# Predicting Online Performance of News Recommender Systems Through Richer Evaluation Metrics

Andrii Maksai, Florent Garcin, and Boi Faltings
Artificial Intelligence Lab, Ecole Polytechnique Fédérale de Lausanne
Lausanne, Switzerland
{firstname.lastname}@epfl.ch

## ABSTRACT

We investigate how metrics that can be measured offline can be used to predict the online performance of recommender systems, thus avoiding costly A-B testing. In addition to accuracy metrics, we combine diversity, coverage, and serendipity metrics to create a new performance model. Using the model, we quantify the trade-off between different metrics and propose to use it to tune the parameters of recommender algorithms without the need for online testing. Another application for the model is a self-adjusting algorithm blend that optimizes a recommender's parameters over time. We evaluate our findings on data and experiments from news websites.

## Categories and Subject Descriptors

H.3.3 [**Information Storage and Retrieval**]: Information Search and Retrieval—*Information filtering*

## Keywords

recommender system;online evaluation;evaluation metrics

## 1. INTRODUCTION

It was recognized early on in the history of recommender systems (recsys) that the most accurate recommendations were not always the best. The first approaches for diversifying recommendations were made by Zhang et al. [29] and Ziegler et al. [31]. Later, many different metrics - such as novelty, coverage, diversity, and serendipity [17] - have been introduced with the aim of enhancing the quality of recommendations. It has even been suggested that the focus on optimizing the accuracy of recsys has been detrimental to the field [13]. Ordering algorithms with respect to their offline accuracy can result in the exact inverse of ordering them with respect to the online click-through rate (CTR), which is the metric most site owners care about [6, 27].

The main reason for this is that recommending popular items is usually accurate – they are popular because people do indeed rate them highly – but such recommendations have little to no effect as people have already seen the items elsewhere. A contributing factor in the domain of news recommendations is that users are often interested in something entirely new. Finally, users often view recommendations concerning a variety of different topics more favorably than the ones concerning several interesting but similar items.


*RecSys'15,* September 16–20, 2015, Vienna, Austria.

DOI: http://dx.doi.org/10.1145/2792838.2800184.

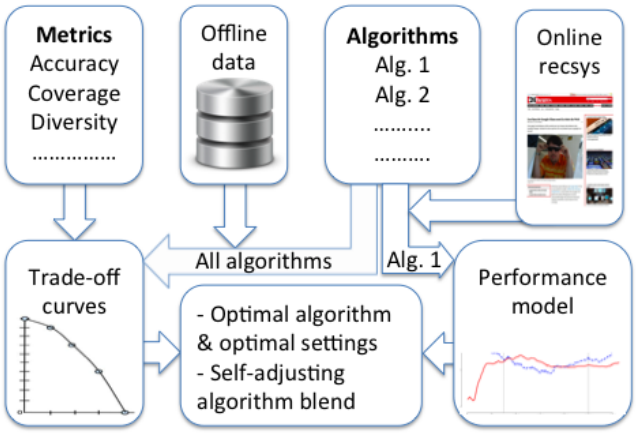


**Figure 1: Workflow for selecting the best algorithm using a performance model with multiple metrics.**

All of the above factors indicate that optimizing the accuracy of recommendations using offline data, gathered from past behavior without a running recommender algorithm, is neither an effective nor efficient way to select the best algorithm [13, 19]. The fairest way to compare algorithms is to launch them online and compare the actual reactions of users to the recommendations. However, this requires the existence of an online environment and a set of dedicated users, and it takes a long time. Another problem is the need of constant re-evaluation of algorithms, especially if they are sensitive to changes in the item or user set over time [2, 10]. Simulation of an online environment is a potential alternative [11, 27]. We discuss these solutions later in the paper.

In this work, we leverage recommendations from real news websites to model their CTR (Fig. 1). Our main contributions are threefold. First, in Sec. 2, we review the different metrics, find those that are most likely to affect online performance, and indicate the presence of a trade-off between them. Next, in Sec. 3, we combine a selected subset of metrics into a prediction model of online performance. Finally, in Sec. 4, using this model and the metric trade-offs, we

show how to select the best version of an algorithm and its parameters using limited online evaluation, and how to create a blend of several recommender algorithms that adjusts over time for optimal performance. Results reported in Sec 5 show that given limited access to online environment, it is possible to model performance over time far better than by averaging performance over time.

## 2. EVALUATION METRICS

Over the years, various metrics have been suggested for evaluating recsys. In the approach presented here, we considered 17 metrics classified into five different groups [17].

### 2.1 Metric groups

**Accuracy/Error** metrics compute the quality of predictions (i.e., rating movies or a correct/incorrect guess of the next news item visited). Error metrics usually penalize errors, whereas accuracy metrics reward correct answers. We ignored metrics, such as RMSE or MAE, that penalize each item separately; we concentrated instead on metrics for the task of top-N recommendation. We investigated the following metrics: Precision, NDPM, Kendall's $\tau$, Spearman's $\rho$ [8], Success [6], Markedness, Informedness, and Matthew's Correlation [15]. All these metrics compute the accuracy of prediction and, for the case of top-1 recommendation, turn into binary indicators of the success of recommendation.

**Diversity** is usually defined as a measure of dissimilarity between items in the recommendation list with respect to a similarity metric. This "intra-list" diversity has been presented and used in several forms [29, 31, 28].

Lathia et al. [10] proposed a definition of the temporal diversity of a recsys which was dependent on the number of new items a user was shown during different visits. For news recommendations where users are often anonymous, we modified the definition to compute the amount of new items the system recommends to all users at a later time.

Zhou et al. [30] proposed a metric called "personalization" that is effectively the normalized pairwise Jaccard similarity between items recommended to each pair of users; it can be viewed as the "extra-list" diversity.

**Novelty** is the quality of an item of being new to a user (i.e., the recommendation of an item from the category that the user already likes is not novel). A variation of such a metric, called "surprisal", is a weighted sum of negative log frequencies of the items in the recommendation list [30].

**Coverage** is defined as the percentage of items that are ever recommended, and prediction coverage is the number of users for whom a recommendation can be made. We also considered the Gini Index and Shannon's Entropy [17].

**Serendipity** is the quality of being both unexpected and useful. One component penalizes the expectedness of the most popular items, whereas the other component measures usefulness – typically just accuracy [7, 14].

### 2.2 Metric correlations

We investigated the correlation between the metrics in each group by applying three different algorithms to the Swissinfo dataset, described later. Each data point was generated by averaging metric values for all the recommendations at equal time intervals. For metrics that are computed cumulatively over recommendations (i.e., temporal diversity) we computed the metric value at each point in time and used the cumulative difference as the metric value.

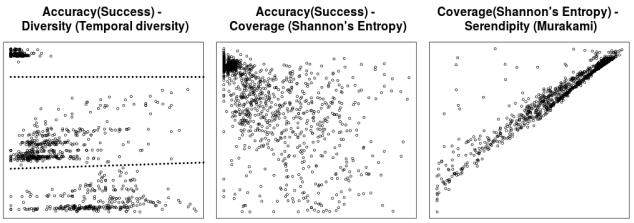


**Figure 2: Pairwise plots between representative metrics from four groups (average over 10 minutes). Example of 3 clusters separated by the dotted lines.**

The metrics in the Accuracy group (NDPM, Kendall's $\tau$, Success, Spearman's $\rho$, Markedness, Informedness and Matthews correlation) showed correlations higher than 0.9 between all the pairs of metrics for the recommendation lists given by the algorithms. The same results were obtained for the other metric groups – Diversity (Intralist diversity, Personalization, and Temporal diversity), Coverage (Gini Index, Shannon's Entropy, and Coverage), and Serendipity (Serendipity by Ge [7], and Serendipity by Murakami [14]). We therefore examined correlation between pairs of "representative" metrics from each of these groups (Fig. 2). There was no strong agreement between metric groups (except for Coverage and Serendipity). This indicates that different metric groups all express different features of the recommendations and therefore at least one representative of each group should be used as a feature of the performance model.

Points often formed three distinct clusters. The clusters corresponded to recommendations given by the different algorithms, indicating that the relationship between metrics might be different for recommendations given by different algorithms.

The results in Fig. 2 were obtained from the recommendation lists comprising three items. We studied the effect of the length of recommendation lists on metric correlations (Fig. 3). For all metric pairs, we first observed a drop, and then almost no change in the absolute value of correlations. This indicates that different sets of metrics might be important for domains with different numbers of items to be recommended.

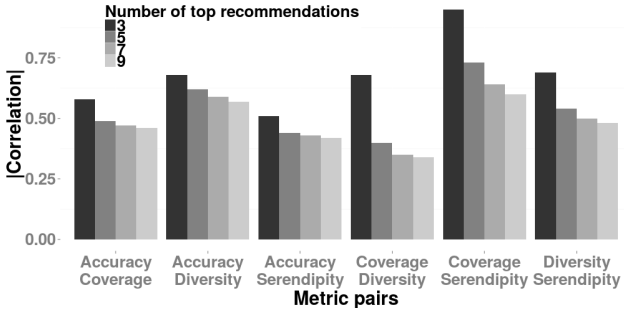


**Figure 3: Absolute value of correlations between metrics in different groups for different numbers of recommendations.**

### 2.3 Metric trade-off

Algorithms often have hyperparameters that affect their performance, and by varying the values of such hyperpa-

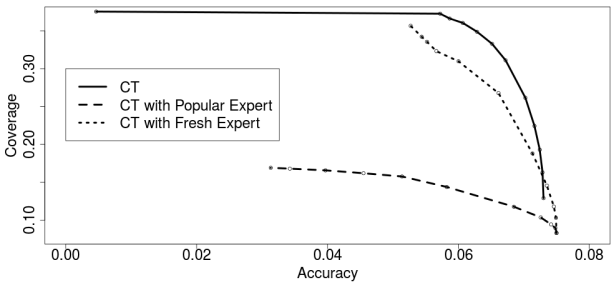


**Figure 4: Coverage–accuracy trade-off curves for three variations of the CT algorithm [5]**

rameters different sets of recommendations can be obtained. The average metric values for all the sets of recommendations given can be computed in order to observe how they change when varying a hyperparameter. The example shown in Fig. 4 was obtained using the Yahoo dataset and several variations of the Context Tree (CT) algorithm, described later in the paper. Each of the algorithms produced a curve that clearly indicated the trade-off between metrics – in this case, Accuracy and Coverage. We also observed these trade-offs using other datasets and algorithms. Getting the best performance requires selecting which metrics to optimize. The next section describes how we build a regression model of performance to achieve this.

# 3. PREDICTING ONLINE FROM OFFLINE

In this section, we first briefly describe the definitions of offline and online accuracy, and click-through rate. We then describe our method of feature selection for the regression model of online metrics, and finally the model itself.

**Offline accuracy** is the percentage of clicks predicted by the recsys when it is applied to a log of user browsing which occurred without the recommender system present.

**Online accuracy** is the percentage of clicks predicted by the recsys when it is online. If a recsys predicted that the user would browse to a particular page and she did so, but without clicking on its recommendations, this still counts towards online accuracy.

**Click-through rate** (CTR) is the percentage of clicks made on recommendations.

For any random user, the model assumes that all the items she visits without a recsys are included in the set of items she would visit with one. This means that if the user visits an item when using the recsys, that she would visit anyway when the recsys was absent, then that click should not be taken into account when measuring the impact of the recsys. A broader discussion of this topic is presented by Garcin et al. [6].

CTR and online accuracy are both important metrics for recsys. Therefore, we build a regression model for each of them. The regression model will use not only offline accuracy, as it is something different from the two above.

## 3.1 Feature selection

To verify the finding that multiple groups of metrics, such as Diversity and Coverage, are important for predicting online performance metrics, we carried out feature selection using the Least Angle Regression (LAR, [4]). The LAR assumes a linear model of the relationship between independent variable $y$ and $n$ dependent variables $x = (x_1, \ldots, x_n)^\intercal$, with an $L_1$ regularizer:

$$y = \beta^\intercal x + \lambda \sum_{j=1}^{n} |x_j|$$

The $L_1$ regularizer promotes sparsity in $\beta$. By decreasing $\lambda$, it is possible to assign each predictor a value of $\lambda$ at which it first enters the model with a non-zero weight. The order of the predictors given by these $\lambda$ values serves as a proxy for their importance. The LAR allows efficient computation of this value for each regressor [4].

The average order position among folds was calculated using average metric values in time intervals of length $\Delta t$ as predictors and average CTR as responses. Details of this approach are given in Alg. 1. Several $\Delta t$ interval sizes were tested, but all values of ten minutes or longer were found to work reasonably well. Ten minute intervals were chosen as shorter intervals gave results with very high variance and longer intervals meant fewer data points and, therefore, less significant results. We used $F = 100$ folds.

**Algorithm 1:** Average model entering time for each metric. $t(i)$ indicates time when recommendation $i$ was given.

**input** : Offline, online data $D$,$D'$,metrics $M$,$\Delta t$,$F$
**output**: Average model entering time $T(m) \forall m \in M$

$T(m) \leftarrow 0 \; \forall m \in M$
**for** $f \leftarrow 1$ **to** $F$ **do**
  $D_f \leftarrow$ Random 10% of $D$
  $D'_f \leftarrow$ Random 10% of $D'$
  Data $\leftarrow \emptyset$
  **for** $w = \min_{i \in D \cup D'} t(i) : \Delta t : \max_{i \in D \cup D'} t(i)$ **do**
    $\mathrm{F}^w_m \leftarrow$Avg($\{m(i) | i \in D_f, t(i) \in [w; w + \Delta t]\}$
    $\mathrm{R}^w \leftarrow$Avg($\{CTR(i) | i \in D'_f, t(i) \in [w; w+\Delta t]\}$
    Data $\leftarrow$ Data $\cup(\mathrm{F}^w_1, \ldots, F^w_m, R^w)$
  **end**
  $T(m)$ += LAREntryIdx(Data, $m$) / $F \; \forall m \in M$
**end**

## 3.2 Regression model

After identifying the best predictors, we used the multiple linear regression $y = \beta^\intercal x$. This simple model allowed us to interpret coefficients of $\beta$ as trade-offs between different metrics for a particular model, or as derivatives of the performance with respect to metrics. We expand on this idea in Sec. 4. More complex models, such as Gaussian process regression or penalized linear regression, do not allow such a simple interpretation. Nevertheless, the results obtained using simple linear regression were compared to those of more complex methods. To build such a model, a limited amount of training data is required. Features should be collected simultaneously using the metrics from a recsys run on a log of offline data (user browsing without the recsys), and the online performance metric should be collected from the live website that is using a recsys.

# 4. ALGORITHM OPTIMIZATION

In this section we discuss two possible ways of using the model of online performance metric. The first allows an effective comparison between several variations of the algorithm, without the need for lengthy access to online data, and the selection of the optimal hyperparameters. The second describes an algorithm able to rebuild the model over time, continually aiming for optimal online performance.

## 4.1 Optimal algorithm selection

A typical task for recsys designers is the comparison of several variations of an algorithm, and the selection of the hyperparameter value and algorithm that perform best. An obvious solution would be to evaluate each of the algorithms online using several values of hyperparameter.

In our model, online performance can be approximated by a weighted combination of two offline metrics (i.e. Accuracy and Coverage). To simplify the argument below, we assume that weights are positive but approach trivially extends to the case when they are negative. Such a model is learned by evaluating one of the algorithm variations online and is assumed to be similar for other variations.

As described in Sec. 2.3, varying an algorithm's hyperparameter produces the metrics trade-off curve. When the curve for one algorithm is located above the curve for a second, the first algorithm is strictly better in terms of performance. Note that curves are produced using offline data.

Given the trade-off curves, there is no need for an online evaluation of all the combinations of algorithms and hyperparameters, but only those that produce points on the upper envelope of the curves. Furthermore, by inspecting the model coefficients it is possible to select an algorithm without evaluating it: if a model gives a much larger weight to one of the metrics, this can be used as a proxy for performance and the best algorithm is the one reaching the highest values for that metric.

Examples of real trade-off curves are shown in Fig. 4. We used several variations of the CT recommender as a source of recommendations ordered by Accuracy [5]. The standard CT algorithm makes predictions based on a count of the items viewed. CT with additional experts exploit the item click count and the last time an item was clicked. We used the items' Shannon's Entropy as a coverage score. Items were ordered by the weighted combinations of Accuracy and Coverage, and we varied the weight ratio, which was thus regarded as a hyperparameter for the algorithms. Curves were computed using the Yahoo dataset, described later.

Results such as these can be used for tuning a recommendation algorithm to a new site by picking the optimal value of a hyperparameter and the best algorithm. For example, for a site where the learned model gives a significant coefficient to coverage, the plain CT algorithm will be the best, whereas in cases where the accuracy coefficient dominates strongly, the "fresh" expert will be useful. Note also that the "popular" expert is never going to be best. The trade-off curves obtained from offline data can therefore be used to save on online experiments to determine the optimal algorithm and hyperparameter.

## 4.2 Self-adjusting algorithm blend

We applied the online performance model to a setting where we wanted to optimize the blend of several algorithms over time, given full access to the online environment. Let the algorithm give each item a rating based on the weighted combination of ratings assigned by several base recommenders: $F(i) = W^\intercal(F_1(i), \ldots, F_m(i))$. For each recommender $F_a$, we introduce a latent variable $Z_a$. $Z_a(t)$ measures how close the items, recommended at time $t$ by the main algorithm, are to the top items recommended by $F_a(t)$. Alternatively, $Z_a(t)$ can measure how high this recommender rates them. We build a regression model of online performance at each time $t$, based on the regressors $Z_1(t), \ldots, Z_m(t)$. In this model, the positive weight of a regressor $Z_a$ suggests that giving recommendations with increased $Z_a$ in the future would improve online performance. Coefficients of linear regression $\beta^\intercal$ effectively form a gradient of online performance with respect to the latent variables. We can therefore perform a gradient descent, updating the weights, with which we mix different base recommenders: $W_{T+1} = W_T + \lambda * \beta_T$, with $T$ and $T+1$ corresponding to two consecutive time frames, and $\beta_T$ being the coefficients of LR model, fitted on the data from frame $T$. This could be an effective alternative to A–B testing, which is analogous to a grid search in the space of all possible weight coefficients. In a simple case where $Z_a$ are ratings given by base recommenders to the items recommended by the main algorithm, and recommenders $F_a$ optimize a particular set of metrics, this approach is equivalent to modeling CTR using the set of metrics as regressors over time. This is especially important for sites with a dynamic user base, that, for example, prefers fresh news in the morning and a more diverse set in the evening.

# 5. RESULTS

In this section, the datasets used in the present experiments are described and then the results of several of those experiments are shown. First, feature selection showed that multiple groups of metrics were important for the prediction of online performance. Second, we demonstrate the regression model's performance using different feature sets on different datasets. We subsequently offer examples of the applications we described in the previous section. We finish by describing the results obtained using unbiased offline evaluation [11] and by discussing why this method is not generally applicable.

## 5.1 Datasets

We use two news datasets which have online and offline browsing logs and online evaluation on a news website.

**Swissinfo** dataset is a combination of three weeks' worth of offline and online browsing logs from the live news website *swissinfo.ch*. The offline data includes more than 227k clicks on 28,525 stories by around 188k users. The online data was gathered in the presence of three recommendation algorithms – random recommendations, most popular recommendations, and Context Tree (CT, [5]). 168k clicks were distributed almost equally between the three algorithms. Three recommendations were made to each user, and items to be recommended were selected from the pool of the last 200 unique articles visited. All users were identified solely by their browsing session, and the only information gathered about the users was from their browsing behavior.

**Yahoo! Front page** dataset is specifically tailored for unbiased offline evaluation [11]. It comprises 15 days' worth of clicks data from the main page of Yahoo!News. Each visit to the page was described by a binary vector of features. The

item pool for recommendations always contains 20 items. The log consists of nearly 28M visits to a total of 653 items.

To make the dataset more suitable for news recommendations, we identified visits belonging to the same browsing session by selecting only visits with at least 50 binary features present. For visits with the same binary features, we assumed that visits were same session if the time between visits did not exceed 10 minutes. Otherwise, we assumed that these were visits from different sessions. This procedure decreased the total number of clicks in the log to $\approx 5.7$M.

With sessions established, online browsing logs were generated using the algorithm from [11] (Section 5.5). For each algorithm, the number of clicks in the simulated browsing logs was around $\approx 285$k. To generate offline browsing logs, we took a random 10% of user sessions; they contained 573k clicks by 401k users.

**LePoint** dataset contains 3.5 days worth of data from the live news website *lepoint.fr* (4.6M clicks and 3.3M users). Sessions that did not result in clicks on recommendations were used as offline data.

## 5.2 Feature selection

The procedure for feature selection described in Section 3 was applied to the Swissinfo dataset using the CT algorithm. Due to the nature of the LAR, if there are several correlated predictors, one of them will enter the model earlier, and the others will enter much later (as their contribution would be smaller after first correlated predictor was used). However, the order in which correlated predictors will enter the model, is unknown. As we are not interested in the predictors themselves, but rather in showing the importance of metric groups (Accuracy, Coverage, Diversity, Serendipity, and Novelty), we calculated the average time for the first metric of each metric group to enter the model (Tab. 1).

**Table 1: Average index at which metrics entered the model. Second column shows the metric that typically entered the model first.**

| Metric group | First to enter | Avg. first entry±std |
|---|---|---|
| Diversity | Personalization | 2.53±0.65 |
| Serendipity | Serendipity [14] | 2.71±0.58 |
| Accuracy | Markedness | 2.82±0.76 |
| Coverage | Shannon's Entropy | 5.94±0.80 |
| Novelty | Novelty [24] | 10.27±2.77 |

Metrics from the Serendipity, Accuracy and Diversity groups are usually the first three to enter the model. This is a strong indicator that these three groups relate to different parts of performance metric and are all important for predicting it.

We also noticed that if we removed the Diversity or Serendipity predictors, then Coverage metrics showed a low average first entry time. This indicates that two out of three groups from Diversity, Serendipity and Coverage might be enough. For Accuracy metrics, it did not matter which predictor was used, and Markedness was easily replaceable by precision or any of the other Accuracy metrics. A very strong correlation was seen to exist between different Accuracy group metrics.

The results obtained with the other two algorithms were similar to those above. In all of the results below, when we say that we have used a predictor from a certain metric group, we mean that we used the predictor that was first to enter in the LAR model from this metric group.

## 5.3 Regression model performance

We used the approach described in Section 2 to generate data points for the regression model. We divided the time interval into parts of 30%, 50% and 20%. The first 30% were used for training the algorithm itself and were not used in the regression model. The recommendations made by the algorithm on the next 50% were used to train the model, and the last 20% were used to test the model's performance.

**Table 2: CTR prediction quality for different sets of features, reported values are RMSE$\times 10^4$. Lowest errors of linear regression models are in bold. Results per algorithm are shown for the Swissinfo dataset. S, Serendipity; C, Coverage; D, Diversity; A, Accuracy. Last three columns: averages for datasets.**

| Set of features | Most popular | Random | CT | Swiss-info | Yahoo | Le-Point |
|---|---|---|---|---|---|---|
| All | 2.28 | 8.54 | 1.33 | 4.05 | 2.92 | 2.37 |
| S+A+C+D | 2.08 | 2.60 | 1.40 | **2.03** | **1.79** | **2.35** |
| S+A+D | 2.24 | 2.04 | 4.17 | 2.81 | 2.50 | 4.90 |
| S+A+C | 15.3 | 1.98 | 6.47 | 7.95 | 2.62 | 4.36 |
| S+C+D | 3.81 | 2.12 | 6.07 | 4.00 | 1.84 | 6.52 |
| A+C+D | 5.32 | 5.73 | 6.06 | 5.70 | 1.87 | 3.01 |
| A | 2.98 | 5.91 | 6.23 | 5.04 | 2.90 | 5.38 |
| S | 14.4 | 1.84 | 1.55 | 5.94 | 2.92 | 9.46 |
| C | 10.2 | 5.93 | 6.22 | 7.46 | 2.64 | 7.06 |
| D | 3.23 | 4.87 | 4.59 | 4.23 | 2.49 | 6.67 |
| Const | 1.75 | 5.99 | 6.01 | 4.58 | 2.45 | 10.01 |
| All+$L_2$ | 1.97 | 2.30 | 1.17 | 1.81 | 1.67 | 2.15 |
| GP+RBF | 1.85 | 2.11 | 1.06 | 1.67 | 1.58 | 2.03 |

*CTR prediction.*

For CTR prediction (Tab. 2), the lowest average error was indeed obtained using one predictor from each of the four metric groups (we omitted novelty here and later on due to its poor results in feature selection). The error for the full set of metrics was much higher, probably due to overfitting. Diversity seemed to be very important in the first dataset, since it gave the best individual result as a predictor. Combinations of different groups including diversity also gave better results than combinations that did not. More complex models, such as penalized LR (All+$L_2$) or the Gaussian process with an RBF kernel (GP+RBF), gave even better results, however these models are more difficult to interpret and use. Note that results were consistent among datasets and that the best ones significantly outperformed the baseline model, which assumes constant CTR through time (Const).

Fig. 5 shows that the aforementioned metrics indeed have a predictive power. The shape of the curve of the performance metric over time was repeated in the predicted results, indicating that there was high probability that the model would be able to predict the behavior and approximate values of the performance metric over time.

The $p$-values associated with the regression coefficients of different metrics indicated that these predictors were significant for the model. A possible explanation for the only exception to this (Serendipity was not found to be an important predictor for Most Popular algorithm) is that all the Serendipity values for recommendations made by the Most

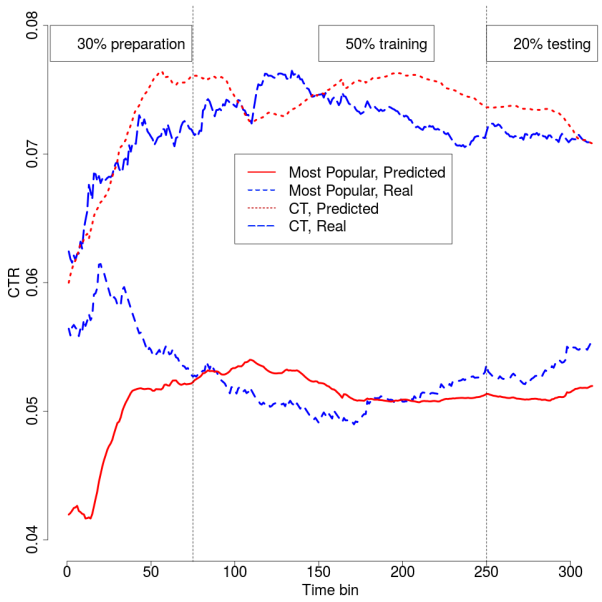


**Figure 5: Performance metric prediction for the CT and Most Popular algoithms.**

Popular algorithm were 0 or close to 0. According to definitions of Serendipity, its values are high for items not recommended by the "naive" recommender, which is precisely the Most Popular recommender.

Inspection of the coefficients revealed that the models were different for different datasets and algorithms. That is an expected indicator that a linear model made to predict the performance regardless of the algorithm, would perform worse than a set of models specifically trained for each algorithm.

*Online accuracy prediction.*

The results above were obtained assuming that the target online metric was CTR. We also trained a model to predict success, also called online accuracy (Tab. 3).

**Table 3: Online success prediction quality for different sets of features, RMSE$\times 10^4$. Lowest errors are bold. Results per algorithm are shown for Swissinfo dataset. S, Serendipity; C, Coverage; D, Diversity; A, Accuracy. Last 3 columns: averages for datasets.**

| Sets of features | Most popular | Random | CT | Swissinfo | Yahoo | LePoint |
|---|---|---|---|---|---|---|
| All | 6.25 | 7.94 | 1.98 | 5.39 | 3.90 | 3.17 |
| S+A+C+D | 13.7 | 1.81 | 1.18 | 5.57 | 2.19 | 3.10 |
| S+A+C | 2.94 | 1.39 | 2.53 | **2.28** | 3.07 | 3.24 |
| S+A+D | 22.2 | 1.89 | 1.89 | 8.66 | 2.30 | 5.15 |
| S+C+D | 15.7 | 1.97 | 9.77 | 9.16 | 2.06 | 4.09 |
| A+C+D | 15.8 | 4.55 | 6.90 | 9.10 | 2.08 | **2.15** |
| A | 17.1 | 4.98 | 6.47 | 9.53 | 2.83 | 3.95 |
| S | 11.9 | 1.64 | 13.0 | 8.87 | 2.96 | 6.23 |
| C | 30.1 | 4.99 | 22.1 | 19.1 | 3.15 | 9.19 |
| D | 11.2 | 4.35 | 4.61 | 6.72 | **1.95** | 7.19 |

Prediction errors were less consistent among datasets, but a combination of Accuracy, Coverage, and Diversity predictors obtained high results for all datasets. On the Swissinfo dataset, best predictors came from three groups that did not include Diversity; on Yahoo, Diversity is the single best predictor (possibly due to the imperfect the visits were identified); and on the LePoint dataset the best predictors did not include Serendipity. Note that the predicted results for the Random algorithm are more accurate than for the Most Popular algorithm – an expected result, as Random performance does not change much over time.

## 5.4 Self-adjusting algorithm blend

For this experiment, we ran algorithms on a live news website. We used four algorithms based on the linear combination of recommendations given by the Context Tree and Most Popular recommenders, as described in Section 4.2. We had two latent metrics, $Z_{CT}$ and $Z_{pop}$, that measured the closeness of the recommendation to that of the two algorithms. Weights for the recommendations from the Most Popular algorithm varied from 20% to 80% in steps of 20%. In each time frame and for each recommender, the updated weight increased or decreased the trade-off. We examined whether changing the algorithm weighting in the direction indicated by the gradient really did give higher CTR (Fig. 6).

In the third, fifth and seventh periods – during daytime – all coefficients suggested increasing the weight of the CT algorithm. At night, the algorithm with 20% of Most Popular was still best, but by a smaller margin, and the magnitude of the coefficients agreed with these results.

For the first two time periods the results were different, probably due to a lack of data. However, the changes suggested for increasing CTR were still consistent and correct – the regression coefficient for the algorithm using 40% of Most Popular was positive, suggesting an increase in weight, which lead to the algorithm using 60% of Most Popular, that did indeed obtain a higher CTR in this time frame.

The performance clearly changed over time, but followed a periodic pattern. This suggests that coefficients from the current time frame should provide suggestions to the same time frame next day, rather than the next chronological time frame.

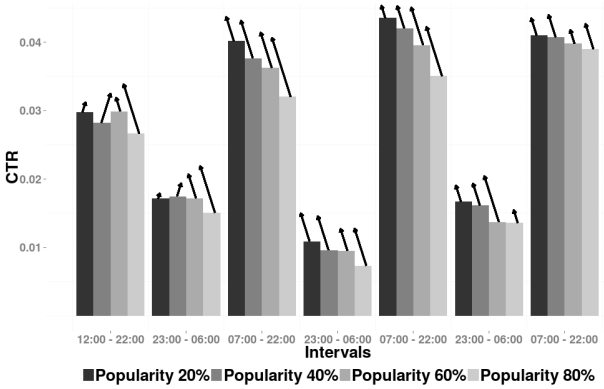


**Figure 6: CTR for four different algorithms in three different time frames. Arrow indicates the sign and magnitude of popularity regression coefficient, suggesting the direction of trade-off change.**

## 5.5 Unbiased offline evaluation

In this subsection, we discuss and apply the unbiased offline evaluation procedure [11]; this is another approach to predicting an algorithm's online performance. This procedure was developed for contextual bandit algorithms and requires the log of interaction with the world of an algorithm that recommends articles at random with equal probability. Based on this log, a simulation of online execution can be made for any other algorithm. If the original log from the random algorithm had $I$ events, then the simulation will contain approximately $\frac{I}{H}$ events, where $H$ is the number of items available for recommendation. In the Yahoo dataset, which is specifically tailored to this procedure, $H = 20$. However, in more realistic scenarios of news recommendations, such as the Swissinfo dataset, three out of 200 candidate items should be recommended, giving $H = 200^3 = 8\text{M}$. This does not produce enough events in the simulation to give significant results.

We used the Swissinfo dataset to test this algorithm. We used the output of the random algorithm to create a log and tested the CT algorithm on it. Due to the limitations described above, we were only able to compare algorithms for the task of top-1 recommendation. Even in the case of top-1 recommendation, after sampling we were left with only 266 points, compared to 55,587 points in the log. Fig. 7 (bottom part) shows the plot using the real and predicted CTR values. If we ignore a slight bias, probably due to the small number of points sampled, the shapes of the curves are quite similar, which proves the effectiveness of the unbiased offline evaluation for single-item recommendation. However, when we tried to use it for top-2 or top-3 recommendations, the number of sampled points decreased exponentially.

To overcome this problem, we applied the same technique in order to predict the third recommendation only: all the algorithms returned their third best prediction and we computed a CTR using this data. However, it is clearly visible that there was little correlation between real and predicted values (Fig. 7, top). This was caused by the fact that the presence or absence of clicks on the third item depends on the first two items in the recommendation list, and this is not taken into consideration in this approach. Statistical tests showed the significance of our findings (Student's t-test, $p$-value $< 0.05$).

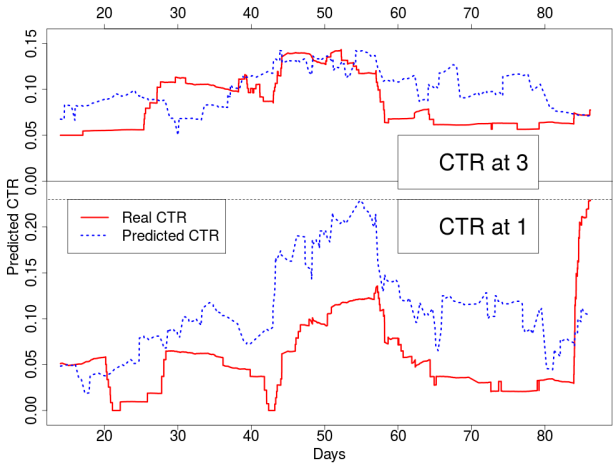


**Figure 7: CTR, real and predicted by an unbiased evaluation: for the first (bottom part), and the third (top part) items in the list of recommendations.**

Other approaches for evaluating multiple recommendations by simulating clicks on the second and later items [23, 27] are not suitable for our task as they do not account for temporal effects in the data, and their hindsight models do not take into account parameters such as coverage, etc.

# 6. RELATED WORK

The work presented here concentrates on using different metrics to predict the online performance of news recommendations. Several recommender systems have been implemented and evaluated on live news websites [3, 12, 20, 21]. A number of previous works have advocated the use of multiple metrics for these evaluations [13, 17]. Below, we describe separately works that introduced and optimized new metrics, and works that combined existing metrics into multi-objective optimization.

*Recommendations using multiple metrics.*

The need to provide more diverse and unexpected recommendations that cover the items from the 'long tail' was identified early in the history of recsys. Zhang et al. [29, 28] proposed optimizing the trade-off between the average accuracy of the recommendations and the average diversity between each pair of recommended items. They pose this as a quadratic programming problem with binary variables that they relaxed by solving in the continuous domain. However, they only measured the results via their newly introduced novelty metric and, they performed neither an online evaluation nor a user study.

Ziegler et al. [31] proposed a greedy strategy for solving a similar problem, where each item was selected in a way that minimized the average similarity to previously selected items. The subsequent user survey and, as well as the regression model built on the top of it, indicated that both diversity and accuracy contributed positively to user satisfaction. To the best of our our knowledge, this work is the only one to have built a regression model to study how performance depends on metrics.

Two serendipity metrics [7, 14] have been proposed for the domains of music and TV show recommendation. The authors compared different algorithms with respect to a newly introduced metric, without any attempt to draw a relationship between the desired performance metric and the serendipity metric.

Vargas et al. [24] proposed multiple probabilistic definitions of novelty and diversity metrics that incorporate certain previous definitions. They showed how probabilities can be used as building blocks for metric definitions and compared several state-of-the-art algorithms. Although the authors clarified the reasoning behind the probabilistic definitions they used, they proposed no algorithm to optimize towards any particular metric, and they drew no relationships with the performance metrics.

Zhou et al. [30] proposed the concepts of "personalization" and "surprisal", and they used a heat diffusion model on a bipartite graph representing links between items and users in order to optimize a linear combination of Accuracy and Diversity. The actual selection of tuning parameters, however, was done by hand.

*Multi-objective optimization.*

Multi-objective optimization of a list of items has been well investigated in the field of information retrieval [22, 25]. In the area of recsys, one of the first attempts at multi-objective optimization [28] used a quadratic objective function that involved a linear combination of Accuracy and Diversity. Jambor et al. [9] enhanced this idea by adding the variance of ratings to the objective function in order to promote items from the "long tail". Rodriguez et al. [18] optimized a smoothed version of the average precision and normalized discounted cumulative gain (NDCG) metrics by using gradient-based methods. Their framework optimized the trade-off between the quality of recommendations and the deviation between given recommendations and the expected ground truth, but did not consider competing objectives. Other works [1, 23] have expanded on ideas about optimizing a linear combination of metrics by using soft clustering of users. In [23], a linear combination of Variance and Accuracy for a contextual bandit algorithm was optimized and the recommendations were shared between different clusters through the similarity of clusters, expressed as a Gaussian kernel. Another line of research has been finding a Pareto-optimal frontier among multiple metrics, using genetic algorithms [16, 26].

# 7. CONCLUSION

We investigated predicting the online performance of news recommendation algorithms by a regression model using offline metrics. Our results confirmed that there is more to online performance than just offline Accuracy. Other metrics, such as Coverage or Serendipity, play important roles in predicting or optimizing online metrics such as click-through rates. The model can then be applied to trade-off curves for each algorithm constructed from offline data to select the optimal algorithm and parameters.

Regression models are best constructed for the particular user and item population; we did not find a universal formula for predicting online performance that would work for all settings. However, training a model separately from the algorithms still saves a lot of effort over blind A–B testing. Another application is to adapt parameters continuously in response to changes in user characteristics. In a setting where recommendations are obtained by mixing different algorithms, we proposed a method using latent metrics defined by the algorithms themselves, and showed that it correctly predicted the right adaptations in a live recommender system.

# 8. REFERENCES


[1] D. Agarwal, B.-C. Chen, P. Elango, and X. Wang. Click shaping to optimize multiple objectives. In *KDD*, 2011.

[2] R. Burke. Evaluating the dynamic properties of recommendation algorithms. In *RecSys*, 2010.

[3] A. S. Das, M. Datar, A. Garg, and S. Rajaram. Google news personalization: scalable online collaborative filtering. In *WWW*, 2007.

[4] B. Efron, T. Hastie, I. Johnstone, R. Tibshirani, et al. Least angle regression. *The Annals of statistics*, 32(2):407–499, 2004.

[5] F. Garcin, C. Dimitrakakis, and B. Faltings. Personalized news recommendation with context trees. In *RecSys*, 2013.

[6] F. Garcin, B. Faltings, O. Donatsch, A. Alazzawi, C. Bruttin, and A. Huber. Offline and online evaluation of news recommender systems at swissinfo. In *RecSys*, 2014.

[7] M. Ge, C. Delgado-Battenfeld, and D. Jannach. Beyond accuracy: evaluating recommender systems by coverage and serendipity. In *RecSys*, 2010.

[8] J. Herlocker, J. Konstan, L. Terveen, and J. Riedl. Evaluating collaborative filtering recommender systems. *Transactions on Information Systems*, 22(1):5–53, 2004.

[9] T. Jambor and J. Wang. Optimizing multiple objectives in collaborative filtering. In *RecSys*, 2010.

[10] N. Lathia, S. Hailes, L. Capra, and X. Amatriain. Temporal diversity in recommender systems. In *SIGIR*, 2010.

[11] L. Li, W. Chu, J. Langford, and X. Wang. Unbiased offline evaluation of contextual-bandit-based news article recommendation algorithms. In *WSDM*, 2011.

[12] J. Liu, P. Dolan, and E. R. Pedersen. Personalized news recommendation based on click behavior. In *IUI*, 2010.

[13] S. McNee, J. Riedl, and J. Konstan. Being accurate is not enough: how accuracy metrics have hurt recommender systems. In *CHI*, pages 1097–1101, 2006.

[14] T. Murakami, K. Mori, and R. Orihara. Metrics for evaluating the serendipity of recommendation lists. In *New frontiers in artificial intelligence*, pages 40–46. 2008.

[15] D. M. Powers. Evaluation: from precision, recall and f-measure to roc, informedness, markedness and correlation. In *Bioinfo Publications*, 2011.

[16] M. Ribeiro, A. Lacerda, E. de Moura, A. Veloso, and N. Ziviani. Multi-objective pareto-efficient approaches for recommender systems. *ACM Transactions on Intelligent Systems and Technology*, 9(1):1–20, 2013.

[17] F. Ricci, L. Rokach, and B. Shapira. *Introduction to recommender systems handbook*. Springer, 2011.

[18] M. Rodriguez, C. Posse, and E. Zhang. Multiple objective optimization in recommender systems. In *RecSys*, 2012.

[19] A. Said. *Evaluating the Accuracy and Utility of Recommender Systems*. PhD thesis, 2013.

[20] A. Said, A. Bellogín, J. Lin, and A. de Vries. Do recommendations matter?: news recommendation in real life. In *CSCW*, 2014.

[21] A. Said, J. Lin, A. Bellogín, and A. de Vries. A month in the life of a production news recommender system. In *CIKM-LL Workshop*, 2013.

[22] K. Svore, M. Volkovs, and C. Burges. Learning to rank with multiple objective functions. In *WWW*, 2011.

[23] H. Vanchinathan, I. Nikolic, F. De Bona, and A. Krause. Explore-exploit in top-n recommender systems via gaussian processes. In *RecSys*, 2014.

[24] S. Vargas and P. Castells. Rank and relevance in novelty and diversity metrics for recommender systems. In *RecSys*, 2011.

[25] H. Wang, A. Dong, L. Li, Y. Chang, and E. Gabrilovich. Joint relevance and freshness learning from clickthroughs for news search. In *WWW*, 2012.

[26] S. Wang, M. Gong, L. Ma, Q. Cai, and L. Jiao. Decomposition based multiobjective evolutionary algorithm for collaborative filtering recommender systems. In *Evolutionary Computation, IEEE Congress on*, pages 672–679, 2014.

[27] J. Yi, Y. Chen, J. Li, S. Sett, and T. W. Yan. Predictive model performance: Offline and online evaluations. In *KDD*, pages 1294–1302, 2013.

[28] M. Zhang and N. Hurley. Avoiding monotony: improving the diversity of recommendation lists. In *RecSys*, 2008.

[29] Y. Zhang, J. Callan, and T. Minka. Novelty and redundancy detection in adaptive filtering. In *SIGIR*, 2002.

[30] T. Zhou, Z. Kuscsik, J.-G. Liu, M. Medo, J. Wakeling, and Y.-C. Zhang. Solving the apparent diversity-accuracy dilemma of recommender systems. *Proceedings of NAS*, 107(10):4511–4515, 2010.

[31] C.-N. Ziegler, S. McNee, J. Konstan, and G. Lausen. Improving recommendation lists through topic diversification. In *WWW*, 2005.