the shopping trip day, are the same for all the users.

Table 3: Precision (P), recall (R), F-measure (F) and Click-through rate (CTR) for random forest classification (RFC), XGBoost, popularity-based baseline (PBB), and God-View PBB (GVPBB) with 10 maximum coupon recommendations. Baseline methods are marked with (*).

Model	P	R	F	NDCG@10	CTR
PBB*	0.91	0.12	0.21	0.776	1.20%
GVPBB*	0.83	0.34	0.48	0.892	3.40%
XGBoost	0.77	**0.83**	0.80	0.857	**8.30%**
RFC	**0.93**	0.78	**0.85**	**0.969**	7.80%

Overall, both RFC and XGBoost outperform the baselines metrics. In addition to precision, recall, and F-measure, we also report the Normalized Discounted Cumulative Gain (NDCG) metric [14] for a list of 10 offers generated by combining 1 positive sample with 9 randomly selected negative samples. NDCG is a ranking quality measure that keeps into consideration the position of the clicked item. Generally, positive samples should be ranked higher than the negative ones.

As further validation, we evaluated the quality of the confidence scores estimated by the RFC model. Well-calibrated predicted probabilities are fundamental when the classification model scores are used for ranking [18].

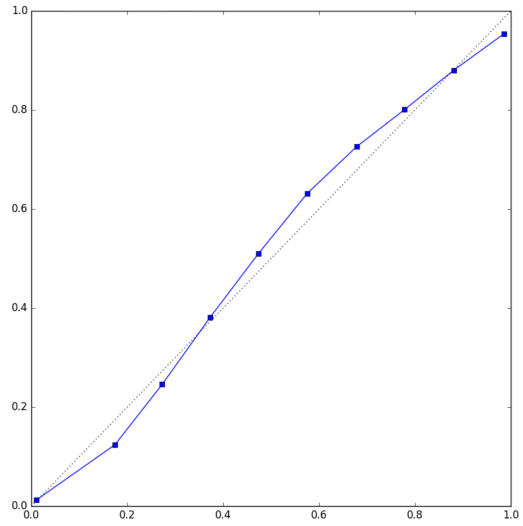

Figure 5: Probability calibration curve for the RFC model. X-axis: mean predicted value; Y-axis: fraction of positive samples.

The confidence calibration plot in Figure 5 shows that the RFC model has a minor bias around a predicted value of 0.2 and 0.6, but

overall the curve follows the ideal behavior by closely mimicking the diagonal.

4.2 User-based cold-start evaluation

When new members join a reward-based system or new offers are issued by retailers, recommender systems are affect by the sparseness of the new data that compromise prediction performance [19]. To mitigate this problem, a different selection strategy is used for the tail of the distribution (i.e., hybrid methods) [19]. However, we rely on the user's and coupon's based features to capture information that may generalize well with minimum shopping trip history.

We evaluated our RFC model with the three segments of the shopping trip distribution (see also Section 3). Table 4 shows the results of such a test.

Table 4: Precision (P), recall (R), F-measure (F), Click-through rate (CTR), and NDCG@10 for the RFC model when testing users in the head, torso, and tail sections of the shopping trip distribution.

Data set	P	R	F	NDCG@10	CTR
Head	0.95	0.77	0.85	0.980	7.70%
Torso	0.94	0.79	0.86	0.985	7.90%
Tail	0.90	0.78	0.84	0.976	7.80%

On average, the RCF model is quite robust across the three distribution segments. When analyzing the main feature predictors, the number of times this user clicked on the coupons before and being a referred member are the major contributions to the predictions, but further investigation is necessary to better understand the feature interaction.

5 RELATED WORK

A large body of work on recommender systems focuses on explicit feedback where users express their preference with ratings [10, 22]. More recently, there is an increasing number of contributions that rely on the more abundant and noisy implicit feedback data [3, 11, 13]. However, most of this work frames the problem as a collaborative filtering (CF) model. CF is notoriously sensitive to the cold-start problem and not flexible to handle extremely volatile items such as online offers and coupons. Additionally, it is hard to incorporate features that capture users' and items' properties.

The hybrid recommender system described in [11] is the closest to our work. To address the cold-start problem, they use implicit feedback to recommend online coupons by integrating both user's information and items features. In this work, the traditional collaborative filtering matrix, including users and items, is enriched with additional user and item features and factorized with SVD. Nevertheless, the hybrid system requires to be retrained every time there is a new coupon or user. In our case, there are thousands of new coupons every day and re-training that often is impractical.

Our work is in line with content-based filtering (CBF) approaches such as [2, 8]. CBF methods are able to abstract items and represent user / item interactions by encoding shopping trip history. There is no need to re-train when adding new offers, although including new retailers requires updating the model to capture the new user / item affinity relations expressed in the new shopping trips. In contrast to SVD and CF-based models, the content-based approach can leverage different classification techniques and can be combined with CF.

6 CONCLUSIONS

In this work, we developed a coupon recommender system with content-based filtering. We experimented with two popularity-based baselines and compared them to both a random forest and gradient boosted tree classification models. To address the one-class classification problem, our framework automatically extracted negative samples from customers' shopping trips. Negative samples simulate the real selection scenario where one coupon, over the visualized ten coupons, is actually selected by members. In this experimental setup, both random forests and XGBoost classifiers perform substantially better than the two baselines except for the NDCG10 metric.

In the future, to improve ranking results, we will extend the classification models with pair-based and list-based cost functions. We will also validate our models under A/B testing evaluations with real customers.

ACKNOWLEDGMENTS

The authors would like to thank Dr. David Inouye for providing insightful suggestions and comments.

The authors would also like to thank the anonymous reviewers for their helpful advice and Yiu-Chang Lin for presenting our work at the workshop.

REFERENCES

[1] Jeff Alstott, Ed Bullmore, and Dietmar Plenz. 2014. powerlaw: A Python Package for Analysis of Heavy-Tailed Distributions. *PLOS ONE* 9, 1 (01 2014), 1–11.

[2] Ana Belén Barragáns-Martínez, Enrique Costa-Montenegro, Juan C. Burguillo, Marta Rey-López, Fernando A. Mikic-Fonte, and Ana Peleteiro. 2010. A Hybrid Content-based and Item-based Collaborative Filtering Approach to Recommend TV Programs Enhanced with Singular Value Decomposition. *Inf. Sci.* 180, 22 (Nov. 2010), 4290–4311.

[3] Immanuel Bayer, Xiangnan He, Bhargav Kanagal, and Steffen Rendle. 2017. A Generic Coordinate Descent Framework for Learning from Implicit Feedback. In *Proceedings of the 26th International Conference on World Wide Web (WWW '17)*. International World Wide Web Conferences Steering Committee, Republic and Canton of Geneva, Switzerland, 1341–1350.

[4] J. Bobadilla, F. Ortega, A. Hernando, and A. GutiéRrez. 2013. Recommender Systems Survey. *Know.-Based Syst.* 46 (July 2013), 109–132.

[5] Leo Breiman. 2001. Random Forests. *Mach. Learn.* 45, 1 (Oct. 2001), 5–32.

[6] Walter Carrer-Neto, María Luisa Hernández-Alcaraz, Rafael Valencia-García, and Francisco García-Sánchez. 2012. Social Knowledge-based Recommender System. Application to the Movies Domain. *Expert Syst. Appl.* 39, 12 (Sept. 2012), 10990–11000.

[7] Tianqi Chen and Carlos Guestrin. 2016. XGBoost: A Scalable Tree Boosting System. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Francisco, CA, USA, August 13-17, 2016*. 785–794.

[8] Keunho Choi, Donghee Yoo, Gunwoo Kim, and Yongmoo Suh. 2012. A Hybrid Online-product Recommendation System: Combining Implicit Rating-based Collaborative Filtering and Sequential Pattern Analysis. *Electron. Commer. Rec. Appl.* 11, 4 (July 2012), 309–317.

[9] Jerome H. Friedman. 2000. Greedy Function Approximation: A Gradient Boosting Machine. *Annals of Statistics* 29 (2000), 1189–1232.

[10] Carlos A. Gomez-Uribe and Neil Hunt. 2015. The Netflix Recommender System: Algorithms, Business Value, and Innovation. *ACM Trans. Manage. Inf. Syst.* 6, 4, Article 13 (Dec. 2015), 19 pages.

[11] J. Grivolla, D. Campo, M. Sonsona, J. M. Pulido, and T. Badia. 2014. A Hybrid Recommender Combining User, Item and Interaction Data. In *2014 International Conference on Computational Science and Computational Intelligence*, Vol. 1. 297–301.

[12] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. 2009. *The elements of statistical learning: data mining, inference and prediction* (2 ed.). Springer.

[13] Xiangnan He, Hanwang Zhang, Min-Yen Kan, and Tat-Seng Chua. 2016. Fast Matrix Factorization for Online Recommendation with Implicit Feedback. In *Proceedings of the 39th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '16)*. ACM, New York, NY, USA, 549–558.

[14] Kalervo Järvelin and Jaana Kekäläinen. 2002. Cumulated Gain-based Evaluation of IR Techniques. *ACM Trans. Inf. Syst.* 20, 4 (Oct. 2002), 422–446.

[15] Thorsten Joachims, Adith Swaminathan, and Tobias Schnabel. 2017. Unbiased Learning-to-Rank with Biased Feedback. In *Proceedings of the Tenth ACM International Conference on Web Search and Data Mining (WSDM '17)*. ACM, New York, NY, USA, 781–789.

[16] Seok Kee Lee, Yoon Ho Cho, and Soung Hie Kim. 2010. Collaborative Filtering with Ordinal Scale-based Implicit Ratings for Mobile Music Recommendations. *Inf. Sci.* 180, 11 (June 2010), 2142–2155.

[17] Pasquale Lops, Marco de Gemmis, and Giovanni Semeraro. 2011. *Content-based Recommender Systems: State of the Art and Trends*. Springer US, Boston, MA, 73–105.

[18] Alexandru Niculescu-Mizil and Rich Caruana. 2005. Predicting Good Probabilities with Supervised Learning. In *Proceedings of the 22Nd International Conference on Machine Learning (ICML '05)*. ACM, New York, NY, USA, 625–632.

[19] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, Jake Vanderplas, Alexandre Passos, David Cournapeau, Matthieu Brucher, Matthieu Perrot, and Édouard Duchesnay. 2011. Scikit-learn: Machine Learning in Python. *J. Mach. Learn. Res.* 12 (Nov. 2011), 2825–2830.

[20] Filip Radlinski and Thorsten Joachims. 2005. Query chains: learning to rank from implicit feedback. In *KDD '05: Proceeding of the eleventh ACM SIGKDD international conference on Knowledge discovery in data mining*. ACM Press, New York, NY, USA, 239–248.

[21] Oxana Ye. Rodionova, Paolo Oliveri, and Alexey L. Pomerantsev. 2016. Rigorous and compliant approaches to one-class classification. *Chemometrics and Intelligent Laboratory Systems* 159, Complete (2016), 89–96.

[22] Ruslan Salakhutdinov, Andriy Mnih, and Geoffrey Hinton. 2007. Restricted Boltzmann Machines for Collaborative Filtering. In *Proceedings of the 24th International Conference on Machine Learning (ICML '07)*. ACM, New York, NY, USA, 791–798.

[23] Chao Wang, Yiqun Liu, and Shaoping Ma. 2016. Building a click model: From idea to practice. *CAAI Transactions on Intelligence Technology* 1, 4 (2016), 313 – 322.