A Live Comparison of Methods for Personalized Article Recommendation at Forbes.com

Evan Kirshenbaum, George Forman, Michael Dugan

HP Laboratories
HPL-2012-95R1

Keyword(s):
personalization; recommender systems; collaborative filtering; content analysis; live user trial

Abstract:
We present the results of a multi-phase study to optimize strategies for generating personalized article recommendations at the Forbes.com web site. In the first phase we compared the performance of a variety of recommendation methods on historical data. In the second phase we deployed a live system at Forbes.com for five months on a sample of 82,000 users, each randomly assigned to one of 20 methods. We analyze the live results both in terms of click-through rate (CTR) and user session lengths. The method with the best CTR was a hybrid of collaborative-filtering and a content-based method that leverages Wikipedia-based concept features, post-processed by a novel Bayesian remapping technique that we introduce. It both statistically significantly beat decayed popularity and increased CTR by 37%.

External Posting Date: July 6, 2012 [Fulltext] Approved for External Publication
Internal Posting Date: July 6, 2012 [Fulltext]

Additional Publication Information: To be published in Proceedings of the 23rd European Conference on Machine Learning and the 15th European Conference on Principles of Data Mining and Knowledge Discovery, ECML/PKDD 2012, Bristol, UK, September 24--28, 2012; Peter Flach, Tijl De Bie, and Nello Cristianini (editors); Lecture Notes in Computer Science, Springer 2012.

A Live Comparison of Methods for Personalized Article Recommendation at Forbes.com

Evan Kirshenbaum[1], George Forman[1], and Michael Dugan[2]

[1] HP Labs, Palo Alto, CA, USA
[2] Forbes Media, New York, NY, USA

Abstract. We present the results of a multi-phase study to optimize strategies for generating personalized article recommendations at the Forbes.com web site. In the first phase we compared the performance of a variety of recommendation methods on historical data. In the second phase we deployed a live system at Forbes.com for five months on a sample of 82,000 users, each randomly assigned to one of 20 methods. We analyze the live results both in terms of click-through rate (CTR) and user session lengths. The method with the best CTR was a hybrid of collaborative-filtering and a content-based method that leverages Wikipedia-based concept features, post-processed by a novel Bayesian remapping technique that we introduce. It both statistically significantly beat decayed popularity and increased CTR by 37%.

Keywords: personalization, recommender systems, collaborative filtering, content analysis, live user trial

1 Introduction

We performed an extensive study on generating personalized recommendations of articles on the Forbes.com web site. Of the many algorithms available, which is the best to deploy in practice? While each research paper on the topic forwards its own opinion, the answer is certainly that it depends on the specific situation. A study done on movie recommendation might draw different conclusions than one conducted on news articles, which have a much shorter half-life and suggest a stronger recency factor in scoring. Even a study conducted specifically on news recommendation may draw different conclusions than one specifically targeting a website like Forbes.com, which includes magazine articles and other long-lived content, such as how-to articles and profiles of top colleges and business people. Even a typical questionnaire-style study with a few volunteer Forbes users is unlikely to generalize well to real use by live users. In short, there is no substitute for trying a variety of methods *in situ* for selecting the best method(s) for one's situation. That said, only a small number of methods can be tested in a live trial, so they should be from among the most likely to succeed.

We conducted our study in two phases. In the first phase, we used a historical dataset of 8.5 million article URL clicks by 1 million unique de-identified users in order to quickly test a wide variety of recommendation methods, including

variants and parameterizations. From this study we determined a short list to evaluate in live testing in the second phase. Evaluation with historical datasets is reproducible and convenient to test many methods, but it has a variety of shortcomings and may not generalize to real-world performance [30, 8, 28]. If an algorithm gives top scores to articles not in the user's history, they may be more desirable to the user than the recorded clicks...or less desirable—only live testing can distinguish. In the worst case, the historical data represents less what users actually like than which links were available or promoted on the home page at the time the user visited, which can have a very strong effect on popularity [20].

In the second phase—the primary focus of this paper—we conducted a five-month live trial on the Forbes.com web site involving 2.1 million URL clicks by a sample of 82,000 de-identified users, each assigned randomly to one of twenty competing methods. Our goals and constraints included the following: It must deal with only de-identified user numbers, not requiring user demographics or any personal profile information. It should be scalable to 100,000 users, 100,000 articles, and dozens of competing methods running simultaneously with sub-second latency for months. The study was made viable by minimizing the changes needed by Forbes. It required only minor HTML template modifications to direct users' browsers to an HP Labs server to (1) notify the server of article page view and (2) fill an HTML iframe on the home page with links to five suggested articles. Each suggested URL included a unique hash parameter in order to allow us to measure the click-through rate (CTR) of each method.

The winning method involves a hybrid of item-item collaborative filtering and a content-based TF·IDF method, including Wikipedia-based features and a novel Bayesian score remapping technique that takes into account the models of other users that read—or were inferred to have chosen not to read—the article. We also evaluated pure versions of each method as well as various lesions, in order to determine the usefulness of the different aspects. Because our goal was to seek the best methods rather than to restrict ourselves to comparing only variants of one technique, we have good confidence that the best method we found is actually quite strong for our situation. Live trials of this kind and scale are relatively scarce in the literature. This work demonstrates a successful pattern for live trial research for researchers who do not themselves work for a major content company with a large user population.

Section 2 describes the recommender algorithms we considered in both Phase I and Phase II, including novel techniques for score remapping and Wikipedia-based concept features. Section 3 describes the live trial, our experiment protocol, and the results, including lesion variants to determine how much certain features of the best hybrid are contributing. Section 4 discusses the findings and lists some of the lessons learned. Section 5 discusses related work, and Section 6 concludes and offers future directions.

2 Recommendation Methods

In the first phase of the experiment, we analyzed a historical snapshot, provided by Forbes Media, of 8.5 million de-identified user visits to the Forbes.com website in order to determine promising candidate recommendation methods for the live trial in the second phase. The snapshot was split into a training set consisting of 8.2 million visits to articles published from May, 2010, through November, 2010, and a testing set consisting of 285,000 visits to articles published in December, 2010, by 117,000 unique users. As there was no possibility of actually making recommendations and due to other constraints imposed by the data set—most notably that the visits were not timestamped—the score given each candidate recommendation method was based on a determination, for each visit, as to whether the article visited would have been one of the top-scoring articles published within five days of the publication date of the visited article (reasoning that the "relevancy window" for an article is approximately five days and so the visit was likely within five days after publication and the competing articles were published fewer than five days before the visit).

In the first phase we ran approximately 15,000 trials, each testing a particular parameterization of a particular scoring method. Among the content-based methods, we tested variants of Naïve Bayes and TF·IDF, each with a variety of feature set combinations and other parameters, such as variants on how exactly Inverse Document Frequency is computed. Among the feature sets tested were unigram words-only and words+bigrams from the text of the article, from its title, and/or from its URL. We also included four classes of features generated by two Wikipedia-based *concept extractors* developed previously, the best of which we describe briefly in Sect. 2.2. Among collaborative filtering methods [30], we tested Item-Item collaborative filtering and User-User collaborative filtering, parameterized by several similarity metrics: Cosine, Jaccard, Euclidean, Pearson correlation, and conditional probability in each direction. We also tested the one-class collaborative filtering methods weighted Alternating Least Squares (ALS) and sampled ALS ensembles, whose regularized matrix factorizations seek to learn latent factors across the entire population [25]. As our control method, we investigated using article popularity. We also investigated linear- and non-linear combinations of article popularity with various scoring methods, as well as a novel score remapping technique, which we discuss next.

2.1 Bayesian Score Remapping Technique

For this experiment, we developed a new technique, based on Bayes' Rule, that adjusts a score from any underlying scoring method by estimating the likelihood that a user with that particular score would actually be interested enough to visit the article. In this way, we are able to distinguish between articles that are broadly popular (i.e., that empirically appeal to users whose underlying scores are relatively low) and those that are narrowly popular (i.e., that only appeal to users with high underlying scores), allowing us to reorder our estimates of a user's rank interest from what the underlying method would recommend.

To accomplish this, for each article we keep track of two score distributions, modeled as normal distributions. The first distribution contains the scores for this article of all users who have read it, based on the current underlying model for those users. This allows us to compute the conditional probability that a user who was interested to read the article has a given score for it. The second distribution similarly contains scores for the article, but in this case, the scores are for users who did not read the article. Rather than take the scores of all such users, we use the scores of users who were active on the site shortly after the article appeared (the article's "relevancy window") but who did not view the article. These users are taken to be ones who would likely have read the article had they been interested and so are inferred to not have been interested. From this second distribution, we can compute the conditional probability that a user who was not interested in the article has a given score for it. Note that in a live system, users' scores for articles will change frequently and when users view the article they move from the *non-interested-* to the *interested*-distribution, so these distributions need to be dynamically maintained.

By Bayes' Rule,

$$\frac{\Pr[\,A \mid B\,]}{\Pr[\,\overline{A} \mid B\,]} = \frac{\Pr[\,B \mid A\,] \cdot \Pr[\,A\,]}{\Pr[\,B \mid \overline{A}\,] \cdot \Pr[\,\overline{A}\,]} \quad . \tag{1}$$

In our context, for a particular underlying score s, the conditional likelihood ratio

$$R = \frac{\Pr[\,\text{interesting} \mid s\,]}{\Pr[\,\text{not interesting} \mid s\,]} \tag{2}$$

$$= \frac{\Pr[\,s \mid \text{interesting}\,] \cdot \Pr[\,\text{interesting}\,]}{\Pr[\,s \mid \text{not interesting}\,] \cdot \Pr[\,\text{not interesting}\,]} \tag{3}$$

$$= \frac{\Pr[\,s \mid \text{interesting}\,]}{\Pr[\,s \mid \text{not interesting}\,]} \cdot \frac{\Pr[\,\text{interesting}\,]}{1 - \Pr[\,\text{interesting}\,]} \quad , \tag{4}$$

which can be computed directly with our stored distributions. Since

$$\Pr[\,\text{interesting} \mid s\,] = 1 - \Pr[\,\text{not interesting} \mid s\,] \quad , \tag{5}$$

$$\Pr[\,\text{interesting} \mid s\,] = \frac{R}{R+1} \quad , \tag{6}$$

by use of equation (2) again. Equation (6) is used as an adjusted score indicating the likelihood that a user with a particular underlying score s will be interested in reading the article. This works for any underlying monotonic scoring method.

Some care must be taken when one or both of the distributions for an article contain too few values for the conditional probabilities to be statistically meaningful. In this case, we consider that we do not yet have enough information about the article and do not recommend it.

2.2 Wikiconcept Features

For a prior project we had developed a Wikipedia-based *concept extractor* [16] that takes unstructured text as input and constructs an analysis that includes a set of *concepts*, identified by Wikipedia articles, each associated with a score indicative of the degree to which the text is "about" that concept, with higher scores indicating that the concept is central to the text and lower scores indicating that, although the concept is mentioned in the text, it is relatively incidental.

The details of the concept extractor are beyond the scope of this paper, but roughly, the text is broken up into sentences (or sentence-equivalents) and scanned for the presence of 1–5 word *anchor phrases*, taken from the text displayed on intra-Wikipedia hyperlinks and implicating (often ambiguously) concepts associated with the Wikipedia articles the hyperlinks point to. For example, the anchor phrase "Clinton" is associated with "Bill Clinton" and "Hillary Clinton", along with "George Clinton", "Henry Clinton", "DeWitt Clinton", "Clinton, Iowa", and "Clinton County, NY", each with a prior degree of likelihood based on the number of times the phrase was used within Wikipedia to link to the concept's article.

The evidence from the detected anchor phrases is passed to an iterative consensus algorithm that determines, based on the evidence and on the conditional likelihood that pairs of Wikipedia articles will both be link targets within a Wikipedia article, the most likely concept referent for each anchor phrase (if any is deemed sufficiently likely). Each concept the extractor knows about is also associated with a set of *categories*, and based on the coocurrence of concepts associated with different categories, one or two (occasionally more) categories are chosen to describe the context of the particular concept. Categories are also inferred directly using a support vector machine trained with non-Wikipedia articles from the Open Directory Project (www.dmoz.org).

The tables used to drive the extractor are generated automatically from a Wikipedia snapshot obtained periodically from Freebase.com. The system used for the live trial includes 6.7 million normalized anchor phrases that impute 3.3 million concepts. The category set includes 913 categories in a hand-crafted hierarchy.

The concept extractor outputs four classes of features for consideration by the content-based methods: (1) detected anchor phrases, annotated by occurrence count, (2) extracted concepts, annotated by centrality score and confidence of detection, (3) recognized categories, associated with categorizer scores, and (4) identified concept/category pairs, annotated by concept centrality and confidence.

2.3 Phase II Methods

Based on the Phase I trials, the details of which must be omitted for space, the clear winner was Bayesian-adjusted TF·IDF, with no IDF component (so just TF), with a logarithmic transformation on the TF counts, and L_2 normalization on the length of the feature vector. Its best-performing feature set included

anchor phrases and concepts from the concept extractor (see Sect. 2.2), words from the article title, and words and bigrams from the article URL, but not, interestingly, words or bigrams from the article body. We therefore chose to use TF·IDF, variously parameterized, for our content-based methods. As we also wanted to investigate collaborative filtering in the live trial, we chose the most competitive method: Item-Item collaborative filtering where a user's score for an unread article U is the conditional probability that other users have read U, given that they also read an article R that the user has read, averaged over all articles R the user has read so far.

Based on anticipated traffic volume and experiment duration, we estimated that we could test 20 methods in parallel in Phase II, with each newly-observed user randomly assigned to a recommendation method in a balanced manner, and expect to be able to statistically significantly distinguish between better-performing and worse-performing methods. We considered it important to run all of the methods at the same time so that we could be confident that differences we found were due to the methods themselves and not due to changes in the set of candidate articles (e.g., that one recommendation method had a popular and easily recommendable candidate article not available to another method, were we to use sequential A-B testing).

In Phase II, our recommendation methods consisted of a *scoring function*, which produced a numeric score for each of a set of candidate articles, a set of *filters*, which constrained the set of candidate articles, and a *selection method*, which seclected the articles to recommend based on the computed scores and possibly other information associated with the articles. Unless otherwise mentioned, all Phase II methods included filters that removed from consideration any article that the user had already read or for which the user had at least twice selected a recommended article further down in a recommendation list. Except for Bayesian-adjusted methods, the selection method selected articles with the highest associated scores. For Bayesian-adjusted methods, unless specified, the selection method selected the most recently-published articles from among the 25 highest (adjusted)-scoring articles, with all articles published in the last 48 hours considered to be equally recent. This is an attempt to capture the notion of recency, which is built in for methods that mixed with decayed popularity.

The Phase II methods are listed in Table 1. We first describe the methods intended as baselines.

Baseline Methods Five of the twenty methods were chosen as representing state of practice. Method 1 prefers articles that have been visited the most times, with an exponential decay. As with all of the Phase II methods that involved decayed popularity, the smooth decay parameter was chosen such that a visit is worth 10% of one 24 hours later. This decay value was empirically chosen as substantially optimal based on an observational period prior to the start of Phase II, during which we observed user visits but did not make recommendations.

Method 2 is like Method 1, except that popular articles are recommended to a user even if the user has previously read the article or if the user has selected

Table 1. Phase II methods

Baseline Methods

1. Decayed popularity
2. Unpersonalized decayed popularity
3. Raw (undecayed) popularity
4. Unadjusted TF·IDF("bag of words" features, no Wikiconcepts features)
5. Unadjusted Item-Item collaborative filtering

Experimental Methods

6. Popularity-adjusted TF
7. Bayesian-adjusted TF
8. 50%-popularity-adjusted Item-Item
9. 10%-popularity-adjusted Item-Item
10. Popularity-adjusted TF/CF hybrid
11. Bayesian-adjusted TF/CF hybrid

Lesion Methods

12. Popularity-adjusted TF (no concepts)
13. Bayesian-adjusted TF (no concepts)
14. Popularity-adjusted TF·IDF (no Wiki features)
15. Bayesian-adjusted TF·IDF (no Wiki features)
16. Unadjusted TF
17. Bayesian-adjusted TF (no recency focus)
18. Popularity-adjusted TF (no negative interest filter)
19. Bayesian-adjusted TF (no negative interest filter)
20. Bayesian-adjusted TF (using CDF)

articles below that article in prior recommendation lists—this represents the commonplace, unpersonalized *most popular* lists at many sites. In Method 3, the most popular articles are recommended to a user with no popularity decay.

For the other two baseline methods we chose one content-based method and one collaborative filtering method. The content-based method, Method 4, is unadjusted TF·IDF (including the typical IDF component and logarithmic TF transform), with L_2 normalization over the usual "bag of words" features taken from the article's body, title, and URL. The collaborative filtering method, Method 5, is unadjusted Item-Item collaborative filtering, as described above.

Experimental Methods The next six methods are the ones that we expected to be serious contenders. Two of them are content-based methods using L_2-normalized, logarithmically-transformed TF over extracted concepts, detected anchor phrases, title words, and URL words and bigrams. The resulting score is, in Method 6, averaged evenly with the score from Method 1 (decayed popularity), while in Method 7, the Bayesian adjustment described in Sect. 2.1 is applied. Even though the Bayesian adjustment outperformed mixing with popularity in the static Phase I trials, we wanted to leave ourselves open to the possibility that in live trials we might get a different result, and so included popularity-adjusted variants to the Bayesian-adjusted methods.

Two experimental methods used a weighted average of Item-Item collaborative filtering with decayed popularity. Method 8 weights them evenly, while Method 9 gives just 10% weight to the popularity component.

In the final two experimental methods, we use both content-based (TF) and collaborative filtering (CF). In Method 10, the scores from TF (as in Method 6),

Item-Item collaborative filtering, and decayed popularity are averaged evenly. In Method 11, the scores from TF and Item-Item collaborative filtering are averaged evenly and Bayesian adjustment is applied to the resulting score.

Lesion Methods Finally, we included nine methods that investigate leaving out or otherwise altering some aspect of one of the experimental methods in order to determine whether that aspect is important. In Methods 12 and 13, we investigate leaving out concepts as features from Methods 6 and 7. While the concept extractor is quite efficient, there is a runtime and software-complexity cost for including the algorithm, and so if it turned out that concepts were needless, omitting them would be an appreciable simplification in feature extraction.

In Methods 14 and 15, we further leave out anchor phrases as features. While the time required to detect these phrases is negligible, if neither concepts nor anchor phrases is required, then there is no need to expend the effort to obtain Wikipedia snapshots and build the required tables. To give these methods the best chance, we chose the best-performing parameterization from Phase I over TF·IDF runs that did not use concept extractor features. The IDF component is included, and the features are title words, URL words and bigrams, and body words and bigrams.

Method 16 is chosen to validate whether there is an on-line benefit for the popularity or Bayesian adjustments in Methods 6 and 7 by running the TF algorithm, with the same parameterization, unadjusted.

In Method 17, we investigate the impact of the recency-biased selection method by running Bayesian-adjusted TF but selecting the top-scoring articles regardless of age.

In Methods 18 and 19 we investigate the benefit of including the "negative interest filter." Recall that in other methods (with the exception of Method 2) if the user accepted two recommendations that were listed below a given recommended article, we inferred that the user was not interested in that article and refrained from recommending it in the future.

Finally, in Method 20, we make an adjustment to Method 7. In all of the other Bayesian-adjusted methods, when we figure the conditional $\Pr[s|(\text{not})\text{interested}]$, we use a *probability density function* (PDF) to determine the probability of getting precisely that score. In Method 20, by contrast, we use a *cumulative density function* (CDF) to determine the probability of getting a score at least that high.

3 Live Trial

In Phase II we received information about user visits in real time and had the opportunity to recommend articles that the user might be interested in and learn when our recommendations were taken. We were interested in answering two questions:

Question 1: Do any of the experimental methods described in Sect. 2.3 represent a significant improvement over the baseline methods in terms of the

click-through rate (CTR)? That is, are users significantly more likely to accept recommendations made by certain methods than others?

Question 2: Do good recommendations increase the amount of time a user spends on the site? That is, do users who take recommendations have longer session lengths than users who do not, and is there a significant difference in user behavior after the user begins to take recommendations?

Click-through rate was chosen over metrics such as accuracy, precision, recall, or F-measure because in a live trial ground truth is unavailable for recommendations not taken and because users' preferences may change over time, so a recommendation skipped and later taken may still have been a mistake.

3.1 Experiment Protocol

For the live trial, Forbes Media identified a subset of visitors to the Forbes.com website whom they could stabily identify and made two changes to their served pages. First, whenever one of these users visited a page representing an article or slide deck, JavaScript code within the page made an asynchronous (Ajax) call to an HP Labs server passing in the URL of the visited page and an opaque numeric identifier representing the user. (The call requested an image, which was not displayed but which directed users worried about privacy concerns to a page explaining the experiment.) Second, whenever one of these users visited the Forbes.com homepage, the HP Labs server was requested to populate an HTML *iframe* with links to five recommended articles for the user identified in the request.

When a visit notification was received, the server first determined whether the associated user was already known and if not, selected a recommendation method to be used for them. Next, if the associated URL was not tied to a known article, the server requested the web page from the Forbes.com server and used it to extract features, including calling the concept extractor described in Sect. 2.2. If an analysis of the HTML code for the retrieved page indicated that it was part of a multi-page article, which might include multiple pages of images with caption text, the server determined the entire "constellation" of URLs associated with the article and based the concept extraction on the text from all of the HTML pages.

The server then informed all of the recommendation methods about the visit, allowing them to update their models. Note that even though only one method would be used to make recommendations for a given user, other methods might also want to make use of the information. For example, in order to ensure that the score distributions used by the Bayesian adjustment were large enough, the positive distributions made use of visits from users assigned to any method, not merely those assigned to Bayesian-adjusted methods. Similarly, collaborative filtering methods made use of all visits, not merely visits by users assigned to collaborative filtering methods. It should be noted that in order to not impact the Forbes.com users, the actual visit notification was logged and replied to

immediately, with the described processing taking place as soon as possible after the fact.

When a recommendation iframe request was received, the server identified the recommendation method associated with the user (selecting one if necessary and associating it with the user for the future) and asked that method for five ordered articles to recommend. As noted above in Sect. 2.3, each recommendation method consisted of a scoring function, a set of filters, and a selection method. First, the current set of recommendable articles was passed through the filters, which typically did things like removing candidate articles that the user had already read. Next, the remaining candidates were passed to the scoring function, which, often using a constructed model for the user, computed a numeric score for each of them (possibly indicating for some of them that no score can be computed). Finally the selection method took the candidates and associated scores and picked a ranked list of five articles to recommend.

The list was then formatted as an HTML iframe containing links to the recommended articles, identified by their titles. The URLs of the articles were modified by the addition of an HTTP query parameter so that if the user clicked on one of them, the resulting visit notification received by the server would be seen to be an accepted recommendation, with the particular recommendation set and the rank within the list identified. The recommendation set itself, along with the user identifier, recommendation method, and timestamp, were stored on the server for later analysis.

The set of recommendable articles was taken to be those mentioned on a set of approximately 85 RSS feeds identified by Forbes Media, filtered to exclude those pages for which we would not receive a visit notification if a recommendation was taken. As with visit notifications, if the server did not already know about an article, it was fetched from Forbes.com, its page constellation determined, and its features extracted.

The system began receiving visit notifications on Sept. 29, 2011, and began making recommendations on Nov. 29, 2011. Phase II ended on Mar. 11, 2012, after making recommendations for 103 days, with only minor outages. Excluding data associated with users involved in running the experiment and those whose behavior made them appear to be robots, we received 2.1 million visit notifications from 82,412 unique users (14,818 of whom were responsible for a single visit each). We made 388,392 recommendations, of which 3,118 (0.80%) were taken overall. These absolute values reflect that the recommendations were plain looking and required some scrolling to notice the iframe; the comparison will focus on relative differences among the methods.

3.2 Comparison of Recommenders

As most non-trivial recommendation methods require a model of the user, and as building such a model typically requires observing user behavior, we are primarily concerned with the relative performance of different methods for recommendations given when a minimum number of observations have been made. For purposes of evaluation, we set our threshold at having observed the user visit

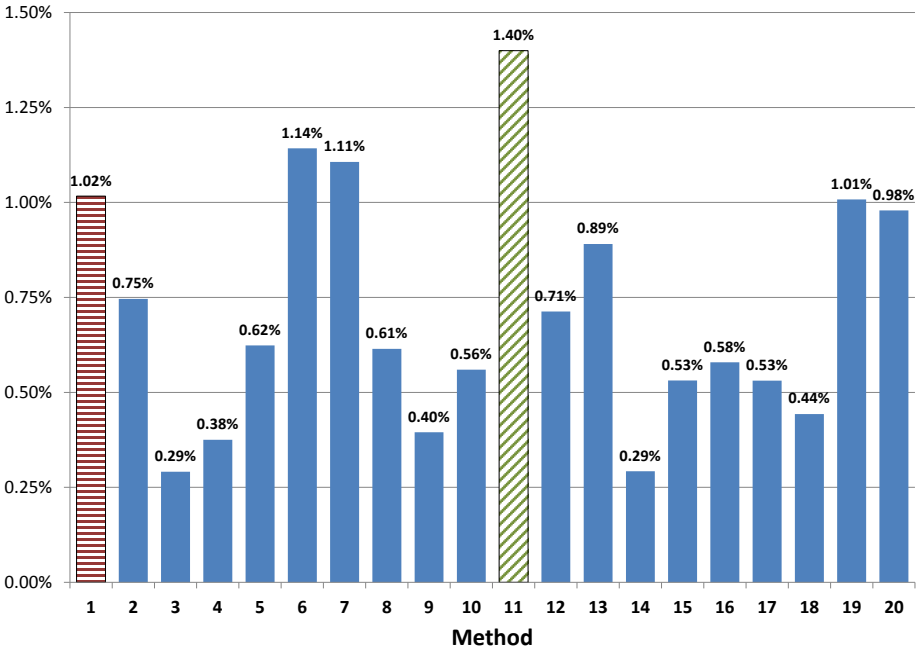


Fig. 1. Click-through rates by method for recommendations given with at least five articles viewed. See Sect. 2.3 and Table 1 for method descriptions.

at least five distinct articles. With this threshold, the system provided 311,015 recommendation lists, with a median of 13,789 recommendations given per recommendation method (max $= 25,022$, min $= 8,416$). Of these, 2,060 resulted in a recommendation being taken for an overall click-through rate of 0.66%. The click-through rates for all methods can be seen in Fig. 3.2. (Refer to Table 1 and Sect. 2.3 for descriptions of the various recommendation methods.)

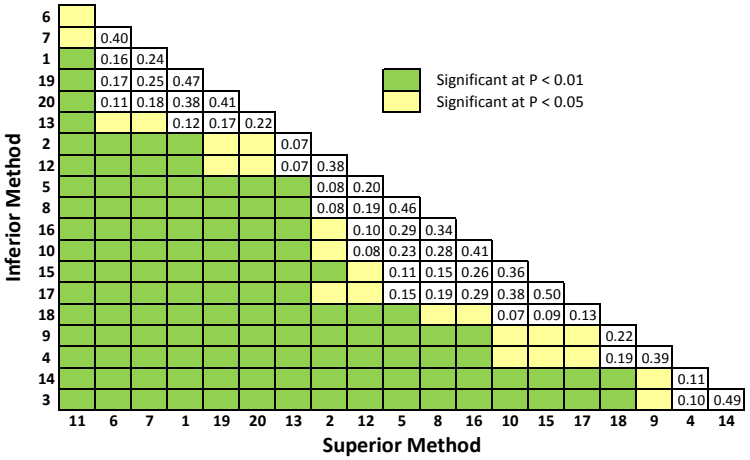
Method 1, decayed popularity, represented the best of the baseline methods and was significantly ($p < 0.01$) better than any other baseline method, achieving a click-through rate of 1.02%. Three of the experimental methods exceeded this level—Method 6 (popularity-adjusted TF, at 1.14%), Method 7 (Bayesian-adjusted TF, at 1.11%), and Method 11 (Bayesian-adjusted TF/CF hybrid, at 1.40%)—but the first two comparisons are not significant ($p = 0.16$ and 0.24, respectively).

Method 11, the Bayesian-adjusted hybrid of content-based (TF) and collaborative filtering (CF) methods, however, was significantly better than decayed popularity and, indeed, all other methods ($p < 0.05$ vs. Methods 6 and 7 and $p < 0.01$ vs. all other methods). It achieved a click-through rate of 1.40%, which is a 37% improvement over decayed popularity and a 23% improvement over the next best method.

A full table of the significance of pairwise comparisons can be found in Table 2. Of particular interest is the relatively poor performance of using simple

undecayed popularity (Method 3), TF·IDF with plain "bag-of-words" features (Method 4), and simple Item-Item collaborative filtering (Method 5)—although between these "pure" methods, collaborative filtering was significantly ($p < 0.01$) better than TF·IDF.

Table 2. Significance levels of click-through rate comparisons by method for recommendations given with at least five articles viewed, ordered by click-through rate of method. See Sect. 2.3 and Table 1 for method descriptions.



3.3 Temporal Stability

A reviewer suggested that the winning method's success might be due to excellent performance during only a portion of the experiment timeline. While it was not possible to perform more experiments at this point, we have analyzed the collected data to determine whether we can say that this was, in fact, the case. To do so, we divided the collected data equally into two time periods or three time periods. Naturally, the smaller amount of data available in each sub-period makes it more difficult to attain statistical significance.

We found that Method 11 was still the best method in each of the two halves, though not significantly better than the runner-up methods ($P = 0.14$ and $P = 0.36$ respectively). When dividing into three time periods, it was significantly ($P < 0.05$) the best in the middle third, and it was not significantly worse ($P = 0.07$ and $P = 0.13$ respectively) than the best methods of the first and last thirds. Comparing Method 11 against itself in different sub-periods, we found that it performed significantly better ($P < 0.01$) in the middle third than in the rest of the experiment or in the other thirds, and significantly better ($P < 0.05$) in the first half than in the second half. Thus, there is some justification for believing that Method 11 found some days easier than others, and future experiments may want to take this concern into account. Note that the runner-

up method, Method 6 did not show any significant differences between any of the sub-periods.

Lesion Comparisons Turning to the lesion methods described in Sect. 2.3, nearly all of them performed significantly worse than the experimental methods they were compared to.

The content-based methods that did not include Wikipedia concepts as features (Methods 12 and 13) performed significantly worse than the corresponding methods that did ($p < 0.01$ for popularity adjustment and $p < 0.05$ for Bayesian adjustment). The content-based methods that further did not include anchor phrases as features (Methods 14 and 15) performed significantly worse ($p < 0.01$) than either those with the full complement of features or those lacking concept features.

The adjustments themselves were helpful, as unadjusted TF (Method 16) performed significantly worse ($p < 0.01$) than either popularity-adjusted TF (Method 6) or Bayesian-adjusted TF (Method 7).

The recency focus for Bayesian-adjusted methods was useful, as the method that used it (Method 7) performed significantly better ($p < 0.01$) than the one that did not (Method 17).

The negative interest filter, in which articles are suppressed that the user has bypassed twice in selecting a recommendation lower on the list, was a mixed bag. With popularity adjustment, the method that used it (Method 6) performed significantly better ($p < 0.01$) than the method that did not (Method 18). But with Bayesian adjustment, while the method that used it (Method 7) outperformed the method that did not (Method 19), the difference was not significant ($p = 0.25$).

Finally, concerning the Bayesian adjustment, using PDFs (Method 7) was insignificantly better ($p = 0.18$) than using CDFs (Method 20).

Performance at Other History Levels On the user's first visit, unsurprisingly, no method significantly outperformed decayed popularity (Method 1).

With a small amount of history (1–4 articles), Methods 6 and 18 (popularity-adjusted TF, with or without the negative interest filter) had the best click-through rates (2.70% and 2.91%, respectively), but they were only suggestively better than the popularity-adjusted hybrid (Method 10, $p = 0.059$) and decayed popularity (Method 1, $p = 0.075$), and most methods were not significantly different from decayed popularity. An interesting exception is Method 11, which was the clear winner with at least five articles of history, and which performed worse ($p < 0.01$) than decayed popularity with 1–4 articles.

With at least ten articles of history Method 11 was still the winner, with a CTR of 1.41%, but the difference between it and decayed popularity (Method 1) was no longer significant ($p = 0.125$). It was better than Method 7 (Bayesian-adjusted TF) at the $p < 0.05$ level and better than all other methods at the $p < 0.01$ level. Decayed popularity was better than all remaining methods except for Bayesian-adjusted TF at the $p < 0.01$ level.

The same findings apply with at least 100 articles of history. Method 1 (decayed popularity) performed better than Method 11 (the Bayesian-adjusted hybrid), but the difference was not significant ($p = 0.142$). It was significantly better ($p < 0.05$) than Method 7 (Bayesian-adjusted TF), which was not significantly worse ($p = 0.144$) than Method 11.

3.4 Session Lengths

The second question we wanted to address was whether good recommendations translated into more page views. To measure this, we put all of the users into the same group, ignoring differences in their assigned recommendation methods, reasoning that regardless of method, a recommendation can be considered good if the user clicks on it.

To measure the number of page views, we broke each user's history into a number of *sessions*, with a session considered to have ended if more than sixty minutes elapsed between successive clicks. For each session, we noted whether at least one of the clicks was the result of taking a recommendation.

We found that sessions in which a recommendation was taken averaged 7.58 ± 0.35 clicks ($n = 2,750$), while sessions in which a recommendation was not taken averaged 3.56 ± 0.14 clicks ($n = 319,024$), for an increase of 4.02 clicks per session on average. Many of the sessions were very short, as users often visited the site by following a link, read an article, and left, never having been shown a recommendation. If such "trivial" sessions are excluded by setting a minimum session length of 3 clicks, sessions in which a recommendation was taken averaged 10.72 ± 0.51 clicks ($n = 1,822$), while sessions in which no recommendation was taken averaged 8.08 ± 0.42 ($n = 106,221$), for an increase of 2.30 clicks on average.

A reasonable question to ask is whether this difference is simply due to users who take recommendations being more likely to have longer sessions. This tendency does, indeed, exist. Sessions for users who ever took recommendations averaged 1.99 clicks longer (5.22 ± 0.05 as opposed to 3.24 ± 0.17) than sessions for those who never did. Indeed, even before the experiment proper began, when we were simply observing visits and not making recommendations, sessions for users who would eventually go on to take recommendations averaged 0.95 clicks longer (3.82 ± 0.08 as opposed to 2.87 ± 0.05). But this is unlikely to be the whole story, as when focusing on users who ever took clicks, their sessions averaged 4.45 ± 0.06 clicks before they took their first recommendation and 5.69 ± 0.08 clicks afterwards, for an increase of 1.24 clicks per session. For non-trivial sessions, the increase was greater, from 8.29 ± 0.11 clicks per session to 10.26 ± 0.15, for an average increase of 1.97 clicks per session.

All comparisons in this section are significant at the $p < 0.01$ level.

4 Discussion

The click-through rates presented in Sect. 3.2 are almost certainly underestimates. When our system was asked to make a recommendation, it had no in-

formation about articles that were displayed elsewhere on the page, and thus often recommended articles that were already among those selected by Forbes editors to appear on their page. When this happened, it is quite likely that users who agreed with our recommendations would choose the editor's link, which was larger and accompanied by a picture, unlike our title-only recommendations. Our system conservatively treats this situation as a miss.

We also had an interesting "cold-start" problem. We attempted to not recommend articles that the user had already seen, but we had no knowledge of what users had seen before they showed up for the first time during the experiment. So some of the recommendations, especially near the beginning of the experiment, were likely rejected not because they were uninteresting, but because the user had already read them before the experiment started.

Our experiments confirmed that sophisticated methods can (and, indeed, may be required to) beat simple popularity-based recommendation approaches. At a broad level, this collaboration demonstrates that, by partnering with a popular commercial website, it is possible to run meaningful large-scale experiments. Probably the biggest lesson we learned is that for experiments like these, upfront planning is key. This is drilled into medical researchers, but is somewhat foreign to most computer scientists. We are very grateful to Forbes Media for trusting us with their customers and their image. Being live on a commercial site meant that we had to be enormously careful that nothing that we did would cause them harm or embarrassment. Also, the scale of the experiment, particularly the amount of time that would be required to gather data to establish significance meant that we had to be prepared to turn it on and walk away for months, with only minor maintenance. This meant that we had to be confident that our code could handle the scale, but it also meant that we had to decide ahead of time what things we were going to want to compare and engineer the system so that we could test them all based on a single period of data collection. This meant that we spent more time than was normal before we began, but the delays were worthwhile.

5 Related Work

For a broad survey of recommendation methods see [1, 29]. Our finding that hybrid methods excel is consistent with a broad literature, e.g. [2, 3, 6, 21, 23, 14]. The Netflix competition [4] enabled a large scale search of methods, of which the superior methods were consistently ensembles of many models, and the work turned toward optimization of these mixtures, e.g. [14]. Further, our finding that features from Wikipedia-based concepts [16, 24] and a category hierarchy are consistent with others who have used taxonomies to attack the data sparseness, e.g. [15, 18, 22].

The vast majority of research publications evaluate methods using only historical datasets. While this may work for explicit user ratings of movies or books, passively collected datasets of user clicks on unrated news or other short-lived content may not represent user preference as much as which specific links were

available or promoted on the web site at that time. A study of Digg popularity found this to be a very strong effect [20], and we have seen some confirming anecdotes in our datasets. Thus, we believe live trials are essential in this domain. Published studies using live feedback are relatively rare and often involve only small sets of users (e.g. 44 users in [7], 50 in [5], 57 in [22], 141 in [10], 239 in [26]). These sizes may support comparisons of content-analysis methods, but they give a large disadvantage to collaborative-filtering methods that depend on learning from a large user population. Further, such studies sometimes include in-depth questionnaires of perceptions and explicit rating by volunteers in a setting that does not match the intended live environment. Two studies of live news recommendation for large user populations (>10,000 users) were reported at Google. Their live trials compared relatively few methods: Liu et al. [21] reported testing a single method against their existing baseline in an A-B comparison; Das et al. [9] compared three algorithms, and also preceded their live trial with a broader search using their historical data.

One barrier to conducting such live studies is that researchers of recommendation methods are typically not situated within media companies. Our study shows that this bridge can be crossed with relatively little complication. A novel workaround proposed recently is to hire a user base via Mechanical Turk [27, 17]. While this may be the best approach for some studies, it will always differ substantially from the intended production setting and its typical users.

A great deal of thoughtful work has been published on ways to evaluate recommendation methods, e.g. [11, 12], including aspects of accuracy, novelty, diversity, explainability, and avoiding embarrassment. Koren [19] demonstrates that although most papers focus on error measures, esp. root-mean-squared-error (RMSE) for compatibility with the Netflix competition and its many papers, methods that optimize RMSE do not correspond to those that optimize the hit rate in the top-N list of recommendations—the common use-case for production. Hurley and Zhang [13] demonstrate that some increased emphasis on diversity can improve top-N recommendation. Online studies in recommendation and advertisement (another form of recommendation) usually measure by CTR, which aligns with financial incentives and implicitly factors in accuracy, novelty, diversity, etc., according to the preferences of the distribution of users. Zheng et al. [30] empirically determined that click-throughs are not correlated entirely with the user-perceived relevance of the items. Although they took this evidence to mean that we should not use CTR when searching for methods with greater relevance, it is consistent with a variety of publications that suggest we should not focus entirely on item relevance when seeking to maximize CTR, e.g. the importance of mixing in popularity, as our leading hybrid methods do.

6 Conclusions and Future Work

From the results of this substantial real-world experiment we can see that it is possible for a recommender system to outperform recommending recently popular articles and to do so not merely statistically significantly but also meaning-

fully, as our winning method had a click-through rate that represented a 37% improvement over the best popularity-based method. On the other hand, doing so is not easy. Our winning method made use of both content-based reasoning and collaborative filtering, and it required sophisticated analysis of the article text in order to provide the Wikipedia-based concept features it needed. Moreover, our lesion studies demonstrated that all of these attributes appear to be required.

We also saw in Sect. 3.4 that providing good recommendations can be useful from a business standpoint, as users who take recommendations stay on the site longer, which likely translates to more loyal customers and potentially greater advertising revenue.

In future experiments, it would be interesting to add a feedback mechanism that would allow the users to directly indicate when they found a listed recommendation to be poor and what they felt about an article (whether or not they viewed it via a recommendation link). In the current work, we could only infer that the user found an article interesting if and only if they viewed it, but there's always the possibility that the user was "tricked" into clicking on the headline of an article they actually were not interested in. With direct feedback, the recommendation methods would be expected to be more accurate and more dynamic, and users might well find recommendations more appealing if they felt that they had a measure of control.

In the same vein, future experiments would profit from being told what other articles were referred to on pages. This would allow the system to more accurately detect lack of interest in articles or that one article is less interesting than another, and it would allow the recommender to avoid duplicating articles that are already displayed on the page.

In the current experiment, the only thing the recommenders knew about the users came from the pages they viewed on the site. In future experiments, it could be useful to add other sources of information, e.g., demographics or other behavior known by the site.

Finally, in this experiment recommendations were produced only on the site's main page. This meant that unless the user navigated back to that page rather than taking a link from the article page, the system had no chance to make use of the article being viewed to immediately display options for where to go next. Indeed, the system may never get the chance to make a recommendation to users that arrive at the site from inward-pointing links and do not use the main page. In future experiments, it would be helpful to be allowed to make recommendations on article pages as well.

Acknowledgments The authors would like to thank David Dunlop, Nick Hallas, Mark Kolich, Carol Ozaki, Shyam Rajaram, Fereydoon Safai, Craig Sayers, Martin Scholz, Jaap Suermondt, Sherry Xia, and the staff at Forbes who helped this project be a success.

References

1. G. Adomavicius and A. Tuzhilin. Toward the next generation of recommender systems. *IEEE Trans. on Knowl. and Data Eng.*, 17(6):734–749, June 2005.
2. M. Balabanović and Y. Shoham. Fab: content-based, collaborative recommendation. *Communications of the ACM*, 40(3):66–72, Mar. 1997.
3. J. Basilico and T. Hofmann. Unifying collaborative and content-based filtering. In *Proc. of 21st int'l conf. on Machine learning*, ICML '04, pages 9–, 2004.
4. R. M. Bell, J. Bennett, Y. Koren, and C. Volinsky. The million dollar programming prize. *IEEE Spectrum*, 46(5):28–33, May 2009.
5. D. Billsus and M. J. Pazzani. Adaptive news access. In P. Brusilovsky, A. Kobsa, and W. Nejdl, editors, *The Adaptive Web*, pages 550–570. Springer-Verlag, 2007.
6. R. Burke. Hybrid web recommender systems. In P. Brusilovsky, A. Kobsa, and W. Nejdl, editors, *The Adaptive Web*, pages 377–408. Springer-Verlag, 2007.
7. J. Chen, R. Nairn, L. Nelson, M. Bernstein, and E. Chi. Short and tweet: experiments on recommending content from information streams. In *CHI'10*, 2010.
8. P. Cremonesi, Y. Koren, and R. Turrin. Performance of recommender algorithms on top-n recommendation tasks. In *RecSys '10*, pages 39–46, 2010.
9. A. S. Das, M. Datar, A. Garg, and S. Rajaram. Google news personalization: scalable online collaborative filtering. In *WWW '07*, pages 271–280, 2007.
10. J. de Wit. Evaluating recommender systems: an evaluation framework to predict user satisfaction for recommender systems in an electronic programme guide context. Master's thesis, University of Twente, The Netherlands, May 2008.
11. M. Ge, C. Delgado-Battenfeld, and D. Jannach. Beyond accuracy: evaluating recommender systems by coverage and serendipity. In *RecSys '10*, page 257, 2010.
12. J. Herlocker, J. Konstan, L. Terveen, and J. Riedl. Evaluating collaborative filtering recommender systems. *ACM Trans. Inf. Syst.*, 22(1):5–53, Jan. 2004.
13. N. Hurley and M. Zhang. Novelty and diversity in top-N recommendation – analysis and evaluation. *ACM Trans. Internet Technol.*, 10(4):14:1–14:30, Mar. 2011.
14. M. Jahrer, A. Töscher, and R. Legenstein. Combining predictions for accurate recommender systems. In *KDD '10*, pages 693–702, 2010.
15. G. Katz, N. Ofek, B. Shapira, L. Rokach, and G. Shani. Using Wikipedia to boost collaborative filtering techniques. In *RecSys '11*, pages 285–288, 2011.
16. E. Kirshenbaum. A Wikipedia-based concept extractor for unstructured text. Technical report, HP Labs, In preparation.
17. A. Kittur, E. H. Chi, and B. Suh. Crowdsourcing user studies with mechanical turk. In *CHI '08*, pages 453–456, 2008.
18. N. Koenigstein, G. Dror, and Y. Koren. Yahoo! music recommendations: modeling music ratings with temporal dynamics and item taxonomy. In *RecSys '11*, 2011.
19. Y. Koren. Factorization meets the neighborhood: a multifaceted collaborative filtering model. In *KDD '08*, pages 426–434, 2008.
20. K. Lerman and T. Hogg. Using a model of social dynamics to predict popularity of news. In *WWW'10*, pages 621–630, 2010.
21. J. Liu, P. Dolan, and E. R. Pedersen. Personalized news recommendation based on click behavior. In *IUI '10*, pages 31–40, 2010.
22. V. Maidel, P. Shoval, B. Shapira, and M. Taieb-Maimon. Evaluation of an ontology-content based filtering method for a personalized newspaper. In *RecSys'08*, 2008.
23. P. Melville, R. J. Mooney, and R. Nagarajan. Content-boosted collaborative filtering for improved recommendations. In *Eighteenth national conference on Artificial intelligence*, pages 187–192, 2002.

24. D. Milne and I. H. Witten. Learning to link with wikipedia. In *CIKM '08*, pages 509–518, 2008.
25. R. Pan, Y. Zhou, B. Cao, N. N. Liu, R. Lukose, M. Scholz, and Q. Yang. One-class collaborative filtering. In *ICDM '08*, pages 502–511, 2008.
26. P. Pu, L. Chen, and R. Hu. A user-centric evaluation framework for recommender systems. In *RecSys '11*, pages 157–164, 2011.
27. T. Sandholm, H. Ung, C. Aperjis, and B. A. Huberman. Global budgets for local recommendations. In *RecSys '10*, pages 13–20, 2010.
28. H. Steck. Item popularity and recommendation accuracy. In *RecSys '11*, 2011.
29. X. Su and T. M. Khoshgoftaar. A survey of collaborative filtering techniques. *Advances in Artificial Intelligence*, 2009:4:2–4:2, Jan. 2009.
30. H. Zheng, D. Wang, Q. Zhang, H. Li, and T. Yang. Do clicks measure recommendation relevancy? an empirical user study. In *RecSys '10*, pages 249–252, 2010.