# Click-Based Content Recommendation for Hierarchical Semantic Streams

## ABSTRACT

We design a unified recommendation system for hierarchical personalized news streams, using three sequential phases to optimize for serving-time efficiency. Within each phase we use user click count feedbacks as training signals. This solves the issue of insufficient training labels in the context of content multi-classification (phase 0). In the personalization stage, we judiciously balance multilinear regression (phase 1) with gradient boosting models (phase 2) to ensure coverage, recency, and relevance at the same time, while maintaining low latency. The models include various features inspired by recency, personalization, popularity, and target stream. Significant improvements are observed in both offline experiments and online bucket test.

## Categories and Subject Descriptors

H.3.m [**Information Search and Retrieval**]: Miscellaneous

## General Terms

Algorithms, Information Retrieval

## Keywords

News recommendation, Information retrieval, Personalization, Semantic stream classifier, Multi-stream recommendation, Recency

## 1. INTRODUCTION

Developing efficient and accurate personalized recommendation systems of news-worthy content has been a focus of today's media industry. On the one hand search engines have enabled users to locate articles or multimedia content of any topic within a few keystrokes. On the other hand, users are constantly overwhelmed by the amount of information generated daily, much of which is irrelevant to their personal interest. Thus directory-style news streams organized by topics are becoming increasingly popular.

Under the latter framework, we not only need to recommend generically interesting articles, but those within a specific category, such as sports, finance, or technology, to a diverse array of users. Underneath each broad category lies more sub-categories that facilitate users to find relevant and interesting content quickly (Figure 1).

Different from traditional news recommendation for a single stream, the first challenge is to assign articles into corresponding semantic streams as shown in the figure. One obvious approach is to implement a document filter for each category node on the content hierarchy. However, the semantic streams are not static structures. It is evolving. New nodes may be added by any reasons such as sports (Olympics games), natural disasters (MH370) or political events (obamacare). To train a new document filter requires the labeling of training examples. Given the number of streams (easily in the 1000's) in question and its dynamic property, it is impossible to solve it by editorial resource. But document filter is still useful as a starting point. We address the problem with a combination of user click feedbacks and editorial labeling.

Even after the filter problem is solved, users visiting different category streams can have very different sets of browsing interests. Furthermore, articles that are popular in a general stream may be less popular in a substream, and vice versa. While lower streams give more accurate user intent, higher streams provide more abundant user feedback. Each stream therefore may require its own personalization and ranking model for optimal user experience. Scalability is again a major practical concern here. In addition, many streams have very limited user statistics to build a meaningful and robust model, which suggests that we take a holistic approach by combining streams at lower branches of the hierarchy for model training.

Justified by the above reasoning, we therefore propose a system design with essentially 2 components: a category stream filter based on linear classifier (phase 0) followed by a personalization and ranking machine. Due to runtime constraints, the latter is further subdivided into two phases, with phase 1 selecting among hundreds of thousands of articles only the top 200 candidates according to stream relevance and user preferences, using a simple linear model with only positive weights (negative pruning). The features consist of entity scores within the user and document profile, the popularity score of the article, measured in terms of discounted click-through rate(CTR), and finally the freshness of the content. Phase 2 then performs re-ranking of those 200 articles using the more sophisticated Gradient Boosted

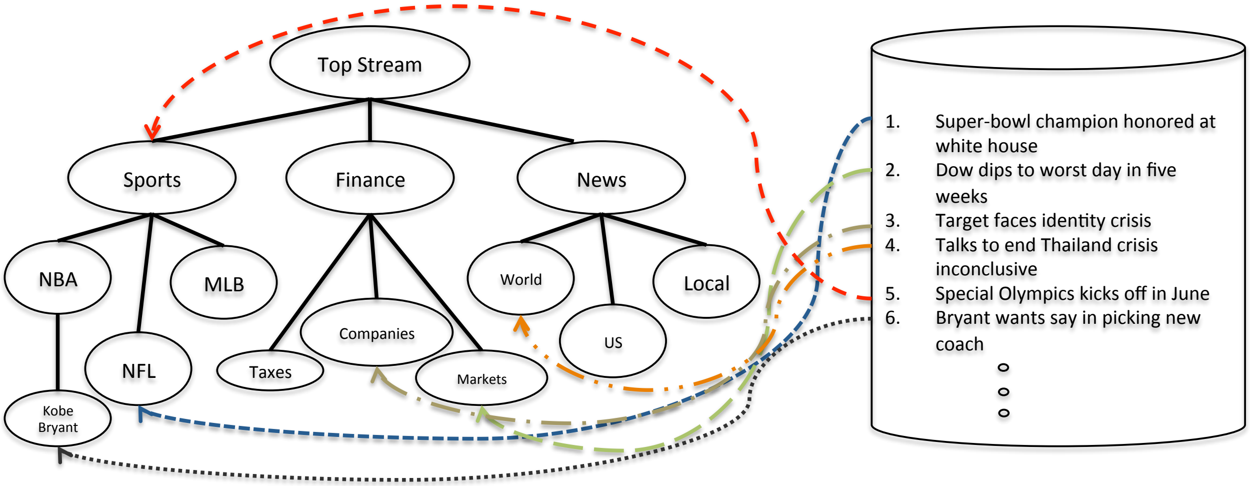

Top Stream
Sports
Finance
News
NBA
MLB
NFL
Kobe Bryant
Taxes
Companies
Markets
World
US
Local
1. Super-bowl champion honored at white house
2. Dow dips to worst day in five weeks
3. Target faces identity crisis
4. Talks to end Thailand crisis inconclusive
5. Special Olympics kicks off in June
6. Bryant wants say in picking new coach

Figure 1: Semantic Stream Hierarchy

Decision Tree (GBDT) model, and integrating tens of new features including a content type feature that is unique for multiple semantic stream ranking. Altogether, the three phases yield an efficient approximate solution to the optimal ranking problem; an exact solution is clearly too expensive for any reasonably sized content/user pools.

The paper is organized as follows. Stream classifier is described in section 3 including feature source discussion (3.2), stream classifier(section 3.3), Phase 1 ranking (3.4) and Phase 2 ranking (3.5). Experimental results are in section 4 where we describes metrics, phase 0, phase 1 and phase 2 results. Section 5 concludes the paper.

## 2. RELATED WORK

We have not found similar work on semantic stream classification based on clicks. Many traditional document classification methods are based on static topic definitions and rely heavily on editorial resources [21]. For instance, web document link structures were exploited to predict document categories in the work of [4, 12, 3]. However, streams change in topical coverage from time to time, and can be created or updated on demand to reflect popular need. In our work, document classifier is a starting point of our stream classifier. On top of it, the stream classifier is rebuilt by optimizing users' click behavior on the steam.

All our models from Phase 0 to Phase 2 are built using targets based on clicks. Using number of clicks as features or target has proved to be effective in many fields, such as recommendation system and search ([2, 17]). Some recent work give more up-to-date improvment on using click [15, 1].

The logistic regression model used in Phase 0 is a classical supervised approach[13], but the click-based label heuristic appears new. An additional challenge is the number of token features used, which go up in the hundreds of thousands. For the experiments we mainly relied on Vowpal Wabbit [24] and LibLinear [8], the latter implemented in Weka [1].

We also could not find any previous work on the twisted dot-product idea behind Phase 1. However it is closely related to tensor factorization models [20], [27]. In our case, the tensor has three dimensions: user, document, and stream, and it is diagonal along each "stream slice".

Recency is an important metric to consider in recommendation and Web search. However promoting recency tends to undercut personalized relevance, making it difficult to balance the two. Some pioneering work([7]) discussed the problem and proposed solutions, where a recency query classifier is used. We deal with recency by incorporating the recency feature into a machine learned model.

Collaborative filtering is widely used in content recommendation [11, 26, 23, 5]. This approach recommends new articles to the user by virtue of similiar interests with other users who have read the articles. Similar users are grouped by some methods like matrix factorization [14]. If a user has read an article, this article can be recommended to other users within the same group. Our work is not based on collaborate filtering, but we can use it to derive new users' profile.

Recently, explore/exploit is a popular strategy [2, 1, 15]. These methods are useful for a single stream, but not scalable for multiple semantic streams. To design a recommendation system we must consider system latency. But we used the idea of exploration bucket to collect unbiased training data in Phase 2.

## 3. SYSTEM DESIGN

### 3.1 Motivations

Designing an actionable system for thousands of article streams with millions of contents to present is a highly nontrivial task that requires both latency consideration and machine learning insight. The latter can further be divided into several categories of objectives, such as relevance, freshness, personalization, segmented popularity, etc. In order to build an agile and well-focused pipeline, it is imperative to modularize the effort into different phases. Thus we devote a document classifier phase (phase 0) to optimize for relevance, a pre-screening stream-specific linear matching phase (phase 1) to optimize for serving-time latency, followed by a more powerful ranking algorithm (phase 2); both phase 1 and phase 2 take into account personalization and popularity. In general, having a multi-phase system can be a

[1] http://www.cs.waikato.ac.nz/ml/weka/

challenge in bucket tests or even offline optimization. Fortunately, due to their conceptual independence, we argue that it is relatively harmless to optimize them in parallel.

## 3.2 Feature Engineering

Throughout the modeling work below, we use a few in-house features that require explanation. Users will be consistently denoted by $u_i$ and entities by $e_j$. One important feature, gmp, captures the general popularity of an entity, as it appears in articles. It is calculated by the following recency-discounted click-through rate formula:

$$\mathrm{gmp} = \frac{\sum_{\mathrm{k}} \lambda^{\mathrm{k}} \mathrm{N}_{\mathrm{k}}^{\mathrm{clicks}}}{\sum_{\mathrm{k}} \lambda^{\mathrm{k}} (\mathrm{N}_{\mathrm{k}}^{\mathrm{clicks}} + \mathrm{N}_{\mathrm{k}}^{\mathrm{skips}})},$$

where $\lambda \in (0, 1)$ is a time-decay factor, $N_k^{\text{clicks}}$ is the number of times the item (i.e., articles containing the entity) has been clicked between $5k$ and $5(k+1)$ minutes ago, and $N_k^{\text{skips}}$ is the number of times the item has been skipped, i.e., not clicked and another item at a lower position in the stream was clicked in the same session, also within that time interval.

For personalization, two other classes of features are used: user profile and document profile. Each user profile consists of a bag of entities and a bag of semantic categories extracted from user's historical readings. $\{(e_0, sp_0), (e_1, sp_1), \cdots, (e_N, sp_N), (c_0, sp_0), (c_1, sp_1), \cdots, (c_M, sp_M)\}$ denotes a user profile with $N$ entities and $M$ categories. Within a fixed profile for user $i$, each entity or category $j$ is assigned a so-called sparse-polarity score, $sp$, to represent the user's level of interests (LOI) in the entity. Sparse polarity score, quantifies a user's above-average interest in a particular entity and category. It's computed as a one-sided log z-score as follows. Let $N_{ij}$ denote the number of clicks of user $i$ and item $j$ and $E_{ij} = (\sum_j N_{ij})(\sum_i N_{ij})/(\sum_{ij} N_{ij})$. The sparse polarity score is then calculated as

$$sp(u_i, e_j) = \max\{\log(N_{ij}/E_{ij})\sqrt{E_{ij}}, 0\}.$$

Similarly, document profile is a bag of entities and categories extracted from the article by an in-house entity extraction tool. $\{(e_0, ab_0), (e_1, ab_1, \cdots, (e_N, ab_N), (c_0, ab_0), (c_1, ab_1), \cdots, (c_M, ab_M)\}$ denotes a document profile with $N$ entities and $M$ categories. The tool extracts entities from the article and assigns a score $ab$ to entities to represent aboutness of the entity to the article. All entities are defined in Wikipedia. The in-house entity extraction tool is a dictionary-based plus a context-based entity disambiguation. It employs a machine learned model with features from contextual phrases and meta features such as length of the article and number of occurrences of entities. Description of the model is out of scope for this paper. Related references can be found in [9]. In addition to entities, there are bags of semantic categories tagged by an in-house document classifier. The classifier defines more than 1,000 categories that include broad ones such as Sports and Finance and narrow ones like NBA, NFL, et al. Each category has an aboutness score to denote its closeness to the document. The document classifier forms the initial step to the stream classifier.

## 3.3 Phase 0: Stream Classifier

In the past the classification task for semantic streams was trivialized to the manual selection of logical filter criteria based on a basket of pre-engineered in-house topical features, or extracted wiki entities, along with their aboutness scores. This approach has the obvious drawback that not all classification criteria can be compactly summarized by a handful (certainly fewer than 100) of logical conjunctives and disjunctives. In fact, since many of the pre-constructed features were not motivated by the stream classification tasks that we are dealing with, chances are they might be completely irrelevant or inaccurate in capturing the essence of the stream definitions. Thus we propose to learn the filter directly from the textual bodies of the articles themselves. In theory at least, this would learn a model that's at least as accurate as the ones obtained from the aforementioned logical patchwork. In reality, however, since logical connectives require higher order non-linear features to exactly replicate, the cost of training an identical model based on token features could be prohibitively high. As we will show in the experimental results section below, however, even with simple unigram model, the precision-recall of a freshly trained model well outperforms the ones based on logical filters constructed by domain experts. Even for a stream as clearly defined as NFL, brute force human filter construction reveals that there are many edge cases (e.g. when both college football players and NFL players appear in the article, or when Super Bowl is mentioned along with other local news) that must be handled with specialized rules, which together can easily go into hundreds of terminal decision nodes.

As a baseline, we also consider the term-frequency inverse-document-frequency (TFIDF) approach [19], whereby a centroid vector is constructed for each stream based on the number of appearances of each token stem in the stream, weighted by the inverse of the logarithm of the total number of documents in which it appears:

$$C_{t_i}^{\mathcal{S}} = |\{k \in [\dim \mathcal{S}] : \mathcal{S}_k = t_i\}| / \log |\{d \in \mathcal{D} : t_i \in d\}|.$$

Here $\mathcal{S}$ stands for the concatenated vector of all tokens from the training set articles in the stream, $C^{\mathcal{S}}$ stands for the centroid vector for the stream represented by $\mathcal{S}$, and $\mathcal{D}$ is the universal set of all documents.

TFIDF is appealing in its dependency mainly on the positive examples; one could think of the entire universe of content pool as representing the negative examples, however the accuracy of the negative labels in this sense is not strictly enforced. More sophisticated variants of TFIDF exist such as instead of a single centroid vector, using a generative model to learn a set of vectors and compute distance of a new candidate document to each of those vectors as a high-dimensional similarity metric. However we feel that the potential gain on top of a simple linear classifier approach does not quite justify the increased model complexity.

Finally we also mention that there are several one-class classification strategies, such as those based on density estimation, clustering, and kernel methods [22], but again the increased model complexity can be a strong hurdle to execution, and the fact that negative examples are not leveraged at all makes them less than ideal.

### 3.3.1 Data Collection

Independent of the model choice, it is important for us to choose a set of positive and negative examples as our initial training set. Due to the large quantity of articles and variety of topics in each stream, as well as the sheer number of streams we are dealing with (ultimately in the thousands), complete reliance on editorial judgment is clearly infeasible. Instead we opt for the following heuristic approach of labeling the articles as to their appropriateness for a particular stream.

For the streams already in production, we have a natural signal for stream appropriateness label, namely the user click feedback. Thus we label all the articles surfaced on the stream, that received clicks from at least 5 unique users, as the positive examples. The assumption is that users who visit a particular stream are most likely interested in content pertaining to the stream definition. Having multiple users interested in a document is a good sign that the article is not just interesting from a general point of view, but is also relevant to the stream. Thus we expect high precision of such labeled set but low recall, as many articles never see the light of day due to low popularity score, which is part of the production ranking criterion, and we are also potentially missing out on the articles that the production filter fails to capture.

Because of the low-recall concern with the above approach, we need to be somewhat careful in selecting the negative examples. In general we want to avoid false negatives at the expense of true negatives, since there are typically many more negative examples than positive examples. The most scalable solution turns out to be including all articles that do not pass the initial hand-crafted logical filter. We do not include the low click count articles as either negatives or positives since they could be unpopular due to either irrelevance or low quality.

For streams not yet launched, clearly domain expertise is required to initialize the classifier. Fortunately we have a variety of well-polished features, such as topical categories, tagged wiki entity, etc, that can be combined through logical connectives to yield an approximate profile for the stream. Once that gets launched and starts receiving user clicks, subsequent batch active-learning cycles will be able to learn the model and fine-tune it to suit the user interests. There is always the risk of flooding a specialized stream with generally popular topics (such as Kim Kardashan in the Sports stream or MH370 in the Finance stream), if we base our model evolution solely on user responses. Furthermore, the initial logical filter can be susceptible to oversight and oversimplification. Thus we periodically inject periodic editorial judgment into the positive and negative training set as well, in a balanced manner to account for the relative lack of editorial resource compared to user signals. Other active learning aspects of the model updating will be explored in future work.

### 3.3.2 Feature Selection

We compared the relative performance of 6 distinct families of features, listed below:

- title and body token stems and frequency counts
- in-house topic categories and aboutness scores
- wiki entities and aboutness scores
- named entities and aboutness scores
- editorial tags as binary features
- publisher ids as binary features

After extensive experimental comparison, we found that while the topic category and publisher id together achieve high performance on testing, it could be that we are simply relearning the filter-based model, which depends to a great extent on the topic categories. On the other hand, token stem frequencies alone perform remarkably well on validation set compared to all other combination of features from the above list. The performance on testing is markedly worse than using the pre-engineered features, as explained above. Since our ultimate goal is to learn the validated labels, and that tokens are primitive features, unadulterated with intermediate processing, we choose them for all streams.

### 3.3.3 Model Selection

Equipped with labeled positive and negative examples and their respective unigram token features, we are in the standard setting of a binary classification task. The combined training and test set size is about 50k for each top level stream (such as Finance and Sports) and 10k for secondary stream, with NFL being the only example considered at this level. The split between training and testing set is 70% to 30%. The ratio of negative to positive examples is typically in the range of 3 to 5 for top streams, but can get much wider for lower level streams. During training we artificially inflate the positive examples to ensure equal weighting.

In addition, we prepared a validation set of about 3,000 articles for the top level streams for editorial labels. These will not only form a validation golden set for the initial model training, but will be recycled for future model updating with appropriate weight that decays with time.

Three model performance metrics are thus considered (training, testing, and validation), each of which is based on precision-recall AUC. For secondary streams such as NFL, we found that a one against all (OAA) approach results in significantly worse metrics than the top stream. This is most likely due to the severe imbalance between positive and negative examples. Fortunately since NFL belongs to Sports as a substream, we could filter out majority of the negative examples that do not pass the Sports filter. This seems a generally applicable strategy despite the fact that there are occasional articles that belong to the lower stream but not to the top.

To continue with the Sports-NFL example, we currently use the rule-based Sports filter as the pre-screening device for NFL articles. In the future, when the Sports filter is replaced by the trained classifier model, we are afforded more flexibility in adjusting the pre-screening strength. How to select the pre-secreening parent classifier threshold ties intimately with the ultimate precision-recall requirement.

Finally to determine the exact classifier form, we found that the performances of logistic loss and hinge loss (SVM) are comparable. However since our preferred training tool, Vowpal Wabbit [2] with L-BFGS optimizer, supports logistic regression far better than SVM (as hinge loss is non-differentiable), we settled upon logistic regression. Some further exploration reveals that a ridge regularization parameter of 10 works well for all three streams tested, and on both testing and validation sets.

## 3.4 Phase 1: Twisted Dot Product

As alluded to before, the classifier instrumented at Phase 0 is not perfect, in particular we do not anticipate 100% preci-

[2] https://github.com/JohnLangford/vowpal_wabbit/wiki

sion. Thus the burden is on subsequent phases to remove any embarassing candidate article for each stream. The Stream Relevance Query (SRQ) proposed at Phase 1 provides the second line of defense against the irrelevant articles. Furthermore it can be viewed as an optimization step for the more serious ranking done at Phase 2.

In a nutshell, SRQ is a stream-enhanced version of the following coarse ranking model applied to the entire content pool:

$$\mathrm{score}_1 = (\alpha_0 \mathrm{gmp} + \alpha_1 \langle \mathbf{u}, \mathbf{d} \rangle) \mathrm{e}^{-\beta \Delta \mathrm{T}}. \quad (1)$$

Here $\mathbf{u}$ denotes the user profile vector, consisting of the sparse polarity scores of all the topical categories and wiki entities, in a sparse format, since a typical user exhibits above average interest in only a couple dozens out of hundreds of thousands of entities. The vector $\mathbf{d}$ consists of document features, expressed in the units of aboutness score. gmp is a popularity score determined essentially by CTR of the article. $\Delta T$ is the time since publishing of the article, which captures the freshness of the article. Thus the exponential factor promotes more recent articles. The weights $\alpha_0, \alpha_1$, and $\beta$ are either trained offline or tuned with split online buckets.

The originally proposed SRQ recipe takes the following form:

$$\mathrm{score}_{\mathrm{SRQ}} = (\alpha_0 \mathrm{gmp} + \alpha_1 \mathbf{q}_0 \cdot \mathbf{d} + \alpha_2 \mathbf{q}_1 \circ \mathbf{u} \cdot \mathbf{d}) \mathrm{e}^{-\beta \Delta \mathrm{T}}. \quad (2)$$

$\mathbf{q}_0$ and $\mathbf{q}_1$ are stream-dedicated feature vectors corresponding to document and user respectively. These are trained using real bucket user click data, before the other weights $\alpha_i$ are learned.

Thus we replace the original dot product $\alpha_1 \mathbf{u} \cdot \mathbf{d}$ with a twisted (triple) dot product $\alpha_1 \mathbf{q}_0 \cdot \mathbf{d} + \alpha_2 \mathbf{q}_1 \circ \mathbf{u} \cdot \mathbf{d}$. Alternatively, one can think of the twisted product as a modification to the user profile $\mathbf{u} \mapsto \mathbf{u}' := \alpha_1 \mathbf{q}_0 + \alpha_2 \mathbf{q}_1 \circ \mathbf{u}$. This admits tremendous flexibility in relevance-biased personalization; each user thus can put on many different "faces" when interacting with the different streams.

One caveat to the above enhancement is that the new stream-specific user profile $\mathbf{u}'$ can now have hundreds of thousands of nonzero entries, making it non-sparse. On the other hand, the Phase 1 relies heavily on the sparsity of the user feature vector to optimize for latency, essentially by dropping candidate articles that do not match the user vector on any entity (hence results in 0 dot product), unless $\mathbf{u}$ is too sparse, in which case gmp dominates and the top ranked articles by popularity get selected. Having a dense $\mathbf{u}'$ makes it significantly harder to implement such optimization heuristics. So we instead consider the following practical SRQ model:

$$\mathrm{score}_{\mathrm{pSRQ}} = (\alpha_0 \mathrm{gmp} + \alpha_1 \mathbf{q} \circ \mathbf{u} \cdot \mathbf{d}) \mathrm{e}^{-\beta \Delta \mathrm{T}}. \quad (3)$$

The main difference is the removal of the old-fashioned dot-product term. One could argue non-rigorously, that the Phase 0 filter is essentially performing the same job as the zeroth order term $\alpha_1 \mathbf{q}_0 \cdot \mathbf{d}$ in the original SRQ model.

Other variations that do not completely lose the document-only signals include the following indicator model:

$$\mathrm{score}_{\mathrm{iSRQ}} = (\alpha_0 \mathrm{gmp} + \alpha_1 \mathbf{q}_0 \circ \chi(\mathbf{u}) \cdot \mathbf{d} + \alpha_2 \mathbf{q}_1 \circ \mathbf{u} \cdot \mathbf{d}) \mathrm{e}^{-\beta \Delta \mathrm{T}},$$

where $\chi(\mathbf{u}) = (1_{u_1 > 0}, \ldots, 1_{u_m > 0})$ is the component-wise indicator vector for $\mathbf{u}$. Under iSRQ, the user vector $\mathbf{u}$ gets transformed into an equally sparse tilted $\mathbf{u}' = \alpha_1 \mathbf{q}_0 \circ \chi(\mathbf{u}) + \alpha_2 \mathbf{q}_1 \circ \mathbf{u}$ that closely resembles the original SRQ, $\mathbf{u}'$.

Regardless of the actual scoring formula used, the output of Phase 1 ranking is a set of around 200 articles, to be ranked further by Phase 2 GBDT machinery. This reduction from thousands of articles facilitates highly sophisticated algorithms to be applied for accurate personalized ranking.

## 3.5 Phase 2: Boosted Ranking

Phase 1 performs a coarse ranking where a linear scoring function is applied to ensure fast and high recall article selection. Phase 1 passes 200 articles to Phase 2, which uses a more complicated ranking model to improve high quality. The number 200 was selected according to Phase 2 complexity and system latency constraints.

Phase 2 ranking model integrates various type of features. Due to the importance of recency in news recommendation, it is the first issue to consider. By assumption, users prefer fresh contents. A rankig model with a time sensitive feature can promote recency. However, it has been observed that promoting recency can cause relevant documents to be ranked lower. Some solutions are proposed in [7]. Document age is the most available indicator of the recency of the article content. Thus we use it as a feature in phase 2.

Article popularity *gmp* is also used as a feature, as described in section 3.2. *gmp* is not stream dependent, but is calculated by using user actions on the Top Stream, as shown in Fig. 1 because the Top Stream has larger user traffic to derive reliable CTR.

Lastly, content based features are used. These are extracted from document profile explained in section 3.2, and include total numbers of entities and categories, sum of aboutness scores and other arithmetic transformations of aboutness scores. Similar features are generated based on user profile such as number of user inferred entities and categories, sum of sparse polarity scores, and related arithmetic transformations of sparse polarity scores.

Correlations between the user profile and the document profile give rise to a few more features. Other than a single dot product used in Phase 1 that combines entities and categories, Phase 2 separates features from entity and from category. For example, features may include the number of matched entities between the user and the document, the number of matched categories, dot product of sparse polarity and aboutness scores between entities, dot product between categories, and the total dot product between all entities and categories.

A very special feature that is unique for our system is the identity of the stream, or stream-id. This is a categorical feature. The feature defines which stream the article belongs to. In the hiearchical structure Fig. 1, each node is assigned an identification number. All articles in this stream have the same ID. The feature is assigned values according to its source. If it is from NFL, then stream-id="nfl". We will show in the experiments this feature is very useful.

The training is based on clicks. To collect training data, we set up a data collection bucket to get online users' response to recommended articles. The articles are presented to a user in the format of a scrollable stream. Only title and a short summary are shown. The user can scroll the articles in the stream until she finds an interesting one. Then she may click the article and goes to the landing page to read the complete article. A click is a positive training example, which proves the article is worth recommending. On the contrary, skipped articles are those that are not clicked but are displayed above any clicked articles in a single stream session. Skipped articles are viewed but not clicked. They form our negative examples. For those articles below clicked articles, we do not know if the user viewed them or not because the user had no action on these articles. These articles were not used in training.

To avoid ranking bias and position bias, we used an exploration bucket (not exploitation bucket) [15] . Articles in the exploration bucket were ordered not based on any feature and model but randomly.

Our training data generation is formalized as follows. If user $i$ clicks some articles in the stream, we generate training examples, $(x_0^i, y_0^i, w_0^i), (x_1^i, y_1^i, w_1^i), \cdots, (x_j^i, y_j^i, w_j^i), \cdots, (x_N^i, y_N^i, w_N^i)$. Each example consists of triplet produced by user $i$ on article $j$. $x_j^i$ corresponds to feature vector of the $j$th article and the user $i$ . $y_j^i$ is a $\{0, 1\}$ binary-valued label. If $j$ is a clicked article, then $y_j^i = 1$. It is 0 otherwise. $w_j^i$ is the weight of the training example, whose value is equal to one plus the logarithm of the dwell-time of the click event. The latter is defined to be the time spent reading the full article $j$, that is, the duration from when user $i$ clicks the article to when he leaves the article. Only clicked articles have dwell-time. $w_j^i = 1$ for skipped articles. $N$ is the index of the last clicked article. The total article number is 200, but in reality $N \leq 200$ due to discarding non-viewed articles from training.

The feature vector $x_j^i$ is defined by

$$x_j^i = (age, gmp, streamid, u_i, d_j, u_i \circ d_j)$$

.

Here, $age$ denotes document age. $u_i$ and $d_i$ denotes a few features generated from user and document profiles respectively. $u_i \circ d_j$ are correlative features between the two.

Given training examples $(x_j^i, y_j^i, w_j^i), i \in (1, M), j \in (1, N_i)$, $M$ the total number of users and $N_i$ the number of samples for user $i$, we can apply multiple machine learning methods to learn a regression model. We employ Gradient Boosted Decision Tree (GBDT) algorithm [10, 25] as the learning method of choice. This algorithm has higher precision, not sensitive to over-training, and supports category features, for instance, stream-id. GBDT is an additive regression algorithm consisting of an ensemble of trees, fitted to current residuals, gradients of the loss function, in a forward step-wise manner. It iteratively fits an additive model as

$$f_t(x) = T_t(x; \Theta) + \lambda \sum_{t=1}^{T} \beta_t T_t(x; \Theta_t)$$

such that certain loss function $L(y_i, f_T(x+i))$ is minimized, where $T_t(x; \Theta_t)$ is a tree at iteration $t$, weighted by parameter $\beta_t$, with a finite number of parameters, $\Theta_t$; $\lambda$ is the learning rate. At iteration $t$, tree $T_t(x; \beta)$ is induced to fit the negative gradient by least squares.
The optimal weights of trees $\beta_t$ are determined by

$$\beta_t = \text{argmin}_\beta \sum_{i}^{N} L(y_i, f_{t-1}(x_i) + \beta T(x_i, \theta))$$

# 4. EXPERIMENT RESULTS

## 4.1 Evaluation Metrics

For phase 0 stream filter, we use precision-recall Area Under the Curve (AUC) as the principle measure of performance. The labels are generated using either heuristic methodology based on production filter in combination with unique user click statistics, or editorial judgments. The latter data is sparse but has high fidelity. AUC of 1 corresponds to the perfect situation where all the positively labeled articles are ranked higher than the negatively labeled ones. This does not mean that a threshold is not necessary, but only that a perfect threshold is feasible, and that for any threshold level, the total number of misclassifications is optimal (minimal) among all rankings.

For phase 1 and 2, we use primarily AUC under the ROC curve, based on users' click feedback. Thus a clicked article $d$ by a user $u$ is considered a positive example, whereas the rest are negative examples. This is an appropriate metric here because the end goal here is ranking. AUC for ROC can be seen as a specialization of a family of permutation statistics, with notable examples such as Kendall's tau, that measures the number of pairs of records whose relative order is reversed compared to the true ranking. It is also conveniently agnostic to the ratio between the numbers of positive and negative labels.

ROC curve is always monotone non-decreasing, which is easily seen by rewriting $t_i = \frac{1}{1+\mathrm{FN_i}/\mathrm{TP_i}}$ and $f_i = \frac{1}{1+\mathrm{TN_i}/\mathrm{FP_i}}$, and noticing the subfractions are both monotone in $i$. Here $\mathrm{TN_i}$ stands for the number of examples with negative labels and negative model prediction score (true negatives) having rank at least $i$. Similarly $\mathrm{FP_i}$ stands for the number of false positives in the top $i$ items. The PR curve however can be highly non-montone in general, since the quantity $\mathrm{FP_i}/\mathrm{TP_i}$ is not monotone in general. An easy example is given by $n-1$ positively labeled data and a single negatively labeled one: in this case the curve starts off with $p_i = 1$ until the index reaches the negatively labeled datum, at which point the curve drops to $\frac{i-1}{i}$, which however is an increasing function in $i$. In a sense, the more monotone the PR curve, the better it is from predicting completely randomly.

It is important also to note that while the AUC for ROC curve always has a mean of 0.5 under uniform ranking, easily proved using the equivalence with Mann-Whitney-Wilcoxon statistics, provided there is at least one positive example and one negative example, the AUC for the PR curve has a uniform baseline distribution that depends on the exact ratio between positive and negative examples. See [6] for a comparative study of Precision-Recall and ROC curves. Again with the example of one negatively labeled datum, where the predicted rank of the negative example is uniformly random, then using flat extrapolation to the left, the expected AUC for PR curve is given by

$$\frac{1}{n} \sum_{k=1}^{n} [\frac{k}{n} + \sum_{i=k+1}^{n} \frac{i-1}{in}] = 1 + o(1).$$

In general it's easy to show (by considering pointwise average) that the average AUC is given by $P/n$, where $P$ is the total number of positively labeled examples. This is especially important for the stream filter evaluation, since there tends to be many more negative examples than positive ones.

## 4.2 Test Results

Table 1: Stream classifier training and test data

| property | training size (+/-) | testing size (+/-) |
|---|---|---|
| Finance | 5410/33890 | 2287/14524 |
| Sports | 6569/34947 | 2821/14972 |
| NFL | 1518/5054 | 651/2167 |

Table 2: Stream classifier precision-recall

| property | labels (+/-) | rule prec/rec | class. prec/rec |
|---|---|---|---|
| Sports | 485/394 | 97%/44% | 99%/51% |
| Finance | 396/86 | 95%/35% | 98%/50% |
| NFL | 350/326 | 92%/59% | 92%/89% |

### *4.2.1 Phase 0*

The precision-recall curves for the three streams (Sports, Finance, and NFL) under linear classifier and tfidf approaches are shown in Figures 2. The table 1 summarizes the training and test data used:

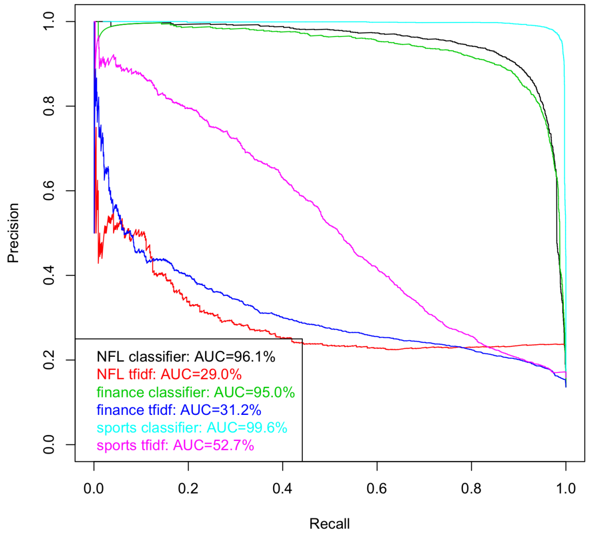
Both training and test data are labeled according to the heuristic rule based on user clicks. For the linear classifier we use threshold at 0, since the negative and positive examples are weighted equally.

The table 2 compares the point-wise precision-recall for smaller editorially labelled datasets, with the numbers of positive and negative examples chosen proportional to real world data. The third column shows the performance of rule-based classifier, compared against the precision-recall of the classifier at a single threshold 0.

For all three streams, we are able to maintain or improve precision while greatly increase recall at a single threshold level. Since the training data is balanced, 0 threshold makes perfect sense.

The performance of tfidf-based classification is clearly inferior to logistic classifier.

Figure 2: Precision-Recall Curves



### *4.2.2 Phase 1*

Table 3: Phase 1 multiple models

| Method | Sports AUC | Finance AUC |
|---|---|---|
| gmp only | 0.58 | 0.62 |
| flat + gmp | 0.61 | 0.61 |
| SRQ + gmp | 0.62 | 0.63 |
| SRQ prune + gmp | 0.64 | 0.65 |

Figure 3: Sports ROC

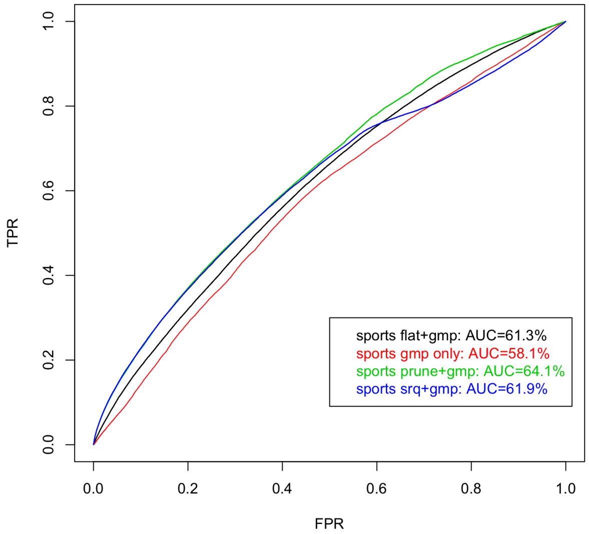
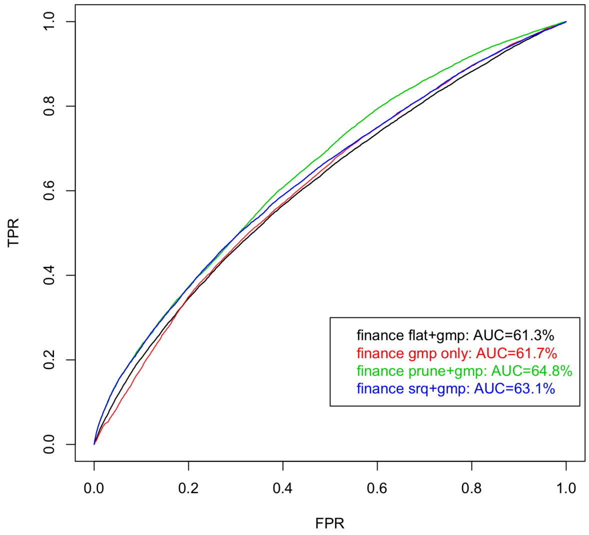


Figure 4: Finance ROC



Here we present the ROC AUC for Sports and Finance under four sets of training methodologies (Table 4.2.2). The ROC curves for the two streams are displayed in Figure 3 and Figure 4. The weights for the consolidated components (gmp, $\langle \mathbf{u}, \mathbf{q} \rangle$, etc) are optimized a second time with respect to the training set. It is interesting to observe that the SRQ with negative regressor pruning results in an ROC curve that dominates all other models in both cases. We modified the exponential recency in the original dot product model in

Table 4: Phase 2 training and test data distribution

|  | training |  |  | test |  |  |
|---|---|---|---|---|---|---|
| property | click | view | total | click | view | total |
| Sports | 108K | 1.5M | 1.6M | 12K | 178K | 191K |
| NFL | 20K | 328K | 347K | 1.5K | 21K | 22K |
| NBA | 7K | 66K | 72K | 649 | 8K | 9K |
| MLB | 7K | 9K | 95K | 971 | 14K | 15K |
| Finance | 57K | 779K | 836K | 5K | 73K | 79K |
| News | 109K | 1.4M | 1.5M | 14K | 207K | 221K |

Table 5: Comparision of Phase 2 ranking models

|  | Sports | Finance | News |
|---|---|---|---|
| Base (linear phase1 feature) | 0.56 | 0.53 | 0.56 |
| GBDT(Phase 1 feature) | 0.59 | 0.62 | 0.58 |
| GBDT(Phase 2 feature) | 0.64 | 0.66 | 0.62 |

favor of a completely linear one, to achieve optimal ranking performance.

1. gmp only: score = $\alpha \text{gmp} + \beta \Delta \text{T}$;
2. flat dot-product (no SRQ) + gmp (baseline): score = $\alpha_0 \text{gmp} + \alpha_1 \langle \mathbf{u}, \mathbf{d} \rangle + \beta \Delta \text{T}$;
3. SRQ + gmp: score = $\alpha_0 \text{gmp} + \alpha_1 \langle \mathbf{q} \circ \mathbf{u}, \mathbf{d} \rangle + \beta \Delta \text{T}$;
4. SRQ with negative pruning + gmp: score = $\alpha_0 \text{gmp} + \alpha_1 \langle \widetilde{\mathbf{q}} \circ \mathbf{u}, \mathbf{d} \rangle + \beta \Delta \text{T}$, where $\widetilde{\mathbf{q}}_i = 0$ if $\mathbf{q}_i \leq 0$.

### 4.2.3 phase 2

We build ranking models for Sports, Finance and News stream respectively, collecting two weeks of user action data from these streams. These data are used as training. Independently, We also collect one more week data for testing. The method of data collection has been described in Section 3.5. Sports, Finance and News are big streams. Under them there are small streams such as NFL,NBA,MLB. Data distribution are shown in Table. 4. The table shows the number of clicks and views for different properties in our training and test data.

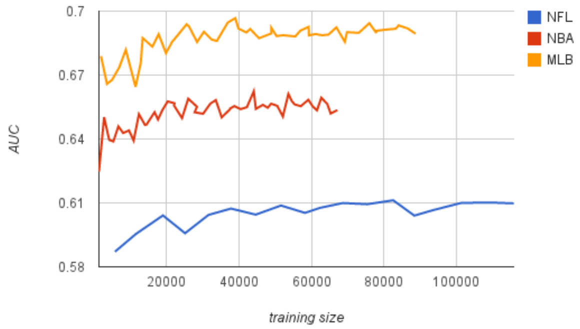
GBDT is our learning method of choice. The comparisions between GBDT and linear regression under different feature size are shown in Table 5. We take as our baseline model linear regression with features from Phase 1 (similar to ”flat gmp” of Table 3). Note experiments on Phase 1 and Phase2 used distinct training set and test set. Hence, the numbers shown on Table 5 and Table 3 are not aligned. Two GBDT model results are shown here. One uses phase 1 features and the other uses a total of 14 Phase 2 features as described in Section 3.5. The same algorithm is applied to three distinct top-level properties: Sports, Finance and News. The linear baseline models gave the worst results. The GBDT with Phase 2 features yields the best results. We use Receiver Operating Chracteristics Area Under the Curve (AUC) as the principle metric (see section 4.1).

Intuitively, one would assume that better results could be derived if each stream has its own model. Actually, we found this assumption was right if the streams are very different in content category such as Sports, Finance and News. We regard these streams as heteogenerous streams. Our study found we should build a separate model each for heteogenerous streams. But this assumption no longer holds for relatively homogeneous steams such as NFL,NBA and MLB, all of which are sub-streams of Sports. To show this, we trained a general model using all sports data, and a dedicated model for each sub-stream using only the data from each sub-stream. Accordingly, we test the models on test data from different streams. The results are shown in Table 6.

Table 6: Substream evaluations

| test data<br>training data | Sports | NFL | NBA | MLB |
|---|---|---|---|---|
| Sports | 0.636 | 0.629 | 0.662 | 0.705 |
| NFL | - | 0.610 | - | - |
| NBA | - | - | 0.645 |  |
| MLB | - | - | - | 0.676 |
| with stream-id | 0.642 | 0.635 | 0.689 | 0.725 |

Figure 5: Test data metrics as increasing sub-stream training data size



We found that the model trained on all sports data performed better than dedicated models. We also found the poor results brought by the dedicated models were not due to small size of training data. Actually, the training data sizes we used for sub-stream models are sufficient to achieve maximal performance. Fig. 5 is training curves which shows the trend of AUC change on the same test set as increasing training data size. The training data size is large enough to converge to the maximal results, which proves increaing more training data size does not help. We interpret this to mean that using all data can have rich feature distributions while feature distributions are biased for sub-streams.

We found that using stream-id as feature can improve substream ranking. This is a categorical feature for multiple stream ranking. Each decision tree in the GBDT ensemble can be split at node with logic such as stream-id $\in \{NFL, NBA\}$. The model with these features is trained on all sports data. The results are even better than using all sports data. They are shown on the last line of Table 6. Ranking metrics are improved for all sub-streams with this additional feature.

In addition to personalized relevance, ranking should also promote fresh contents. Indeed, if recency is not specifically

Table 7: Recency evaluation

| | AUC | age@top1 |
|---|---|---|
| Sports Base (Phase2 gbdt) | 0.64 | 10.3 |
| Sports (expontial age decay) | 0.58 | 7.5 |
| Sports (age as feature) | 0.66 | 8.3 |
| Finance Base | 0.66 | 6.6 |
| Finance (expontial age decay) | 0.60 | 4.6 |
| Finance (age as feature) | 0.68 | 6.4 |

Table 8: bucket test

| | CTR | Dwell time |
|---|---|---|
| Base (Linear) | - | - |
| Sports,GBDT(phase 1 feature) | +6% | +6% |
| Sports,GBDT(Phase 2 feature) | +25% | +21% |
| Finance,GBDT(phase 1 feature) | +3% | +3% |
| Finance,GBDT(Phase 2 feature) | +20% | +18% |

treated, many old articles would be placed towards the top of the streams because old articles can accumulate higher popularity scores. Usually, there are two methods in handling recency. First, as used in Phase 1, recency can be boosted by applying an exponential age decay factor. Old age documents are punished. Related works can be found in [16, 18]. A second option is to use document age as a feature. Recency feature is integrated with other features. The relative importance of recency and relevance feature is completely determined by the training data and the algorithm.

We tested both methods, with results shown in table 7. Here we take as the baseline model the best ranking model from the previous expriments, namely, GBDT with 14 phase-2 features.

"age@top1" is the average document age (hour) of the first articles in the stream over all sessions. A session is the period from when a user comes to the site to when she leaves, or 30 minutes, whichever is shorter. For example, "age@top1=10.3" means that the average age of the top 1 articles ranked by the Sports baseline model in all sessions is 10.3 hours. The model "expontial age decay" discounts the baseline ranking score exponentially by document age, similar to the exponential multiplier $e^{\beta\Delta T}$ as in Eq. (3). The model "age as feature" used age as a feature in GBDT. We found that using document age as a feature achieves better results than using expontial age decay: both AUC (relevance metric) and age@top1 are improved while expontial age decay results in a big relevance drop in terms of AUC. The results are consistent for both Sports and Finance streams.

We also did online user experience test (bucket test) to evaluate GBDT models (Table 8). We were able to do bucket test for Sports and Finance except News at the time. The models were compared to a linear ranking model (as Phase 1). Both CTR and Dwelltime improved by more than double digits.

# 5. CONCLUSIONS

This paper describes a system to recommend contents for multiple semantic streams. We present a semantic stream hierarchy to guide internet users to browse contents and help them find personally interesting articles more easily and quickly. To solve the content recommendation problem for multiple semantic streams at various depths, we face unique challenges such as scalable multi-layer stream classification and stream ranking.

Several new methods are proposed such as click-based stream classifier and using stream-id to improve multiple stream ranking. All these methods are found effective and improved upon baseline model significantly.

The whole system is decomposed into three phases: phase 0 as a multi-stream classifier and phase 1 and phase 2 as ranking models. Phase 0 can arguably be combined with phase 1, since both have linear forms. This is the spirit of model Eq. (2). In practice, however, we find the latency unacceptable when applying the combined algorithm at serving time, since document profiles can have significantly more features than users. Instead, phase 0 takes away some of the burden in Eq. (2) by computing the document-only component scores during content ingestion time, and a simplified phase 1 model Eq. (3) is adopted for fast document retrieval.

We demonstrate order of magnitude improvement in precision-recall of the linear classifier over tfidf approach. Compared with hand-crafted rule-based approach, the classifier significantly increases recall while maintaining the same or better precision. For phase 1 personalized stream-specific linear ranking, we measure the performance in terms of click/skip ROC curves and show an increase in Area Under the Curve (AUC) of 3-4% under model Eq. (3) over the stream-independent model. Finally phase 2 refines the ranking results from phase 1 by taking into account a select set of 14 features related to user, document, and their interactions. We observe huge CTR and dwell-time gains over the baseline from online user experience test.

To summarize, we design a highly efficient, scalable stream content recommendation system that optimizes user experience measured in terms of CTR, by seeking a balance among the following four desirable attributes: relevance (to the individual stream), popularity, freshness, and personalization.

In practice, not all 3 phases are applied to all streams uniformly. Some small streams have few articles, or may not need personalization from Phase 1 or Phase 2. It is flexible to adjust the 3 phases for different steams. This will be examined in an upcoming work.

# 6. REFERENCES

[1] D. Agarwal. Recommending items to users: An explore/exploit perspective. In *Proceedings of the 1st Workshop on User Engagement Optimization*, UEO '13, pages 1–2, New York, NY, USA, 2013. ACM.

[2] D. Agarwal, B.-C. Chen, P. Elango, and R. Ramakrishnan. Content recommendation on web portals. *Commun. ACM*, 56(6):92–101, June 2013.

[3] P. Calado, M. Cristo, M. A. Gonçalves, E. S. de Moura, B. Ribeiro-Neto, and N. Ziviani. Link-based similarity measures for the classification of web documents. *J. Am. Soc. Inf. Sci. Technol.*, 57(2):208–221, Jan. 2006.

[4] P. Calado, M. Cristo, E. Moura, N. Ziviani, B. Ribeiro-Neto, and M. A. Gonçalves. Combining link-based and content-based methods for web document classification. In *Proceedings of the Twelfth International Conference on Information and Knowledge Management*, CIKM '03, pages 394–401, New York, NY, USA, 2003. ACM.

[5] A. S. Das, M. Datar, A. Garg, and S. Rajaram. Google news personalization: Scalable online collaborative filtering. In *Proceedings of the 16th International Conference on World Wide Web*, WWW '07, pages 271–280, New York, NY, USA, 2007. ACM.

[6] J. Davis and M. Goadrich. The relationship between precision-recall and roc curves. In *Proceedings of the 23rd international conference on Machine learning*, pages 233–240. ACM, 2006.

[7] A. Dong, Y. Chang, Z. Zheng, G. Mishne, J. Bai, R. Zhang, K. Buchner, C. Liao, and F. Diaz. Towards recency ranking in web search. In *Proceedings of the Third ACM International Conference on Web Search and Data Mining*, WSDM '10, pages 11–20, New York, NY, USA, 2010. ACM.

[8] R.-E. Fan, K.-W. Chang, C.-J. Hsieh, X.-R. Wang, and C.-J. Lin. Liblinear: A library for large linear classification. *The Journal of Machine Learning Research*, 9:1871–1874, 2008.

[9] J. R. Finkel, T. Grenager, and C. Manning. Incorporating non-local information into information extraction systems by gibbs sampling. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, ACL '05, pages 363–370, Stroudsburg, PA, USA, 2005. Association for Computational Linguistics.

[10] J. H. Friedman. Stochastic gradient boosting. *Comput. Stat. Data Anal.*, 38(4):367–378, Feb. 2002.

[11] F. Garcin, K. Zhou, B. Faltings, and V. Schickel. Personalized news recommendation based on collaborative filtering. In *Proceedings of the The 2012 IEEE/WIC/ACM International Joint Conferences on Web Intelligence and Intelligent Agent Technology - Volume 01*, WI-IAT '12, pages 437–441, Washington, DC, USA, 2012. IEEE Computer Society.

[12] L. Getoor, E. Segal, B. Taskar, and D. Koller. Probabilistic models of text and link structure for hypertext classification. In *IJCAI Workshop on Text Learning: Beyond Supervision*, 2001.

[13] T. Hastie, R. Tibshirani, J. Friedman, T. Hastie, J. Friedman, and R. Tibshirani. *The elements of statistical learning*, volume 2. Springer, 2009.

[14] Y. Koren, R. Bell, and C. Volinsky. Matrix factorization techniques for recommender systems. *Computer*, 42(8):30–37, Aug. 2009.

[15] L. Li, W. Chu, J. Langford, and R. E. 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, New York, NY, USA, 2010. ACM.

[16] X. Li and W. B. Croft. Time-based language models. In *Proceedings of the Twelfth International Conference on Information and Knowledge Management*, CIKM '03, pages 469–475, New York, NY, USA, 2003. ACM.

[17] 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, New York, NY, USA, 2010. ACM.

[18] D. Metzler, R. Jones, F. Peng, and R. Zhang. Improving search relevance for implicitly temporal queries. In *Proceedings of the 32Nd International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '09, pages 700–701, New York, NY, USA, 2009. ACM.

[19] J. Ramos. Using tf-idf to determine word relevance in document queries. In *Proceedings of the First Instructional Conference on Machine Learning*, 2003.

[20] S. Rendle, L. Balby Marinho, A. Nanopoulos, and L. Schmidt-Thieme. Learning optimal ranking with tensor factorization for tag recommendation. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 727–736. ACM, 2009.

[21] A. Sun, E.-P. Lim, and W.-K. Ng. Web classification using support vector machine. In *Proceedings of the 4th International Workshop on Web Information and Data Management*, WIDM '02, pages 96–99, New York, NY, USA, 2002. ACM.

[22] D. M. Tax. One-class classification. 2001.

[23] J. Wang, A. P. de Vries, and M. J. T. Reinders. Unifying user-based and item-based collaborative filtering approaches by similarity fusion. In *Proceedings of the 29th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '06, pages 501–508, New York, NY, USA, 2006. ACM.

[24] K. Weinberger, A. Dasgupta, J. Langford, A. Smola, and J. Attenberg. Feature hashing for large scale multitask learning. In *Proceedings of the 26th Annual International Conference on Machine Learning*, pages 1113–1120. ACM, 2009.

[25] J. Ye, J.-H. Chow, J. Chen, and Z. Zheng. Stochastic gradient boosted distributed decision trees. In *Proceedings of the 18th ACM conference on Information and knowledge management*, CIKM '09, pages 2061–2064, New York, NY, USA, 2009. ACM.

[26] Y. Yue, C. Wang, K. El-Arini, and C. Guestrin. Personalized collaborative clustering. In *Proceedings of the 23rd International Conference on World Wide Web*, WWW '14, pages 75–84, Republic and Canton of Geneva, Switzerland, 2014. International World Wide Web Conferences Steering Committee.

[27] N. Zheng, Q. Li, S. Liao, and L. Zhang. Flickr group recommendation based on tensor decomposition. In *Proceedings of the 33rd international ACM SIGIR conference on Research and development in information retrieval*, pages 737–738. ACM, 2010.