Comparative Evaluation of Top-*N* Recommenders in e-Commerce: an Industrial Perspective

Dimitris Paraschakis
Dept. of Computer Science
Malmö University
SE-205 06 Malmö, Sweden
email: dimitris.paraschakis@mah.se

Bengt J. Nilsson
Dept. of Computer Science
Malmö University
SE-205 06 Malmö, Sweden
email: bengt.nilsson.TS@mah.se

John Holländer*
Framtiden Consulting AB
Drottninggatan 1D
SE-212 11 Malmö, Sweden
email: johnchollander@gmail.com

***Abstract*—We experiment on two real e-commerce datasets and survey more than 30 popular e-commerce platforms to reveal what methods work best for product recommendations in industrial settings. Despite recent academic advances in the field, we observe that simple methods such as best-seller lists dominate deployed recommendation engines in e-commerce. We find our empirical findings to be well-aligned with those of the survey, where in both cases simple personalized recommenders achieve higher ranking than more advanced techniques. We also compare the traditional *random* evaluation protocol to our proposed *chronological* sampling method, which can be used for determining the optimal time-span of the training history for optimizing the performance of algorithms. This performance is also affected by a proper hyperparameter tuning, for which we propose *golden section search* as a fast alternative to other optimization techniques.**

I. Introduction

An intelligent recommender system (RS) is a powerful selling tool of any contemporary e-commerce platform. According to Forrester[1], product recommendations account for up to 30% of e-commerce sites' revenues. That was already apparent in 2006, when Amazon[2] claimed that 35% of its sales were triggered by recommendations [1]. The same year, Netflix[3] announced its famous million-dollar prize for improving its recommendation engine. This contest has spurred active research within the field, yielding a variety of recommendation algorithms over the past decade. Interestingly, the prize-winning algorithm was never put to real use: Netflix concluded that the measured accuracy gains "did not seem to justify the engineering effort needed to bring them into a production environment" [2]. This fact motivated us to explore how receptive *e-commerce* platforms are to academic advances in the field of recommender systems. The connection between research contributions and industrial solutions has rarely been in focus of RS literature. As Pradel *et al.* [3] noted, "case-studies are necessary to better understand the specificities of purchase datasets and the factors that impact recommender systems for retailers". Our work is such a case study.

Originating from the Netflix problem, the vast majority of proposed recommenders have been designed for rating prediction. In e-commerce, however, rating products is not commonly practised even when this functionality is supported by a shopping platform. The problem thus boils down to predicting a purchase rather than a rating score. Purchase prediction based solely on transactional history is the common approach known as *collaborative filtering* (CF). In the absence of explicit ratings, transactional datasets become binary and are referred to as *implicit feedback* datasets. When it comes to missing data, implicit feedback is ambiguous: if a customer has never bought a particular product, it is not always a sign of negative preference (e.g. the product might be out of stock). That is why algorithms designed for implicit feedback often model heuristics of relationships between observed and unobserved events in a dataset, as we shall see in Section IV-A. This direction is less researched than movie ratings, partly due to the lack of publicly available e-commerce datasets. In our experience, retailers are reluctant to release their sensitive purchase data. Another factor that differentiates these datasets from movie ratings is the extreme sparsity of data: a typical retail store only sells a fraction of its product catalogue.

Evaluation of recommender systems in industrial context deserves special attention. Whereas the main academic interest appears to be the accuracy of predictions, industrial recommenders are meant to be optimized for revenues. As Aiolli [4] pointed out, the choice of an accurate predictor is not all that is required to build a good recommender. Other evaluation criteria that play an important role in e-commerce are coverage of product catalogue, utility and serendipity of recommendations, adaptivity and scalability of an algorithm, etc. [5]. Since our work aims at a comparison between algorithms and not a rigorous testing of a particular recommender, we only consider accuracy and speed as our evaluation criteria. The most reliable way of testing a deployed RS is to perform an online evaluation (e.g. AB testing), which may however be too costly, time-consuming and therefore often infeasible in practice. That is why customer behaviour is usually simulated offline from the recorded purchase data. The choice of a proper user modelling strategy is a crucial factor, which must provide a realistic estimate of the real performance of an algorithm. Numerous approaches for such modelling have been discussed in [5]. In this paper we propose one such technique for offline evaluation.

The contributions of this work are the following:

1) We conduct a survey of deployed recommendation engines in the e-commerce domain. This is done to measure the extent to which recent academic

*This research was conducted while the author was a student at the master's programme in media software design at Malmö University

[1]http://www.forrester.com

[2]http://www.amazon.com

[3]http://www.netflix.com

achievements in the field of recommender systems are adopted by real-world applications. To the best of our knowledge, this is the first study of this kind.

2) We find that the performance of simpler but faster algorithms such as association rules can often be superior to more elaborate techniques such as matrix factorization (MF). This confirms the results of other recent works on purchase data, such as [3].
3) Our experiments show that training on all available data is inferior to training on recent purchases for most algorithms. To optimize recommendation accuracy, we propose the expanding time window evaluation method for finding the optimal training time-span.
4) We adapt the Golden Section Search (GSS) algorithm for multi-dimensional hyperparameter optimization.

The paper is organized as follows. Section III presents our methodology, where we describe our datasets, evaluation protocol and metrics. We then present the GSS algorithm and the survey. In Section IV, we give a brief presentation of the recommendation techniques used in our study. In Section V, we present and analyze our experimental results and survey responses. Finally, Section VI concludes the paper.

II. Related work

A few case studies, closely related to ours, performed comparative evaluations of different families of recommendation algorithms on purchase datasets. A study by Huang *et al.* [6] points to the need for better understanding of relative strengths and weaknesses of different types of algorithms in e-commerce applications. Thus, they performed a comparison of six different algorithms and found that *spreading activation* and *link analysis* produced the best results. Just as in our study, they experimented on a fashion and a book store. A more recent case study by Pradel *et al.* [3] experimentally evaluated various CF algorithms on a dataset of a French home improvement and building supplies chain. The simple *bigram rules* recommender was found to yield the best overall accuracy. One important conclusion of this work is that the relative performances of algorithms depend on the *setting* to which they are applied.

In relation to this, both studies evaluate recommenders in two settings: complete and reduced transaction history. In [6], the reduced training set contains a fraction of randomly sampled events, whereas in [3] it contains the event history of the last two weeks of the training data. In our paper, we adopt a more flexible setting where we evaluate algorithms in various history lengths (see Section III-B).

III. Methodology

In our study we conduct offline experiments on two real e-commerce datasets: a fashion store (denoted as D1) and a book store (denoted as D2). All experimentation is done on an Intel Xeon 2.53 Ghz PC with 12 Gb RAM. Our offline experiments aim to compare the performance of algorithms from the three families: MF-based CF, memory-based CF, and data mining (detailed in Section IV). Two methods that recommend most popular and random products are used as baselines. The representative algorithms from each of the aforementioned families were selected from the MyMediaLite library[4] [7] to aid reproducibility. Semantic- and context-aware methods have not been considered due to the nature of the input data, which consists solely of purchase events. All techniques are evaluated for their predictive power and speed. Our experiments are complemented with an online survey, which assesses recommendation engines of existing e-commerce platforms. Survey responses are analyzed in Section V-C.

TABLE I: Summary for datasets D1 and D2

	D1	**D2**
Domain	Fashion	Books
Timespan	9 months 26 days	3 months 10 days
Customers	26,091	505,136
Products	4706	210,455
Events	78,449	1,623,576
Sparsity	99.936%	99.998%

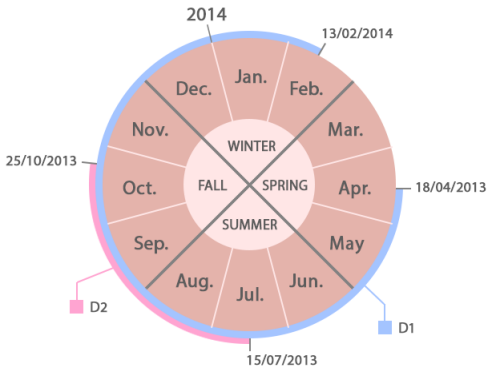

Fig. 1: Timespan of datasets

A. Datasets

The summary of the two datasets is given in Table I. Both datasets are of implicit feedback and contain a history of purchases over a period of time. Each dataset is represented as a binary purchase matrix $P^{[m \times n]}$, whose entry $P_{ui} \in \{0, 1\}$ indicates whether customer $u \in U^{[m]}$ bought product $i \in I^{[n]}$ at least once. The two datasets differ in size, sparsity, type of merchandize, and timespan (the timespan is illustrated in Figure 1). The distribution of purchases per customer is given in Figure 2, which depicts the characteristic sparsity of purchase data. For our experiments, we pre-filtered the datasets to contain customers with at least 2 purchases, which is the bare minimum that enables customer profile splitting (see Section III-B), while maintaining high sparsity levels.

B. Experimental protocol

The common strategy for offline experiments is to split the dataset into training and test sets, where the latter simulates future purchases and contains a fraction of events withheld from the original dataset. The remaining events are kept in

[4]with the exception of the ARM recommender and the LLR-based similarity, which were implemented separately

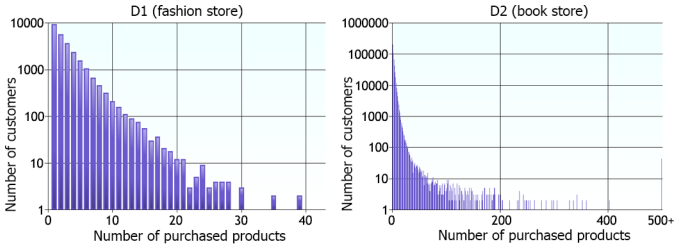

Fig. 2: Distribution of purchases per customers

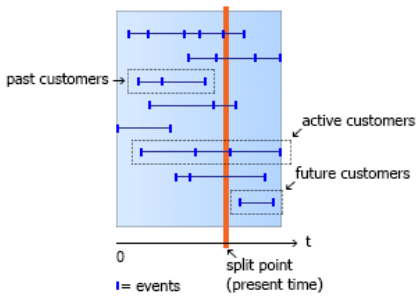

Fig. 3: Representation of a temporal dataset. Horizontal lines represent customer profiles on a timeline, where purchases are marked with blue bars.

the training set, which is fed to a recommendation engine that outputs a ranked list of unobserved products based on some underlying algorithm. The top-N products are then taken from the list as final recommendations. The choice of the split ratio between training and test sets is arbitrary and usually ranges from 80/20 to 50/50 percent.

In our experiments, we follow two splitting strategies:

1. **Traditional approach (non-chronological split)**. As it is often practised in RS literature [8], [9], [10], [11], we start by discarding "cold-start" customers with profiles below a constant threshold. In our case, we only keep those with at least 10 purchases. We then divide all customers into 5 folds for cross-validation. In each test fold, we randomly withhold half of events[5] from customer profiles to build the test set. The other half of events together with the remaining folds is used for training the models.

2. **Proposed approach (chronological split)**. The availability of timestamps allows us to attempt a more realistic setup, where we train on past events to predict future events. Furthermore, we deliberately keep all customers in place from the initial pre-filtering step, when only those with at least 2 purchases were kept. This is the shortest user profile length that can be divided in two parts (training and testing). This choice was motivated by our intention to simulate a real cold-start environment. A temporal dataset is illustrated in Figure 3. In this setting, we set a split point on the dataset's timeline (indicated with the red vertical line), which acts as our "present". Past events located to the left of the split point are used for training, whereas future events located on the right side are used for testing. Customers whose entire profiles lie in the "future" are discarded, whereas "past customers" only appear in the training set. Keeping the split point (and hence the test set) fixed, we train the models using the expanding time window, starting from the most recent history and then going back in time at a fixed rate up to full history. This approach allows us to determine an optimal training interval from our measurements.

In both cases, the task is to recommend products matching those of the test set. The motive for attempting both approaches was to determine whether the widely used random, non-chronological splits can reliably reproduce the same ranking of algorithms as the more realistic temporal, cold-start splits.

C. Evaluation metrics

Top-N recommendation lists are typically evaluated in terms of their precision and recall. Precision measures the percentage of recommended products that are relevant,[6] whereas recall measures the percentage of relevant products that are recommended.
Given a fixed size N of recommendation list, we calculate the harmonic mean of precision and recall, known as $F1\text{-}score$:

$$F1@N = \frac{1}{|\mathcal{U}|}\sum_{u=1}^{|\mathcal{U}|}\frac{2\cdot precision_u@N \cdot recall_u@N}{precision_u@N + recall_u@N} \quad (1)$$

where $\mathcal{U}$ is the set of "active customers" (see Figure 3).

In our experiments we choose $N = 5$. Notably, evaluation at fixed N suffers from bias introduced by unevenly distributed number of events in the test set across customers. When this number is larger than N, even a perfect recommender would never reach the maximum accuracy score. Therefore, metrics with fixed N do not average well and can only be used to estimate the relative performance between algorithms. To tackle this problem, we also measure precision at the level of recall, known as $R\text{-}precision$:

$$R\text{-}precision = \frac{1}{|\mathcal{U}|}\sum_{u=1}^{|\mathcal{U}|} precision_u@|T_u| \quad (2)$$

where T_u is the test set of customer u.

Thus, $R\text{-}precision$ adjusts for the size of the customer u's test set and hence ensures that evaluation has an attainable maximum value [10]. Thus, the aforementioned bias is eliminated and averaging across customers makes more sense [12]. As opposed to $F1\text{-}score$, $R\text{-}precision$ allows to estimate the absolute quality of a recommendation algorithm [10].

The disadvantage of the above evaluation metrics is that they do not capture the utility of recommendations, making them all look equally interesting to the customer. However, usability studies show that customers are used to scan recommendation lists from top to bottom, often observing only a few products on the top of the list [5]. Therefore, we use a special ranking measure to account for a positional discount down the list, known as Mean Average Precision (*MAP*):

$$MAP = \frac{1}{|\mathcal{U}|}\sum_{u=1}^{|\mathcal{U}|}\frac{1}{|T_u|}\sum_{k=1}^{|T_u|} precision_u@k \quad (3)$$

[5] if the number of events is odd, the training set will be larger than the test set by 1 event

[6] being *relevant* means being a member of the test set

where $precision_u@k = 0$ if $product\_at\_position\_k \notin T_u$

Thus, *MAP* is the arithmetic mean of precision values across recall levels, averaged over all customers. *MAP* is a very popular metric in information retrieval and is favored for its discrimination and stability [12].

The above evaluation metrics are measured within the range [0,1], where higher values are better.

D. Hyperparameter optimization via Golden Section Search

Most recommendation algorithms are affected by their hyperparameters that need to be properly "tuned" when applied to a new dataset. This is a problem of optimizing a cost function over a (multi-dimensional) configuration space. Evaluating a cost function is often computationally expensive, rendering common approaches such as manual and grid search inefficient. We address this issue by adapting a Golden Section Search (GSS) to our multi-dimensional hyperparameter optimization, under the assumption of unimodality of the objective function. In one dimension, the problem can be formulated as finding a point x in the configuration space $[a, b]$ such that the objective function $f(x)$ is maximized:

$$x^* = \arg\max_{a \leqslant x \leqslant b} f(x) \tag{4}$$

The GSS algorithm finds x^* through iterative shrinkage of the configuration space by sampling points between a and b using the inverse golden ratio $\varphi \approx 0.618$, until the resulting interval becomes smaller than the pre-set tolerance ϵ:

Algorithm 1 Golden Section Search

Input: List $\mathcal{P}(param_1\langle a_1, b_1, \epsilon_1\rangle, ..., param_n\langle a_n, b_n, \epsilon_n\rangle)$
Output: Optimal values $x_1^*, ..., x_n^*$ of parameters in $\mathcal{P}$

function MULTI-GSS($\mathcal{P}$)
  while $|\mathcal{P}| > 0$ **do** ▷ unoptimized parameters exist
    for $i \leftarrow 1, |\mathcal{P}|$ **do**
      $x_1 \leftarrow b_i - \varphi(b_i - a_i)$
      $x_2 \leftarrow a_i + \varphi(b_i - a_i)$
      if $f(x_1) < f(x_2)$ **then** ▷ x_i^* must be in $[x_1, b]$
        $a_i \leftarrow x_1$
      else ▷ x_i^* must be in $[a, x_2]$
        $b_i \leftarrow x_2$
      if $b_i - a_i \leqslant \epsilon_i$ **then** ▷ $param_i$ is optimized
        $x_i^* \leftarrow (b_i - a_i)/2$
        remove $param_i$ from $\mathcal{P}$
        return x_i^*

Given a set $\mathcal{P}$ of hyperparameters, the algorithm iterates over them and the function $f(x)$ evaluates a recommender in the current hyperparameter setting. Based on the given values of a, b, and ϵ of each hyperparameter, the required number of iterations can be pre-computed as follows:

$$\#iter = \frac{\sum_{i \in \mathcal{P}} \log \frac{\epsilon_i}{b_i - a_i}}{\log \varphi} \tag{5}$$

This allows for adjusting the value of ϵ prior to running GSS in accordance with the available computational resources. GSS has fast convergence because it never samples past a suboptimal point, but does not guarantee finding a global maximum when the objective function is multimodal. This leads to situations when the last iteration does not necessarily produce the highest function value. To tackle this problem, we save the values of hyperparameters every time when the evaluated function peaks at its current best value.

For tuning hyperparameters in our chronological setup, we used validation sets dating from 17/07/2013 to 31/07/2013 for D1 and from 29/07/2013 to 05/08/2013 for D2. We trained all algorithms on events prior to the validation events of each dataset. For the non-chronological case we validated algorithms on customers with 9 events, where random 65%-35% user profile splitting was done. This ensures that our validation sets are always disjoint from the test sets, where only customers with at least 10 events are considered.

E. Survey of deployed recommender systems

The survey was aimed at commercial and open-source e-commerce software vendors, who were asked to describe their recommendation engines as accurately as possible. This was done via an anonymous online questionnaire, which assessed the deployed RS on three criteria: a) properties of recommendations, b) data utilized for recommendations, and c) recommendation techniques. The selection of these techniques is based on the classification by Amatriain *et al.* [13], with the addition of extra (non-personalised) techniques, such as random, manual, or best selling product recommendations. Each question had a set of pre-defined answers with a possibility to select multiple options and provide a free-form answer if so desired. The questionnaire is available online at `http://goo.gl/forms/zFEfLLlFHO`.

IV. RECOMMENDATION ALGORITHMS

In this section we give a brief theoretical foundation for various algorithms that were used in our experiment.

A. MF-based methods

MF-based models decompose the original purchase matrix P into lower-rank factor matrices $W \in \mathbb{R}^{[m \times d]}$ and $H \in \mathbb{R}^{[d \times n]} (d \ll min(m, n))$, which capture latent relationships between customers and products. The interest of customer *u* in product *i* is estimated by the dot product of the corresponding factor vectors $\hat{P}_{ui} = W_{u:} H_{i:}^T$. The major challenge is to compute the mapping of customers and products to a joint latent factor space.

Most prevalent factorizations encountered in RS literature are based on singular value decomposition (SVD), where the mapping is performed iteratively either through the alternating least squares (ALS) or the stochastic gradient descent (SGD) optimization process. In our study, we examine several SVD-based algorithms which have been specifically designed for implicit feedback datasets.

1) WRMF: The Weighted Regularized Matrix Factorization model [14], [15] is a state-of-the-art algorithm based on SVD, which is formulated as a least squares problem with two core components: a regularization term to prevent overfitting, and a confidence value associated with each event. The intuition

behind this measure is that the confidence of observed events being positive is higher than the confidence of unobserved events being negative. One way to enforce this is to assign higher weights to positive entries in P, although other weighting schemes have also been discussed by Pan *et al.* [15]. The WRMF optimization criterion is given as:

$$min \sum_{ui} C_{ui}((P_{ui} - W_{u:}H_{i:}^T)^2 + \lambda(||W_{u:}||_F^2 + ||H_{i:}||_F^2)) \quad (6)$$

where C_{ui} is a confidence of event P_{ui}, λ is a regularization parameter, and $||\cdot||_F$ is a Frobenius norm of a matrix.

The factor matrices are first initialized with normally distributed random numbers. The optimization process then alternates between updating W and H as follows: [15]:

$$W_{u:} := P_{u:}C_{u:}H(H^T C_{u:} H + \lambda I)^{-1} \quad (7)$$

$$H_{i:} := P_{:i}^T C_{:i} W(W^T C_{:i} W + \lambda I)^{-1} \quad (8)$$

The advantage of ALS is that it can be easily parallelized and its running time is linear in the size of the input [14].

2) BPRMF: The BPRMF [9] model is formulated as a ranking problem and is based on Bayesian Personalized Ranking optimization criterion (BPR-OPT). Unlike WRMF which scores single products in P, BPRMF optimizes for correct ranking of product pairs for each customer, i.e. the triple $\langle u, i, j \rangle$. The basic idea of BPR is that if customer u consumed product i but did not consume product j, then $P_{uj} < P_{ui}$ ranking-wise. The generic BPR-OPT criterion is given as [9]:

$$\textit{BPR-OPT} = \sum_{uij} \ln \sigma(\hat{x}_{uij}) - \lambda_\theta ||\theta||^2 \quad (9)$$

where $\sigma(x)$ is a logistic sigmoid $1/(1+e^{-x})$, $\hat{x}_{uij}$ is an arbitrary function that specifies the relationships in $\langle u, i, j \rangle$, θ is a parameter vector of an arbitrary model, and λ_θ is a model-specific regularization parameter.

In BPRMF, the estimation of $\hat{x}_{uij}$ is performed through matrix factorization. But since the MF model can only predict single events, the estimator is decomposed into single prediction tasks: $\hat{x}_{ui} - \hat{x}_{uj}$. The optimization is performed through SGD with bootstrap sampling of training triples using the following update rule [9]:

$$\theta := \theta + \alpha(\frac{e^{-\hat{x}_{uij}}}{1 + e^{-\hat{x}_{uij}}} \frac{\partial}{\partial \theta} \hat{x}_{uij} + \lambda_\theta \theta) \quad (10)$$

where α is the learning rate.

B. Data mining (association rules)

Association rules mining (ARM) has been a popular instrument for market basket analysis (e.g. in supermarkets), where associations between products are established from customer purchase patterns. This technique is therefore a natural choice for recommender systems and has demonstrated good performance on real-world e-commerce datasets, as shown by Pradel *et al.* [3]. In its simplest form, an association rule $i \Rightarrow j$ implies that whenever a customer buys product i, he or she is likely to buy product j. The strength of a rule is measured on the basis of its confidence and support, which are calculated as follows:

$$con(i \Rightarrow j) = \frac{P_{:i}^T P_{:j}}{\sum_{u \in U} P_{ui}} \qquad sup(i \Rightarrow j) = \frac{P_{:i}^T P_{:j}}{\sum_{u \in U} \sum_{i \in I} P_{ui}}$$

In other words, confidence is the probability of buying product j given that product i was bought, while support is the fraction of events where products i and j were bought together.

The training phase involves computation of all available rules, whose confidence values are stored in a sparse matrix of size $n \times n$. To produce recommendations for customer u, a set of supported rules is first identified from a customer profile. Then, unobserved products are typically ranked by either the maximum confidence [3], [11] or the sum of confidences [8] of corresponding rules. In our case, the linear combination of confidence and support was found to yield better results:

$$rank(j) = \sum_{i}^{|I_u|} con(i \Rightarrow j) \cdot sup(i \Rightarrow j) \quad (11)$$

where I_u is the profile of customer u (e.g. the set of observed products), and j is a candidate product for recommendation.

C. Memory-based CF (K-nearest neighbors)

Memory-based CF relies on the notion of similarity between customers or products. Thus, the key training step is the formation of a neighborhood, which becomes a source of recommendations. For a given customer, recommendations can be computed either from the neighborhood of similar customers (user-based KNN), or from the neighborhood of products similar to those he or she has bought (item-based KNN). Although the latter approach has been favored in RS literature, we found user-based KNN to be more effective on both our datasets, in line with the results of [4], [16].

The performance of a KNN recommender is much dependent on the similarity measure used to calculate the neighborhood. Cosine and Jaccard similarity are commonly used for this task. In our tests, the Log-Likelihood ratio proposed by Dunning [17] and adapted for CF by Owen *et al.* [18] produced superior results. In short, this metric estimates the probability that the overlap of events in customer profiles is not due to chance [18]. Let I_u and I_v denote two customer profiles. The Log-Likelihood ratio (LLR) is then defined as:

$$LLR = f(I_u, I_v) \cdot (H[I_u, I_v] - H[I_u] - H[I_v]) \quad (12)$$

where $f(I_u, I_v)$ is a count-dependent scaling factor, $H[I_u, I_v]$, $H[I_u]$ and $H[I_v]$ are the entropy of the joint and the marginal probabilities, respectively (refer to [17], [19] for details).

During training, an LLR-based neighborhood is calculated for each customer. The testing phase ranks unobserved products based on the accumulated LLR scores of each neighbor who consumed the products in question.

V. Results

In this section, we present the results from our experiments on datasets D1 and D2. All algorithms were tuned using GSS and then evaluated on the three metrics described in Section III-C. After reporting the experimental results, we also present a summary of the survey responses.

TABLE II: Scores for the random split on D1

Recommender	*MAP*	*F1@5*	*R-prec*	*Running Time*
WRMF	0.0927	0.1013	0.1036	00:05:21
BPRMF	0.0650	0.0676	0.0698	00:00:32
UserKNN	0.1000	0.1078	0.1073	00:00:05
ARM	**0.1100**	**0.1162**	**0.1155**	00:00:01
Most Popular	0.0523	0.0458	0.0468	00:00:00
Random	0.0028	0.0000	0.0010	00:00:00

TABLE III: Scores for the random split on D2

Recommender	*MAP*	*F1@5*	*R-prec*	*Running Time*
WRMF	0.1293	0.1322	0.1363	07:10:10
BPRMF	0.0188	0.0148	0.0187	00:12:46
UserKNN	**0.1796**	**0.1859**	**0.1854**	00:56:21
ARM	0.1774	0.1813	0.1825	00:16:24
Most Popular	0.0139	0.0149	0.0156	00:07:02
Random	0.0002	0.0001	0.0001	00:09:09

TABLE IV: Average scores for the chronological split on D1

Recommender	*MAP*	*F1@5*	*R-prec*	*Running Time*
WRMF	0.0244	0.0188	0.0147	00:05:27
BPRMF	0.0214	0.0121	0.0095	00:00:06
UserKNN	0.0312	**0.0229**	0.0222	00:00:06
ARM	**0.0334**	0.0214	**0.0225**	00:00:01
Most Popular	0.0206	0.0113	0.0080	00:00:00
Random	0.0037	0.0014	0.0009	00:00:00

TABLE V: Average scores for the chronological split on D2

Recommender	*MAP*	*F1@5*	*R-prec*	*Running Time*
WRMF	0.0351	0.0260	0.0224	02:27:10
BPRMF	0.0160	0.0130	0.0114	00:18:37
UserKNN	0.0469	0.0363	**0.0352**	00:50:14
ARM	**0.0488**	**0.0377**	0.0346	00:20:35
Most Popular	0.0150	0.0114	0.0094	00:10:24
Random	0.0001	0.0000	0.0000	00:14:42

A. Experiment 1. Non-chronological split

We first present the results from the "traditional" approach, where we have:

- test customers with at least 10 purchases
- random sampling of events (50/50 profile splitting)
- 5-fold cross-validation

The performance of algorithms in this setup is given in Tables II and III. It can be seen that UserKNN and ARM outperform MF-based methods on all metrics, which is especially prominent in case of D2, where BPRMF has performance comparable to that of the most popular books recommender. This finding conforms to those reported in [4], [3].

The presented runtimes of algorithms were calculated as the sum of training and testing time. In Tables IV and V the average running times (across training sets) are reported. The results are given in the format *hh:mm:ss*. Aside from the implementation details, the running time of algorithms depends on their hyperparameter values, such as K for UserKNN or *iterations / factors* for MF-based methods. We can see that WRMF is consistently slower than other methods, despite having been trained with fewer iterations than BPRMF in all cases. We attribute it to the fact that ALS involves a least squares solution, which is more expensive than an SGD iteration. The noticeable boost in WRMF speed in Table V is merely because the algorithm was trained with fewer factors than in random mode (determined by the GSS).

B. Experiment 2. Chronological split

The second experiment was conducted maintaining the cold-start issue in the data and splitting the datasets chronologically using the expanding time window approach described in Section III-B. The set partitioning was performed on a monthly basis for D1 and on a bi-weekly basis for D2. The test set contained the last month of data for D1 and the last 2 weeks of data for D2. The training sets were of increasing length from 1 to 9 months for D1 and from 2 to 12 weeks for D2. The different partitioning scales were chosen because of considerably shorter timespan of D2 (even though it contained much more events). Figures 4 and 5 show the performance of algorithms for varied history lengths. Purple dots indicate peaks in performance.

From observing the performance peaks in Figure 4, it becomes clear that the optimal training history for D1 lies within the 3 recent months, which appears logical for a fashion store. A similar trend of favoring recent events is observed in Figure 5, with the exception of UserKNN, whose performance increases proportionally with the size of the history. It can also be seen that best-seller recommendations are generally most effective when recommending the most recently sold merchandize. These observations suggest the adoption of time-aware recommenders in e-commerce, which have shown improved quality of predictions in the movie ratings domain [20].

Tables IV and V show the performance of algorithms after averaging their accuracy scores over all training sets. The results indicate that in general, time-based splits preserve the same ranking we observed in random splits, although with less significant difference in performance scores. The low measurement values is the consequence of cold-start conditions: predicting purchases for customers with short profiles is difficult (but realistic). However, this is not the only reason. We extended our experiment 2 by keeping test customers with at least 10 purchases. We then performed a chronological split followed by a 50/50 random split (as in experiment 1). The effect we observed after evaluating algorithms in these two protocols was quite remarkable: randomizing events in customer profiles boosted the performance of some algorithms by more than 300% (e.g. for WRMF). This observation tells that preserving the sequence of events during evaluation is crucial and that traditional random splits may produce too optimistic estimates of a recommender's accuracy.

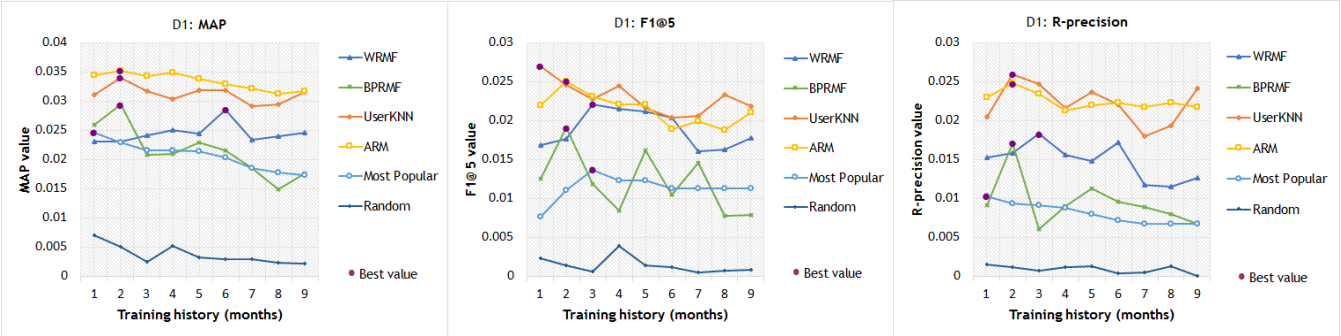

Fig. 4: Chronological split of D1 with 9 training sets

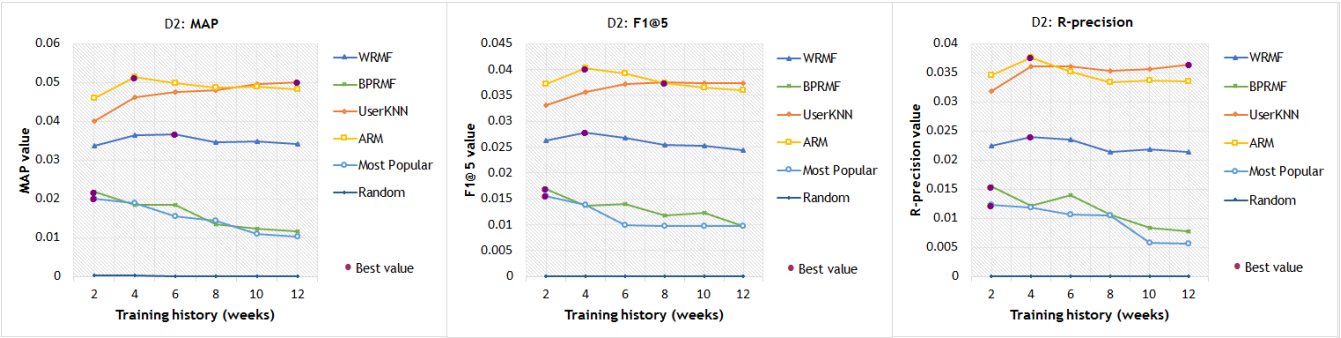

Fig. 5: Chronological split of D2 with 6 training sets

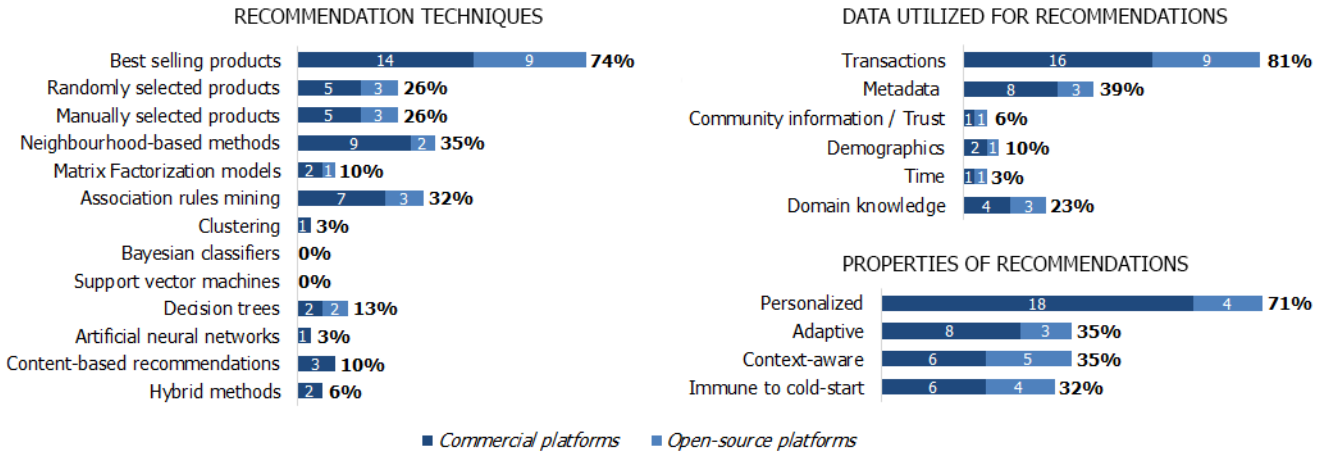

Fig. 6: Survey responses

C. Survey responses

31 responses were obtained in total from 20 commercial and 11 open source e-commerce platforms[7]. The responses to each of the 3 questions are summarized in Figure 6. Below we discuss some of the interesting findings.

Transactional data is used by more than 80% of respondents, which points to the clear dominance of CF over the content-based approach. At the same time, timestamps of events are neglected by most companies, which seems unreasonable in the light of our experimental findings. Other contextual sources of data have also shown low usage, even though more than a third of recommendation engines have been reported to be context-aware. This must be due to the comparatively higher percentage of ad-hoc recommenders based on domain knowledge.

It seems that commercial platforms tend to put much more effort in personalization than their open-source competitors. We observe that neighborhood-based filtering and association rules mining are the most popular personalized recommenders, which totally conforms to our experimental results.

[7]Among participants who chose not to stay anonymous were VP-Cart, Ecwid, WebJaguar, YesCart, NitroSell, S3Cart, Graphflow, NationKart, Whirlwind eCommerce, and PinnacleCart

Recommending *best selling* products is the preferred approach for most platforms. Indeed, our experiments show that outperforming best-seller lists can be challenging even to personalized recommenders. Furthermore, trivial methods of recommending *random* or *manually selected* products are more widespread than state-of-the-art methods, such as MF.

It appears that some platforms can be opposed to RS as such. One of the respondents argued that "if a shopper goes to all the trouble to choose the right product, it is almost insulting to link them to a page that says 'if you don't like this, then here are some recommendations.' Adding recommendations to a page that is the end result of a buying decision is a distraction and is likely to lower the rate of conversion".

VI. Conclusions

Recommender systems is an important research field in academia and a vital revenue-generating tool in e-commerce. In this paper, we raised the question of the adoption of techniques in industry and the effectiveness of recent machine learning models for typical e-commerce settings, where explicit ratings are inexistent and implicit information is sparse.

We have shown that deployed e-commerce platforms take rather simplistic approach to product recommendations, with best-seller lists being at the heart of their engines. When it comes to personalized recommendations, the incline toward "good old methods" is not as surprising as it seems: our experiments confirm the superior performance of KNN collaborative filtering and association rules mining to more advanced MF-based methods when applied on binary purchase data. Similar findings have been reported in [3].

Whereas in academic context researchers tend to prioritize accuracy over speed (according to [21], only 11% of RS research papers report runtimes), in industrial settings where real-time interaction with the system is employed, the time factor is undoubtedly vital. This might be another reason in favor of simpler techniques. We show that time is also important in terms of the recency and the sequence of events. Depending on the type of merchandize, e-commerce datasets exhibit varied effects of seasonality. Thus, our experiments indicate that training on all available history is in most cases not optimal. We present a time-based splitting strategy that helps to establish the optimal timespan of a training set. This optimization can lead to improved recommendation accuracy and runtime efficiency. We also show that traditional random sampling, which often implies training on future events to predict past events, is prone to producing too optimistic performance estimates due to capturing erroneous shopping patterns.

Finally, we propose fast Golden Section Search for automatic tuning of algorithms with multiple hyperparameters.

VII. Acknowledgment

We express our gratitude to Apptus Technologies for the provided datasets and computational resources.

References

[1] M. Marshall. (2006, Dec.) Aggregate knowledge raises $5m from kleiner, on a roll. Accessed: 2015-03-12. [Online]. Available: http://venturebeat.com/2006/12/10/aggregate-knowledge-raises-5m-from-kleiner-on-a-roll/

[2] X. Amatriain and J. Basilico. (2012, Apr.) Netflix recommendations: Beyond the 5 stars. Accessed: 2015-03-12. [Online]. Available: http://techblog.netflix.com/2012/04/netflix-recommendations-beyond-5-stars.html

[3] B. Pradel, S. Sean, J. Delporte, S. Guérif, C. Rouveirol, N. Usunier, F. Fogelman-Soulié, and F. Dufau-Joel, "A case study in a recommender system based on purchase data," in *Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ser. KDD '11, 2011, pp. 377–385.

[4] F. Aiolli, "Efficient top-n recommendation for very large scale binary rated datasets," in *Proceedings of the 7th ACM Conference on Recommender Systems*, 2013, pp. 273–280.

[5] G. Shani and A. Gunawardana, "Evaluating recommendation systems," in *Recommender Systems Handbook*, F. Ricci, L. Rokach, B. Shapira, and P. B. Kantor, Eds. Springer-Verlag New York, Inc., 2009, ch. 8, pp. 257–297.

[6] Z. Huang, D. Zeng, and H. Chen, "A comparative study of recommendation algorithms for e-commerce applications," *IEEE Intelligent Systems*, 2006.

[7] Z. Gantner, S. Rendle, C. Freudenthaler, and L. Schmidt-Thieme, "Mymedialite: A free recommender system library," in *Proceedings of the 5th ACM Conference on Recommender Systems*, 2011.

[8] C. Kim and J. Kim, "A recommendation algorithm using multi-level association rules," in *Proceedings of the 2003 IEEE/WIC International Conference on Web Intelligence*, ser. WI '03, 2003.

[9] S. Rendle, C. Freudenthaler, Z. Gantner, and L. Schmidt-Thieme, "Bpr: Bayesian personalized ranking from implicit feedback," in *Proceedings of the Twenty-Fifth Conference on Uncertainty in Artificial Intelligence*, ser. UAI '09, 2009. [Online]. Available: http://dl.acm.org/citation.cfm?id=1795114.1795167

[10] A. Said, A. Bellogín, and A. de Vries, "A top-n recommender system evaluation protocol inspired by deployed systems," *LSRS Workshop at ACM RecSys*, 2013.

[11] B. Sarwar, G. Karypis, J. Konstan, and J. Riedl, "Analysis of recommendation algorithms for e-commerce," in *Proceedings of the 2Nd ACM Conference on Electronic Commerce*, ser. EC '00, 2000, pp. 158–167.

[12] C. D. Manning, P. Raghavan, and H. Schutze, "Evaluation in information retrieval," in *Introduction to Information Retrieval*. Cambridge University Press, 2008, pp. 151–175.

[13] X. Amatriain, A. Jaimes, N. Oliver, and J. Pujol, "Data mining methods for recommender systems," in *Recommender Systems Handbook*, F. Ricci, L. Rokach, B. Shapira, and P. B. Kantor, Eds. Springer-Verlag New York, Inc., 2010, ch. 2, pp. 257–297.

[14] Y. Hu, Y. Koren, and C. Volinsky, "Collaborative filtering for implicit feedback datasets," in *Proceedings of the 2008 Eighth IEEE International Conference on Data Mining*, ser. ICDM '08, 2008, pp. 263–272.

[15] R. Pan, Y. Zhou, B. Cao, N. N. Liu, R. Lukose, M. Scholz, and Q. Yang, "One-class collaborative filtering," in *Proceedings of the 2008 Eighth IEEE International Conference on Data Mining*, ser. ICDM '08, 2008, pp. 502–511.

[16] E. G. Vozalis and K. G. Margaritis, "Recommender systems: An experimental comparison of two filtering algorithms," in *Proceedings of the Ninth Panhellenic Conference in Informatics*, 2003.

[17] T. Dunning, "Accurate methods for the statistics of surprise and coincidence," *Comput. Linguist.*, vol. 19, no. 1, pp. 61–74, 1993. [Online]. Available: http://dl.acm.org/citation.cfm?id=972450.972454

[18] S. Owen, R. Anil, T. Dunning, and E. Friedman, *Mahout in Action*. Manning Publications Co., 2011.

[19] T. Dunning, "Surprise and coincidence - musings from the long tail," Online: http://tdunning.blogspot.se/2008/03/surprise-and-coincidence.html, mar 2008, accessed: 2015-03-09.

[20] Y. Koren, "Collaborative filtering with temporal dynamics," in *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ser. KDD '09, 2009.

[21] J. Beel, S. Langer, M. Genzmehr, B. Gipp, C. Breitinger, and A. Nürnberger, "Research paper recommender system evaluation: A quantitative literature survey," in *Proceedings of the International Workshop on Reproducibility and Replication in Recommender Systems Evaluation*, ser. RepSys '13, 2013.