

Content Recommendation on Web Portals

Deepak Agarwal, Bee-Chung Chen, Pradheep Elango, Raghu Ramakrishnan

December 2, 2012

1 Introduction

Information discovery has been transformed by the web, and the impact on how information is gathered, published, and delivered has been profound. Web search is one form of discovery, and is preferred when a user has a specific objective, e.g., wants directions to an address. It is less effective when users simply want to be informed about news that may be relevant to them, or to learn more about topics of interest. In these latter scenarios, the user experience depends crucially upon the quality of *content recommendations*. In contrast to search, the explicit signal about what the user wishes to see is much weaker, and the importance of a broad range of complementary indicators increases greatly. Novel techniques are required to best leverage a broad array of weak signals.

In this paper, we present an overview of the *content recommendation* problem, namely how to recommend a small set of items to a user from an underlying pool of content items, in settings where the user does not explicitly express what is desired. In contrast to traditional media such as newspapers, every item shown on a page can be dynamically selected to reflect which items are currently the most engaging. In fact, the choice can be made taking into account a given user’s interests, and even what they are looking at in the current session. Data-driven dashboards allow editors to program new stories to reflect trending topics and changes in the demographics of users visiting the site, and to monitor performance metrics in near real-time. These changes are fundamentally altering how websites are designed, and indeed, how journalists and editors work.

Overview of Content Recommendation: The following issues must be considered in addressing the content recommendation problem:

Input signals: In building machine-learned models of what items a user is likely to engage with in a given context, we can draw upon many signals, including the content and source of each article, a user’s interest profile (reflecting both long-term interests based on prior visits and short-term interests as reflected in the current session), and “popularity” indicators such as observed *click-through rates* or CTRs (the fraction of time the item is clicked on when a link to it is presented to users) and extent of social-sharing (e.g., the number of times the item is tweeted or shared or “liked”).

Objective to optimize: There are many objectives a website could choose to optimize for, including near-term objectives such as CTR and revenue per click, as well as long-term metrics such as increased time spent on the site, higher return and user retention rates, increase in social actions, and many others.

Algorithmic techniques: Algorithms need to be developed to address a number of tasks.

- *Content analysis:* Create item profiles (e.g., feature vectors) that capture the content with high fidelity.
- *User profile modeling:* Create user profiles that reflect the items they are likely to consume.
- *Scoring:* Estimate the likely future “value”, for different types of values (e.g., CTRs, semantic relevance to the user’s current goal, or expected revenue), of showing an item to a user in a given *context* (e.g., the page the user is viewing, the device being used, the current location).
- *Ranking:* Select a ranked list of items to recommend so as to maximize the expected value of the chosen objective function.

Editorial tools: A complete recommendation framework must additionally provide tools for editors and journalists to: (1) observe in real-time what items are most interesting (to which user segments on which parts of the website, at what times, from what locations, and on what devices), (2) quickly identify emerging trends in order to create additional content to meet these information needs, (3) constrain the algorithmic recommendations to follow guidelines associated with the site, and (4) tune the objective function and balance trade-offs (e.g., maximize revenues while still keeping CTRs high and delivering personalized user experiences).

While creating user and item profiles and providing editorial tools are important aspects of content recommendation, we focus on scoring and ranking in this review. We argue that since user intent in content recommendation is significantly weaker compared to applications like web search (where user specified query serves as a strong signal of intent), expected click-through rate or CTR (appropriately weighted) is a more fundamental measure in scoring items for a user in a given context than semantic relevance. Also, ranking is not merely sorting items by scores — we must balance considerations like diversity (ensuring that users see a range of topics over time), serendipity (ensuring that we do not overfit our recommendations to the user, thereby limiting the discovery of new interests) and *editorial voice* (the general tone of the content associated with a given portal). Further, the objective function for ranking might be based on several criteria, e.g., we may want to maximize the number of article shares while not sacrificing more than 10% of the achievable CTR.

Table 1: Web Portals and Recommendation Modules

Portal Category	Examples	Typical Recommendation Modules	
General portal	www.yahoo.com www.msn.com www.aol.com	Home page	Featured module (FM: general) Featured module (FM: domain-specific)
Personal portal	my.yahoo.com igoogle.com	Home page	Featured module (FM: personalized)
Domain-specific portal	sport.yahoo.com	Home page	Featured module (FM: domain-specific)
	money.msn.com music.aol.com	Detail page	Related content module (RM)
Social network portal	facebook.com	Home page	Network update module (NM)
	linkedin.com twitter.com	Detail page	Related content module (RM)

2 Application Settings

In Table 1, we identify four types of portals: general, personal, domain-specific and social network. General portals publish a broad range of content; the home page of a content network typically falls in this category. A personal portal allows a user to customize the page with desired content; a domain-specific portal publishes content related to a specific topic or domain, e.g., sports; and a social network portal allows users to connect to other users and disseminate information through their network. Content modules published on these portals broadly fall into one of the following three categories:

- **Featured Modules (FMs):** These recommend “interesting” and recent content to users. Figure 1 shows examples of FMs on three different general portals. Such FMs show heterogeneous content with links to items hosted across the content network (e.g., sports, finance, etc.), and serve as a distribution channel sending users to different properties. General portals also have a set of *domain-specific* FMs that only recommend items from specific domains. For example, Figure 2 shows three domain-specific FMs for News, Sports and Entertainment on www.msn.com. On personal portals, *personalized* FMs provide recommendations that match each users interests. For example, Figure 3 shows a personalized FM called “News For You” on my.yahoo.com, which recommends content items from the RSS feeds that a user subscribes to. General and domain-specific FMs can also be lightly personalized. The content recommendation algorithms required for these three types of FMs have strong similarities.
- **Network-update Modules (NMs):** These recommend updates from (i.e., any information generated by) a user’s neighbors in a social network. In contrast to FMs, items shown in NMs often reflect updates that are

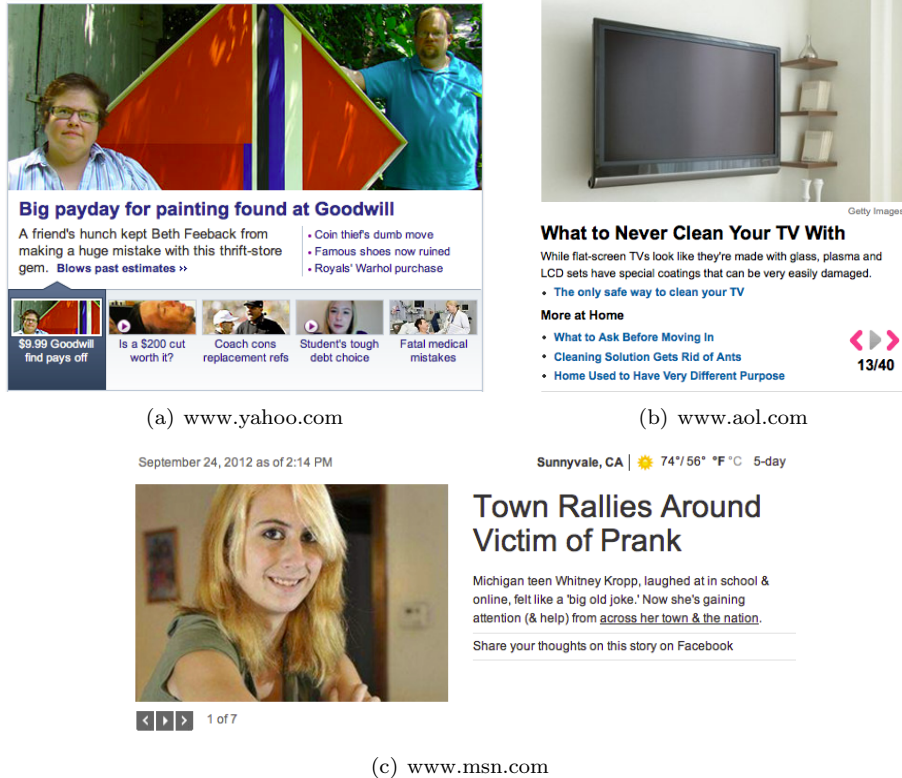


Figure 1: General FMs on Three Different Web Portals

restricted to be seen only by connections of the user.¹ They include social actions like sharing, liking and commenting. To make good recommendations, it is important to consider the reputation of the producer of an item, the strength of the connection between the producer and the recipient and the nature of the social actions involved.

- **Related-content Modules (RMs):** These usually appear on a page whose primary content (e.g., a news article) is the *context*, and recommend items “related” to that context. Figure 4 shows an example RM on huffingtonpost.com on the article “Mitt Romney health care plan wouldn’t slow rising costs.” In contrast to FMs and NMs, an RM has an additional piece of information, the context. Good recommendations are usually made by blending semantic relevance, item popularity and how closely an item matches the user’s interests.

Content recommendation techniques are important and in use by almost all major web portals. Instead of providing a review of specific techniques

¹Because of the private nature of NMs, we do not show screenshots.




NEWS	SPORTS	ENTERTAINMENT
 <p>'Risky' partnership helps students read</p> <ul style="list-style-type: none"> ▪ Congress faces debt crisis ▪ Why Ohio is battleground ▪ Vet's love story goes viral ▪ US couple slain in Caribbean Ⓢ US general speaks out ▪ Ben Franklin bust stolen 	 <p>Power Rankings: Are Ducks closer to No. 1?</p> <ul style="list-style-type: none"> ▪ NFL coaches fined for abuse ▪ Raiders WR out of hospital ▪ Climbers killed in avalanche ▪ Jets star likely out for season ▪ Hurt Miami RB got 'great news' ▪ First marathon for autistic man 	 <p>Dianna Agron's gleeful red-carpet fashions</p> <ul style="list-style-type: none"> ▪ Olsens skip 'House' reunion ▪ 'SpongeBob' actor, wife split ▪ 'Voice' to keep Green Day star ▪ Bieber gives sports car to pal ▪ Chevy: Louis C.K. 'not funny' ▪ Swift hints at CD surprises

Figure 2: Domain-Specific FMs on www.msn.com

News for You

Options



Can Obama's 'insourcing' plan revive industry?
 Yahoo! News - Latest News & Headlines - 3 hours ago

President Obama is touring a Master Lock factory in Milwaukee on Wednesday to urge manufacturers to return jobs to the United States-- part of a wider plan by his administration to increase the number of domestic manufacturing jobs. Right now we h...

Lin hits game-winner as Knicks beat Raptors Sports News Headlines - Yahoo! News - 20 hours ago

How Santorum, unafraid to talk poverty, differs from GOP on economic policy Yahoo! News - Latest News & Headlines - 8 hours ago

Lin's amazing story gets a thrilling finish Sports News Headlines - Yahoo! News - 6 hours ago

Bail-out politics: Even Michigan's economy is improving Yahoo! News - Latest News & Headlines - 9 hours ago

Dreamliner jet 'draws' Boeing logo across North America Odd News Headlines - Yahoo! News - 6 hours ago

Iran trumpets atom advances, deepening standoff with West Yahoo! News - Latest News & Headlines - 11 hours ago

How to Interpret Iran's New Nuclear Advances Iran News - Yahoo! News - 2 hours ago

Why Ahmadinejad is eager to show off new Iran nuclear facilities Odd News Headlines - Yahoo! News CA - 20 hours ago

Black customer who received "N-word" receipts settles with restaurant Odd News Headlines - Yahoo! News - 3 hours ago

1 - 10 of 30

Figure 3: News-For-You Module



Figure 4: Related-content Module on huffingtonpost.com

used by various portals, we review the broad framework that consists of two main technical components – scoring and ranking in various application settings. We believe this provides a crisp mathematical formulation of various use cases faced by web portals in practice. The actual methods used can be adequately described by a combination of techniques within this framework.

Technical solutions to scoring and ranking in content recommendation modules depend on the goals of the application and the nature of available signals; Table 2 lists a number of typical scenarios. We start our review with a simple application setting, where the goal is to maximize clicks in a recommendation module that has a relatively small content pool, and makes no use of user or context information. Although simple, this setting poses the challenge of finding the right balance between exploration and exploitation when estimating click-through rates (CTRs) of items using near real-time user click feedback. It also serves as a strong baseline method for non-personalized FMs, especially for portals that employ editors to select or create a small set of high quality content items. We then extend the application setting to personalized recommendation with a large content pool, which poses a data sparsity challenge because the amount of click feedback at detailed user interest levels is too little to support exploration of even a modest number of items. The key is to reduce dimensionality by leveraging user features and users’ past activities on the portal. The techniques reviewed here are useful for implementing personalized RMs and NMs. After tackling the data sparsity challenge, we discuss multi-objective ranking, which is important for RMs (where semantic relevance to the context page and CTR estimates need to be blended), and more generally, to optimize multiple objectives (e.g., clicks, social actions, revenue, time spent on the portal).

Table 2: Available signals

User	Are reliable user identifiers available?
	What user features (e.g., demographics, location) are available?
Item	What is the size and quality of the pool of candidate items?
	What item features (e.g., category, entities, keyword) are available?
Context	Is contextual information available?
	If so, what features of a context are available?
Feedback	Is user feedback (e.g., clicks, ratings) available?
	How quickly can we use current feedback to update models?

3 Scoring of Items

The fundamental technical challenge in scoring is to estimate the “value” of an item for a given user in a given context. Although one can use semantic relevance between query-document pairs as our score (in content recommendation, a user-context pair is a query), a more appropriate measure is the expected click through rate or CTR since the main signal from users in content recommendation is whether the user clicks the recommended items. Further, knowing the expected CTR for an item opens the door to a principled approach to ranking items, by weighting the CTR with the utility of a click on that item, which gives the expected utility. Hence, CTR (appropriately weighted) is the primary scoring function we consider in this review. While scores based on other signals like explicit feedback (e.g., like/dislike) useful in some applications can be estimated by some of the reviewed techniques, they are not our main focus. With known item CTRs, we could maximize clicks by always recommending the item with highest CTR. But since CTRs are not known, a key task is to accurately estimate the click-through rate (CTR) of each candidate item in the selected pool, perhaps through an *exploration* process (displaying each item to some number of user visits). However, exploration has an opportunity cost of not showing items that are empirically better; balancing these two aspects constitutes the explore/exploit trade-off. Dynamic item pools and non-stationarity of CTR over time adds more complexity.

3.1 The Explore/Exploit Trade-off

To obtain further intuition on the explore/exploit problem, consider a simplified setting with a content pool consisting of two items, where the goal is to obtain an optimal algorithm to maximize overall expected CTR for the next 100 user visits. Note that the solution space here is *astronomical*—there are 2^{100} different possible recommendation sequences (over two trillion!). This is similar in spirit to the classical multi-armed bandit problem, which considers how to dynamically allocate a single resource to alternate projects [32]. Remarkably, an optimal solution exists and involves adaptively changing future decisions based on past feedback as discussed by Gittins in [11]. It illustrates the fundamental

“explore/exploit” tradeoff between “exploit” (display the item that appears to be doing well so far) and “explore” (display the item that may appear inferior, but perhaps due to an unlucky streak, to determine its true popularity). For instance, suppose the estimated CTR after 20 visits for items 1 and 2 are $1/3$ and $1/5$ respectively. It is tempting to abandon item 2 and persist with item 1 for the remaining 80 visits, but that may not be optimal since the true CTR for item 2 could be *potentially* higher than the estimated one, which is noisy due to the small sample size. We refer interested readers to [3, 11, 7] for more details on multi-armed bandit problems.

For content recommendation, several assumptions required to obtain Gittins’ optimal solution are violated. The item pool is not static and changes over time, the CTR themselves could be non-stationary, and the click feedback is delayed (due to delay by users in clicking an item after display and delay in data transmission from web servers to the backend machines). But perhaps the most difficult issue is the curse of dimensionality introduced due to the need to provide personalized recommendation with large/dynamic content pool, and hence the dearth of experimental budget to estimate popularity at fine resolutions. Hence content recommendation problems require different solutions and cannot be solved through classical multi-armed bandit schemes, we review the current approaches subsequently.

3.2 Explore/exploit for Content Recommendation

We shall now discuss technical approaches to solving the explore/exploit problem for content recommendation assuming CTR to be our score function.

3.2.1 Most Popular Content Recommendation

We begin with the problem of recommending the most popular item on a single slot to all users to maximize the total number of clicks. Although simple, this problem of *most popular recommendation* includes the basic ingredients of content recommendation and also provides a strong baseline for more sophisticated techniques.

Estimating item popularity (CTR) involves several nuances. Ideally, popularity for each item should be estimated by displaying the item to a representative sample of the current user population. For instance, serving most popular at night with popularity estimated using data in the morning may not perform well due to differences in user populations. Several other sources of bias can also affect popularity estimates when using retrospective data collected from existing systems. To protect against such bias, it is useful to update popularity estimates rapidly in an adaptive fashion through procedures like a Kalman filter [30], using data obtained through randomization, i.e., randomly assigning some fraction of visits in a given time epoch to each item in the content pool. The optimal amount of randomization for each item can be computed by using extensions of bandit schemes described in [3].

3.2.2 Personalization and Large Content Pools

A natural extension to most popular recommendation is to classify users into coarse segments based on attributes like demographics and geographic location, and then apply most popular recommendation techniques to each segment. Such an approach works well if segments are coarse and item affinities for users within a segment are relatively homogeneous. Techniques like clustering and decision trees [13] can be used to identify such segments.

However, this *segmented most popular recommendation* approach only works when the number of user visits per segment available to explore each candidate item is “sufficiently” large to ensure reliable identification of the highest CTR items. It is therefore challenging to provide personalized recommendations from a large pool of items, since it may not even be possible to show each item to each user even once!

3.3 Explore/Exploit with Data Sparsity

Several applications require personalization with a large and dynamic content pool, this contributes to data sparsity.

Approaches to Tackling Sparsity:

- **Dimension reduction:** Creating homogeneous groups through user and item meta-data, and/or user behavioral data reduces dimensionality and helps in combating data sparsity.
- **Coupling dimension reduction with explore/exploit and online updates:** Although it is ideal to couple explore/exploit techniques with dimension reduction, obtaining an optimal solution is difficult. Heuristics that combine dimension reduction techniques with classical explore/exploit algorithms are often used in practice. In addition, it is important to update model parameters for time-sensitive items in an online fashion using most recent user feedback.

In the following, we first review dimensionality reduction techniques and then discuss how to apply bandit schemes to the resulting low dimensional user and item representations. Subsequently, we review methods for constructing such representations using online updates, and how to effectively initialize these online methods.

3.3.1 Dimensionality Reduction Techniques

A proper survey of dimensionality reduction goes beyond the scope of this review; we only cover a few representative approaches.

Grouping through hierarchies: In some scenarios, users and/or items are hierarchically organized. For instance, the city in which a user lives is nested within a state, which in turn is nested within a country (Figure 5). If such

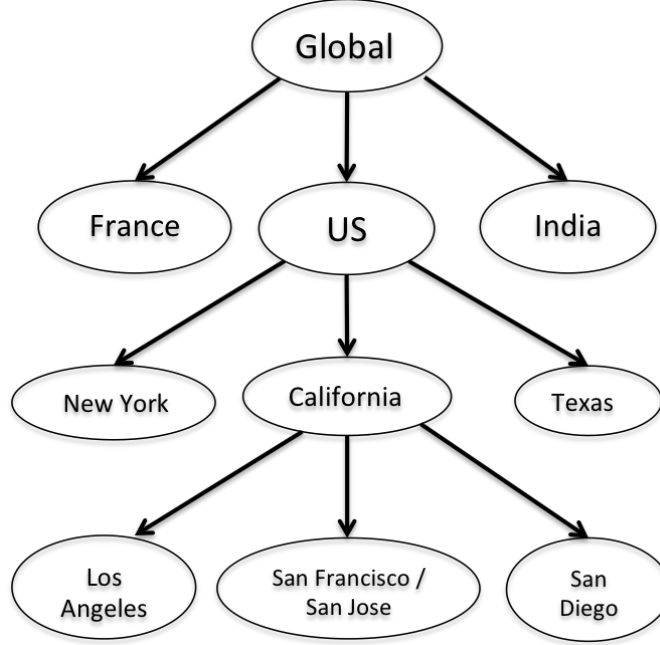


Figure 5: A sample hierarchical organization of a user’s location

hierarchies do not exist, hierarchical clustering or decision tree learning [13] may be applied to automatically create hierarchies from data.

Instead of exploring/exploiting individual items for each individual user, one can start with exploring/exploiting coarse content categories for coarse user segments and gradually transition to fine-grained user segments and items as we obtain more data [17, 29].

Dimension reduction through linear projections: Another popular approach is to work in a generalized linear model framework [28]. In many applications, we have a rich set of user features such as demographics, geo-location, and behavioral activities such as searches. Let us denote by $\mathbf{x}_i = (x_{1i}, \dots, x_{Mi})$ the feature vector for user i . The number M of such features is typically large (thousands or even millions). A generalized linear model assumes that the CTR of item j for user i is a monotone function of a linear combination of features $\mathbf{x}_i' \boldsymbol{\beta}_j$, and the unknown item coefficient vector $\boldsymbol{\beta}_j$ is estimated by using item interaction data across users. Although such a model reduces the CTR estimation problem from estimating a CTR for each (user, item) pair to estimating M coefficients for each item, it is still daunting when M is large. One approach is to project $\boldsymbol{\beta}_j$ into a low-dimensional space through a linear projection matrix B ; i.e., $\boldsymbol{\beta}_j = B\boldsymbol{\theta}_j$ where $\boldsymbol{\theta}_j$ is low dimensional. The projection B can be estimated in an unsupervised fashion by using principal component analysis (PCA) [13] on user features, a supervised approach that uses additional click

feedback information provides better performance as discussed in [4].

Grouping through collaborative filtering: In typical content recommendation applications, a certain fraction of users tend to interact frequently with the recommendation module. For such users, it is possible to derive features that capture item affinity based on past user interactions alone. It is also possible to derive such features for users who are less frequent by using techniques like collaborative filtering. For instance, based on data from the entire user population, one can estimate associations of the form: “users who like item A also like item B ”. The stronger these associations, the smaller the effective dimensionality, because they induce soft constraints on the joint distributions of CTR of all user-item pairs. Such approaches work well in practice even in the absence of other user and item features [1, 8].

Collaborative filtering approaches based on factor models currently provide state-of-the-art performance [19]. The idea is to map user i and item j into the same Euclidean space as *factor vectors* \mathbf{u}_i and \mathbf{v}_j respectively—user-item affinity is then given by the inner product $\mathbf{u}_i' \mathbf{v}_j$ of \mathbf{u}_i and \mathbf{v}_j , which is a measure of similarity between the two. Unlike explicit interest categories, these groups are latent and are estimated from retrospective data; a small number (few tens to hundreds) of factors usually provide good performance in applications.

3.3.2 Explore/Exploit and Dimension Reduction

As we discussed before, obtaining an optimal explore/explore solution for personalized recommendation using dimension reduction is non-trivial and hence heuristics are often used in practice. The simplest solution is called ϵ -greedy. It serves an item selected at random with probability ϵ to each user visit and, with probability $1 - \epsilon$, it serves the item having the highest estimated CTR for that user (see [20, 16] for more details). Upper-Confidence-Bound (UCB) scheme [7, 21] that rank items based on an overestimate of mean (e.g., mean + k std, where k is some positive number and std is the estimated standard deviation) and Thompson sampling [37] that sample parameters from the posterior distribution are other commonly used schemes.

3.3.3 Online CTR Estimation

CTR estimation models for time-sensitive items needs to be updated frequently using the most recent user feedback. Such *online models* start with an initial estimate of the model parameters and continuously update parameters as new data becomes available. For instance, consider the factor model that predicts the CTR (on the log-odds scale) of item j for user i by $\mathbf{u}_i' \mathbf{v}_j$. Assume user factor estimates are more stable than items; thus, only \mathbf{v}_j need to be updated in an online manner. From a Bayesian perspective, without any click feedback on item j , we postulate a prior distribution for \mathbf{v}_j with mean $\boldsymbol{\mu}_j$ and some variance. After receiving feedback (click or no-click), we update the prior to obtain the posterior distribution of \mathbf{v}_j through Bayes’ rule; the resulting posterior serves

as the prior for subsequent updates. See [5] for an application of this technique to a web portal.

3.3.4 Initializing Online Models

For an item j that is new or has received little click feedback, CTR prediction depends crucially on the prior mean μ_j , which is the initial estimate. One may naively set μ_j of all items to the same value for all users, say $\mathbf{0}$. This can be improved by initializing item factors v_j through feature vector z_j for item j as $v_j = D(z_j) + \eta_j$, where D is a multivariate regression function and η_j are correction terms learning item-specific idiosyncracies not captured through item features. Similar considerations apply to user factors. We refer the reader to [2, 33] for more details.

4 Ranking

After obtaining scores of some particular type (e.g., CTR estimates) for each user-item pair, one could simply rank items by sorting scores. However, different types of scores (e.g., CTR and semantic relevance) may have to be combined, and a number of competing recommendation objectives may have to be balanced. We first present two ranking scenarios to illustrate these nuances, and then discuss how to combine measures for multi-objective ranking.

- **Personalized ranking with multiple objectives:** Although CTR is an important signal, other types of complementary signals can also be leveraged to improve personalized ranking. Examples include post-click actions (e.g., time spent reading, sharing, commenting, and others), declared interests, explicit feedback (like/dislike), and serendipity measures (to ensure that users are not overly pigeon-holed).
- **Related Item Recommendation:** CTR and semantic relevance need to be combined to recommend related items. Figure 6 shows an example of a Related Content Module on a Yahoo! News article page, which we call the context article. The candidate items are news articles and videos “related” to the context article. In this setting, contextual information is important. Users expect to see items that are semantically relevant to the article that they are currently reading; although highly clicked irrelevant items are not appropriate in this setting, items with low CTRs are also undesired. The semantic relevance measure needs to ensure that the candidate item not only is similar to the context item, but also provides sufficient new information not in the context item [26]. Also, CTR estimation needs to take not only the user into consideration, but also the context item, to predict the probability $p(j|i, k)$ that user i would click item j given that he/she likes the context item k , where “like” can be interpreted as click, read, share, etc. As in Section 3.3, reducing dimensionality is even more important here since data becomes more sparse after incorporating



Figure 6: Related Content Module

the context. For an example of such dimensionality reduction, see [31]. To prevent popular items from dominating recommendations, one may down-weight conditional CTR by unconditional CTR (e.g., as in [10]), $p(j|i, k)/p(j)$ or $p(j|i, k)/p(j|i)$, or using an affinity measure like the odds ratio).

One potential approach to combining multiple objectives is to compute a score quantifying the value of an item with respect to each of the aspects of interest and then combine these scores through a weighted average. A more general framework of *multi-objective optimization* can also be used.

Multi-Objective Optimization: Balancing multiple objectives has broad applicability beyond the two scenarios we just presented. For example, it is often desired to consider various weighted CTR objectives that measure different expected post-click utilities like ad-revenue, time spent, likelihood of sharing, etc. Multi-objective optimization provides an attractive mathematical framework to achieve such trade-offs. Although there is a rich and mature literature on multi-objective programming [34], its application to content optimization is fairly new. For instance, [6] use this approach to simultaneously optimize CTR and time spent reading the article to recommend content links on the Yahoo! homepage. Two recent studies that apply multi-objective optimization in the areas of collaborative filtering and learning to rank are described in [14, 35].

5 Evaluation

As discussed earlier, optimal explore/exploit solution for personalized content recommendation is still elusive. Existing approaches are based on heuristics, so no single strategy necessarily dominates and the actual solutions are application dependent. As such, it is essential to have rigorous and principled approach to empirically evaluate the quality of solutions.

Strategies to evaluate the performance of an algorithm depends on the aspect we are interested in quantifying. In general, we evaluate two aspects of algorithms used in content recommendation — out-of-sample prediction accuracy and overall recommendation performance.

5.1 Evaluating Predictive Accuracy

A prediction algorithm predicts an unobserved quantity. For example, algorithms that try to predict the CTR of an item by a user fall into this category, including methods discussed earlier in this review. Standard machine-learning or statistical evaluation methods can be used to evaluate the accuracy of predictions. A common approach is to collect a click log from the system that records whether users clicked items shown to them, and then split the log data into a training set and a test set. The training set is used to build a model, which is then evaluated using the test set. Commonly used metrics include log-likelihood of the model on the test data, precision-recall curve, ROC curve, AUC (area under ROC curve) [38]. We note that algorithms that predict quantities other than CTR (e.g., semantic relevance, time-spent) can also be similarly evaluated as long as ground-truth data is obtainable (e.g., by human judgment or processing system logs).

5.2 Evaluating Recommendation Performance

Overall recommendation performance is typically measured by using a notion of “reward” (e.g., total number of clicks) that the system receives from its users. Explore/exploit algorithms and ranking algorithms (both of which use the output of some prediction algorithms to decide which items to show to each user) fall into this category. As long as the reward is measurable, algorithms can be compared through online experiments, in which random sub-populations of users are subjected to different serving algorithms to adjust for population bias in measuring performance improvements. Such experiments are frequently conducted by Web portals, and are usually referred to as *bucket tests*. See [18] for a survey.

Bucket tests are expensive and may be risky, since an inferior serving algorithm may degrade user experience. It is desired to be able to evaluate an algorithm using retrospective data collected from the system, instead of experimenting with live traffic. Offline evaluation helps in reducing experimental cost and also facilitates research by the broader community, many of whom may not have access to experimental infrastructure. However, it can be difficult to obtain an unbiased estimate of the reward from retrospective data. For example, if an item has never been shown to a user in the retrospective data, estimating the number of clicks that will be generated when the algorithm recommends the item is difficult. If we base the evaluation only on the items that have been shown to the user by the current “serving algorithm”, then this evaluation would be biased. Fortunately, as long as the historical algorithm has a non-zero probability of showing each item to each user, the bias can be removed by using importance sampling techniques. For example, Li et al. [22] developed such an unbiased evaluation method for explore/exploit algorithms, and empirically showed that the reward of an algorithm estimated using retrospective data correlates well with its actual reward in an online bucket test.

5.3 Data for Research Purposes

To facilitate further research in this area, it is important to have benchmark datasets available for evaluating new algorithmic approaches. Several such datasets are available to test predictive accuracy of new approaches ([12, 40, 15]), while not many datasets are available to evaluate the overall recommendation performance. Fortunately, Yahoo! [39] recently provides the first such dataset. More such datasets are required to facilitate participation by a large number of researchers in this area. This is a challenging task since such datasets are most easily generated by large web companies but such companies are often constrained by concerns over user privacy. Further collaboration between academic and industrial research is necessary.

6 Content Recommendations in Current Web Portals

Content recommendation techniques have been adopted by every major web portal to increase user engagement [27]. We have presented a rigorous technical framework covering all real use cases that we are aware of, rather than a survey of individual methods used by various portals. The actual methods used can be adequately described by a combination of techniques within this framework.

Several portals use methods that are close to related item recommendation techniques described earlier; typically, they use a combination of collaborative filtering and semantic similarity. Such techniques have been successfully used in e-commerce (e.g., Amazon [24]) and entertainment (e.g., deezer.com). One of the earliest uses of these techniques for content recommendation was by Findory [23] to recommend news articles based on semantic similarity between a user’s content profile (obtained through historical reads) and article content, blended with a collaborative filtering component that recommends articles read by an user to others with similar profiles. More recently [9] describes an on-line collaborative filtering framework to recommend news on Google, using pure collaborative filtering (semantic relevance is not incorporated). In a subsequent paper [25], content-based user profiles are used to enhance the collaborative filtering algorithm and popularity estimates. The YouTube video recommendation system described in [10] scores a candidate pool of videos in a multi-objective framework with respect to video quality, user specificity and diversity, via a linear combination. The candidate pool of videos for a user is generated by considering related videos (based on co-visitation patterns within the same session) in the relatedness graphs that are at most n hops from the seed set (which consists of videos previously watched, commented, shared or liked by the user).

Several portals recommend articles using some estimates of article popularity. For instance, [36] describe methods to estimate long-term article popularity based on historical data. [5] showed that this approach suffered from the presence of serving bias in historical data, and used popularity estimation methods based on randomized data to recommend articles on the Yahoo! front page.

Several other portals including AOL, LinkedIn and MSN have modules on their homepage that rely on methods to estimate article popularity. Factor models to personalize recommendations are known to be used by Netflix [19] for movie recommendations and Yahoo! for personalized content recommendations [2].

7 Conclusion

In this overview, we motivated the content recommendation problem and observed that it is sufficiently different from search and ad targeting to merit careful study and distinct technical approaches. We presented a rigorous framework that allows us to capture current recommendation approaches, used widely on several web portals.

However, challenges and open problems remain, including the following:

- *Incorporating externalities*: How does the CTR of an item link depend on other links on the page, and how can we leverage this dependency to better optimize the whole page?
- *Combining heterogeneous user feedback from multiple sources*: How can we take advantage of correlations among many kinds of user feedback (e.g., clicks, tweets, shares, and email actions) from many sources (e.g., browsing and device histories), and combine these signals to better rank items?
- *Scalable content curation*: How can we acquire and curate a large number of content items to ensure content quality? Large, high-quality content pools are essential for good personalized recommendations, since we must cater to a wide range of user tastes.
- *Algorithm-enabled journalism*: The blending of algorithmic recommendations with journalism and editorial oversight is a fascinating aspect of the rapidly growing role of algorithmic content recommendation. The role of editors and journalists is clearly changing, but algorithms work best when judiciously leveraged with humans in the loop.

References

- [1] G. Adomavicius and A. Tuzhilin. Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. *IEEE Trans. on Knowledge and Data Engineering*, 17:734–749, June 2005.
- [2] D. Agarwal and B.-C. Chen. Regression-based latent factor models. In *KDD*, pages 19–28, 2009.
- [3] D. Agarwal, B.-C. Chen, and P. Elango. Explore/exploit schemes for web content optimization. In *ICDM*, pages 1–10, 2009.

- [4] D. Agarwal, B.-C. Chen, and P. Elango. Fast online learning through offline initialization for time-sensitive recommendation. In *KDD*, pages 703–712, 2010.
- [5] D. Agarwal, B.-C. Chen, P. Elango, N. Motgi, S.-T. Park, R. Ramakrishnan, S. Roy, and J. Zachariah. Online models for content optimization. In *NIPS*, 2008.
- [6] D. Agarwal, B.-C. Chen, P. Elango, and X. Wang. Click shaping to optimize multiple objectives. In *KDD*, pages 132–140. ACM, 2011.
- [7] P. Auer, N. Cesa-Bianchi, and P. Fischer. Finite-time analysis of the multiarmed bandit problem. *Machine Learning*, 2002.
- [8] A. Das, M. Datar, A. Garg, and S. Rajaram. Google news personalization: scalable online collaborative filtering. In *WWW*, pages 271–280. ACM, 2007.
- [9] A. Das, M. Datar, A. Garg, and S. Rajaram. Google news personalization: scalable online collaborative filtering. In *WWW*, 2007.
- [10] J. Davidson, B. Liebald, J. Liu, P. Nandy, T. Van Vleet, U. Gargi, S. Gupta, Y. He, M. Lambert, B. Livingston, and D. Sampath. The youtube video recommendation system. In *ACM conference on Recommender systems*, RecSys ’10, pages 293–296. ACM, 2010.
- [11] J. Gittins. Bandit processes and dynamic allocation indices. *J. of the Royal Statistical Society, B*, 41, 1979.
- [12] GroupLens Research. Movielens Data Sets, <http://www.grouplens.org/node/73>.
- [13] T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning*. Springer, 2009.
- [14] T. Jambor and J. Wang. Optimizing multiple objectives in collaborative filtering. In *Proceedings of the fourth ACM conference on Recommender systems*, RecSys ’10, pages 55–62, New York, NY, USA, 2010. ACM.
- [15] Kaggle. <http://www.kaggle.com>.
- [16] S. Kakade, S. Shalev-Shwartz, and A. Tewari. Efficient bandit algorithms for online multiclass prediction. In *ICML*, pages 440–447. ACM, 2008.
- [17] L. Kocsis and C. Szepesvari. Bandit based monte-carlo planning. In *Machine Learning: ECML*, Lecture Notes in Computer Science, pages 282–293. Springer, 2006.
- [18] R. Kohavi, R. Longbotham, D. Sommerfield, and R. Henne. Controlled experiments on the web: survey and practical guide. *Data Mining and Knowledge Discovery*, 18:140–181, 2009. 10.1007/s10618-008-0114-1.

- [19] Y. Koren, R. Bell, and C. Volinsky. Matrix factorization techniques for recommender systems. *Computer*, 42(8):30–37, 2009.
- [20] J. Langford and T. Zhang. The epoch-greedy algorithm for contextual multi-armed bandits. In *NIPS*, 2007.
- [21] L. Li, W. Chu, J. Langford, and R. Schapire. A contextual-bandit approach to personalized news article recommendation. In *Proceedings of the 19th international conference on World wide web*, WWW ’10, pages 661–670. ACM, 2010.
- [22] L. Li, W. Chu, J. Langford, and X. Wang. Unbiased offline evaluation of contextual-bandit-based news article recommendation algorithms. In *Proceedings of the fourth ACM international conference on Web search and data mining*, WSDM ’11, pages 297–306, New York, NY, USA, 2011. ACM.
- [23] G. Linden. In <http://glinden.blogspot.com/2008/01/brief-history-of-findory.html>.
- [24] G. Linden, B. Smith, and J. York. Amazon.com recommendations: Item-to-item collaborative filtering. *IEEE Internet Computing*, 7(1):76–80, 2003.
- [25] J. Liu, P. Dolan, and E. R. Pedersen. Personalized news recommendation based on click behavior. In *Proceedings of the 15th international conference on Intelligent user interfaces*, IUI ’10, pages 31–40, 2010.
- [26] Y. Lv, T. Moon, P. Kolari, Z. Zheng, X. Wang, and Y. Chang. Learning to model relatedness for news recommendation. In *WWW*, pages 57–66. ACM, 2011.
- [27] Mondaynote. In <http://www.mondaynote.com/2009/02/15/recommendation-engines-a-must-for-news-sites>, 2009.
- [28] J. Nelder and R. Wedderburn. Generalized linear models. *Journal of the Royal Statistical Society. Series A (General)*, 135:370–384, 1972.
- [29] S. Pandey, D. Agarwal, D. Chakrabarti, and V. Josifovski. Bandits for taxonomies: A model-based approach. In *SIAM International Conference on Data Mining*, 2007.
- [30] A. Pole, M. West, and P. J. Harrison. *Applied Bayesian Forecasting & Time Series Analysis*. Chapman-Hall, 1994.
- [31] S. Rendle, C. Freudenthaler, and L. Schmidt-Thieme. Factorizing personalized markov chains for next-basket recommendation. In *WWW*, pages 811–820. ACM, 2010.
- [32] H. Robbins. Some aspects of the sequential design of experiments. *Bull. Amer. Math. Soc.*, 58:527–535, 1952.

- [33] D. Stern, R. Herbrich, and T. Graepel. Matchbox: large scale online bayesian recommendations. In *WWW*, pages 111–120. ACM, 2009.
- [34] R. Steuer. *Multi-criteria optimization: theory, computation and application*. Wiley, 1986.
- [35] K. M. Svore, M. N. Volkovs, and C. J. Burges. Learning to rank with multiple objective functions. In *Proceedings of the 20th international conference on World wide web*, WWW '11, pages 367–376, New York, NY, USA, 2011. ACM.
- [36] G. Szabo and B. A. Huberman. Predicting the popularity of online content. *Commun. ACM*, 53(8):80–88, Aug. 2010.
- [37] W. R. Thompson. On the likelihood that one unknown probability exceeds another in view of the evidence of two samples. *Biometrika*, 25(3-4):285–294, 1933.
- [38] I. H. Witten, E. Frank, and M. A. Hall. *Data Mining: Practical Machine Learning Tools and Techniques*. Morgan Kaufmann, Burlington, MA, 3 edition, 2011.
- [39] Yahoo! Academic Relations. R6A - Yahoo! Front Page Today Module User Click Log Dataset, Version 1.0, <http://webscope.sandbox.yahoo.com>, 2012.
- [40] Yahoo! Academic Relations. Yahoo! webscope rating datasets, <http://webscope.sandbox.yahoo.com/catalog.php?datatype=r>, 2012.