Click-Based Content Recommendation for Hierarchical Semantic Streams

ABSTRACT

Keywords

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 everyday, much of which is irrelevant to their personal interest. Thus directory-style news streams organized by categories 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 **??**).

One obvious approach to the hierarchical stream problem is to reduce it back to the problem of a single stream, by implementing a filter for each category node on the content hierarchy. Indeed this will be an important step in the system described below. One outstanding challenge in this direction is the labeling of training examples. Given the number of streams (easily in the 100's) in question, editorial resource becomes limited. We address the problem with a combination of user click feedbacks and editorial labeling.

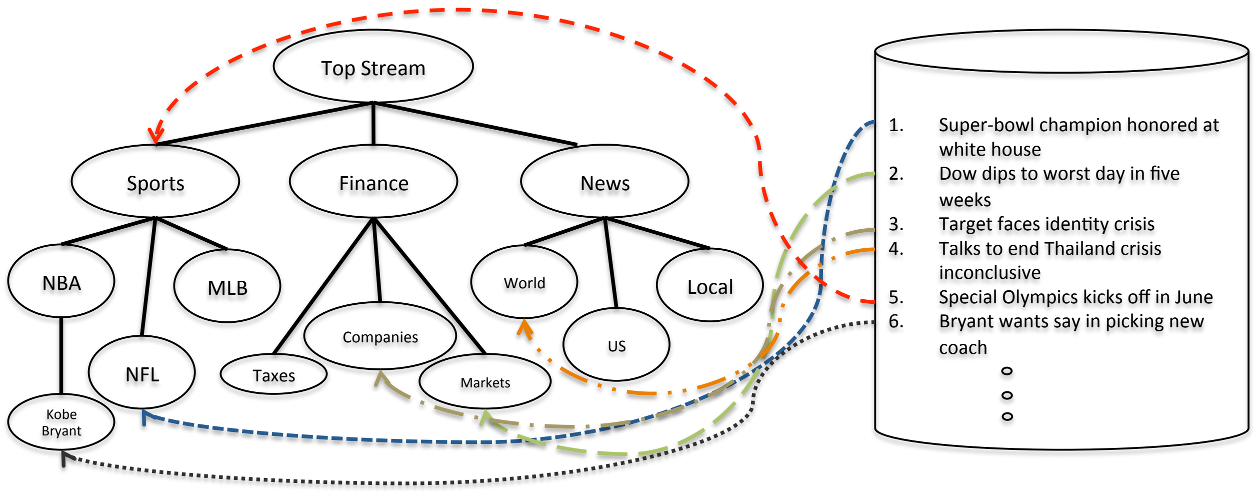
Even after the filter problem is solved, users frequenting different category streams can have very different sets of browsing interest. Furthermore, articles that are popular in a general stream may be less popular in a substream, and vice versa. In short, going to a lower stream category can be viewed as a probabilistic conditioning of the general user and document features distribution. 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 constraint, the latter is further subdivided into three 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 CTR, and finally the freshness of the content. Phase 2 then performs re-ranking of those 200 articles using the more sophisticated Gradient Boosted Decision Tree (GBDT) model. 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.

Phase 0 can arguably be combined with phase 1, since both have linear forms. This is the spirit of model (**??**). In practice, however, we find the latency unacceptable when applying the algorithm during serving time, since document profiles can have significantly more features than users. Instead, phase 0 takes away some of the burden in (**??**) by computing the document-only component scores during content ingestion time, and a simplified phase 1 model (**??**) is adopted for fast document retrieval.

We demonstrate order of magnitude improvement in precision-recall of the linear classifier over tfidf approach. Compared with decision rule based approach, the classifier significantly increases recall while keeping precision the same or better. 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 (**??**) over the stream-independent model. The exact metric definitions are reviewed in Section **??**.

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 five desirable attributes: relevance (to the individual stream), popularity, freshness, personalization, and diversity.


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

2. RELATED WORK

According to [?], there are two main types of recommendation: content-based and collaborate filtering. Content based recommendation is recommending new articles based on the user's reading history. Articles is recommended if they have the same entities or categories as appears before in the user's reading history. This paper's work falls into this category. A second method is collaborate filtering [?, ?, ?, ?]. This approach recommends new articles to the user by virtue of similiar interests with other users that have read the articles. Similar users are grouped by some methods like matrix factorization [?]. If a user has read an article, this article can be recommended to other users within the same groups.

In this paper we propose a machine learned stream classifier based on click. We have not found related works but there are many existing work about web document classifier [?, ?]. Document classifier makes use of information from content page like titles and anchor text or web page link structure. 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 by optimizing clicks. Using click has been proved effective in many fields. There are many works about using click in recommendation and search [?, ?]. Some recent work give more up-to-date improvment on using click [?, ?].

Recency is widely considered in recommendation and Web search. In Facebook's Edgerank algorithm, recency is an important factor. Recency is tackled in [?]'s Web search work. Regarding to recency, one puzzling fact is it has controvery effect in content recency and relevance. Improving recency will cause relevance reduction. Some methods are proposed to ballance the two, like smart boosting and recency query classifier in [?].

3. SYSTEM DESIGN

3.1 Motivations

Designing an actionable system for thousands of article streams with millions of content to present is a highly non-trivial 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, an 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. Finally we want to ensure that similar articles stay far from one another, a task that requires pairwise comparison operations, hence must be addressed separately at the end (phase 3). In general, Having a multi-phase system can be a 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

gmp is a popularity feature. $gmp_{t_i} = \alpha * ctr_{t_i} + \beta * gmp_{t_{i-1}}$, gmp_i is article popularity score at time i. gmp measures the article's click through rate. gmp is timely updated, about every 6 minutes. The latest CTR has a higher weight in the formula.

Each user profile consists of a bag of entities and a bag of semantic categories extracted from user's reading history on the web portal. Within a fixed profile for user i, each entity or category j is assigned a so-called sparse-polarity score u_j to represent the user's level of interests (LOI) to it.The basic idea is similar to TFIDF, in which generally popular entities such as "politics" require many more clicks by user i to reach a noticeably high score, whereas if the user clicked even once on an article about an obscure entity, then his/her profile will register a high score for u_j. In addition, u_j's are floored at 0.

Similarly, a document profile consists of aboutness scores a_j for all entities that appear in the document, adjusted as usual by their inverse document frequencies (IDF). We skip describing details about how the user profile score and document profile score are calculated. There are some references about this work. Document file is a bag of entities extracted from the article by an in-house entity extraction program. Each entity is assigned a score to represent aboutness of this entity to the article. The score is given by a machine learned model taking a few features such as tfidf, entity count. In addition to bag of entities, there are bag of semantic categories tagged by a in-house document classifier. There are more than 1,000 categories defined in the classifier. Some categories include Politics, Sports, Finance, NBA, NFL, et al. For each category is also assigned an aboutness score to the document.

3.3 Phase 0: Stream Classifier

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:

$$\mathrm{gmp} = \frac{\sum_k \lambda^k N_k^{\mathrm{clicks}}}{\sum_k \lambda^k N_k^{\mathrm{clicks}} + N_k^{\mathrm{skips}}},$$

where $\lambda \in (0, 1)$ is a time-decay factor, N_k^{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^{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. The first one, sparse polarity score, quantifies a user's above-average interest in a particular entity. 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

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

Finally the document aboutness score of (u_i, e_j) is computed by an in-house GBDT model with features from contextual phrase as well as global document features such as length of the article and number of occurrences of the entity in the document.

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 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 [?], whereby a centroid vector is constructed for each stream based on the number of appearances of each token stem in the stream, weighted by 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 [?], 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 examples 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 at least 5 unique user clicks, 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 production filter. This is clearly the right thing to do since weakly positive examples (ones that didn't receive enough user clicks) could nonetheless be highly relevant.

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 named 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 over-simplification. Thus we require periodic editorial judgment to be injected 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 aspect 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 production model, which depends almost exclusively on these features. 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 the former, but againt this could be due to noisy labels. 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 testing 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 3000 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. ***** Q:why have this paragraph?

To continue with the Sports-NFL example, we currently use the production 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 [?] 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% precision. 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 generic model applied to the entire content pool:

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

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. ΔT 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

α_0, α_1, and β are either trained offline or tuned with split online buckets.

The originally proposed SRQ recipe takes the following form:

$$\mathrm{score}_{\mathrm{SRQ}} = (\alpha_0 \mathrm{gmp} + \alpha_1 \mathbf{q}_0 \cdot \mathbf{d} + \alpha_2 \mathbf{q}_1 \circ \mathbf{u} \cdot \mathbf{d}) \mathrm{e}^{-\beta \Delta \mathrm{T}}. \quad (1)$$

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 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 Homerun 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:

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

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. Our system passes 200 articles from Phase 1 to Phase 2. Phase 2 is a fine ranking. Because the target is only 200 articles, a slower but higher precision model is feasible, in which more features than Phase 1 can be used.

We consider a few factors in ranking. Recency is an important feature for news recommendation. Users want to read newest contents, especially for frequent users. They want to read fresh contents every time when they visit the website. Recency has been viewed as a tough problem in Web search and recommendation, and some solutions are proposed in [?] where some recency related features are used in ranking model and good results are observed. Document age is a representative of recency. It is the time difference from the timestamp when the user viewed the article to the timestamp when the article was created. Our Phase 2 model uses document age as the recency feature.

Article popularity *gmp* is used as a feature, as described in section **??**. *gmp* is not steam dependent. *gmp* is calculated by user actions on the Top Stream, as shown in Fig. **??** because Top Stream can have sufficient user traffic to derive reliable CTR.

Content based features are used. These content based features are from document profile explained in section **??**. A few features are derived from document profile such as total number of entities and categories, sum of aboutness scores and other arithmetic calcuation of aboutness scores.

Similar features as content-based are generated for user-based that uses user profile such as number of user inferred entities and categories, sum of sparse polarity scores and related arithmetic calculation of sparse polarity scores.

Correlations between user profile and document profile contribute some more features. Other than there is a single dot product used in Phase 1, Phase 2 uses number of matched entities between the two, 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 called source space id, srcspaceid. The feature defines the stream the article belongs to. In the hiearchical structure **??**, 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 srcspaceid="nfl". These are internally transformed into a 1-dimensional discrete valued feature, so that for instance, "nfl" ,"nba" ,"mlb" are assigned, $0, 1, 2$ respectively. We will show in the experiments this feature is very useful.

The training is based on click. 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 one interested by her. Then she may click the article and go to the landing page to read full of the article. A click is a positive signal/example, which proves the article is worth to recommend. The article is defined as clicked article. On the contrary, skipped articles are those that are on above of clicked articles but not clicked. Skipped articles are viewed articles but not clicked. For those articles below clicked articles, user may or may not view them because user has no action on these articles. These articles were not used in training. User's click or skip actions are used as positive or negative examples in the model training.

To avoid ranking bias and position bias, we used an exploration bucket (only exploration not exploitation bucket) [?] . Articles in the exploration bucket were ordered randomly. No features or any models took effect. Everything starts from scratch regarding to ranking.

Our training data generation is formalized as follows. If user i clicks some articles on 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 is a triple element produced by user i on article j. x_j^i corresponds to feature vector of the jth article and the user i . y_j^i is a binary value $\{0, 1\}$. If j is a clicked article, then $y_j^i = 1$. It is 0 otherwise. w_j^i is weights of the training example. In our system, its value equal to logarithm of dwell-time which is the time spent in the reading of the full article j, that is, the time difference from when user i clicks the article to when she leaves the article. Only clicked article has dwell time. $w_j^i = 1$ for skipped articles. N is the last clicked article number. The total article number is 200, but in reality $N \leq 200$ to exclude non-viewed articles from training.

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

Above, $age, gmp, srcspaceid$ are article age, popularity and source space id. u_i is a few features generated from user-based profile. d_j is a few features generated from content-based document profile. $u_i \circ d_j$ are correlative features between user profile and document profile.

Given training examples $(x_j^i, y_j^i, w_j^i), i \in (1, M), j \in (1, N_i)$, M is total users, N_i is total samples for user i, we can apply multiple machine learning methods to learn a regression model. We employ Gradient Boosted Decision Tree (GBDT) algorithm [?, ?] as the learning method of choice. This algorithm has higher precision, not sensitive to over training, and accomodate string value, srcspaceid. Gradient Boosted Decision Tree 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 β_t, with a finite number of parameters, Θ_t; λ 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 β_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 judgment. 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 the user 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 label 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 $\mathrm{FP_i}/\mathrm{TP_i}$ has no monotonicity 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 [?] 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

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 **??**.

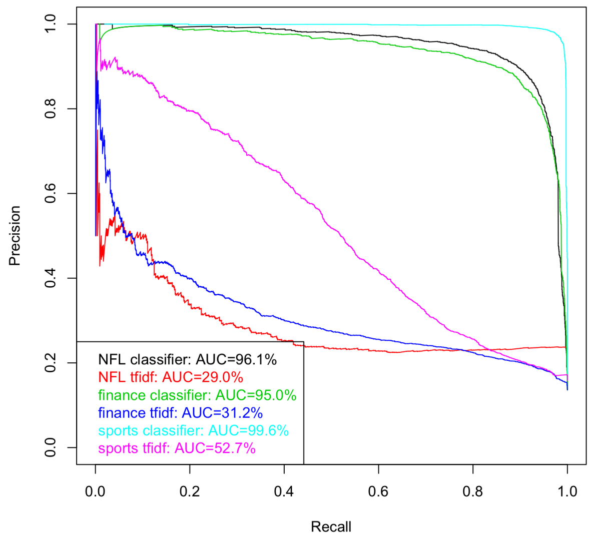


Figure 2: Precision-Recall Curves

For the linear classifier we use threshold at 0. The data set is always the editorially labeled validation set, since for training and testing data the labels are generated using production filter.

property	labels (+/-)	prod. 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%

So for all three streams, we are able to maintain or improve precision while greatly increase recall rate 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.

4.2.2 Phase 1

Here we present the ROC AUC for Sports and Finance under four sets of training methodologies:

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

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

4.2.3 phase 2

Our data were provided by an internet company. We built ranking models for Sports, Finance and News stream respectively. We collected two weeks of user action data from these streams. These two week data was used as training. We also collected one more week data as test. The method of data collection has been described in Section **??**. Sports, Finance and News are big streams. Under them there are small streams such as NFL,NBA,MLB for Sports. These structure is shown in Fig. **??**. Data distribution are shown in Table. **??**. Only Sport's sub-streams are given because some results are related to them. The table shows the number of clicks and views for different properties in our training and test data.

property	click	view	total
Sports	108K	1.5M	1.6M
NFL	20K	328K	347K
NBA	7K	66K	72K
MLB	7K	9K	95K
Finance			
News			

Table 1: Click data distribution

	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

Table 2: Comparision of Phase 2 ranking models

The size of articles in Phase 2 for user's each visit is 200. This is a relative small number so that we can afford to use a much more expensive in computation but higher in precision. We chose GBDT as Phase ranking model. The comparision between linear regression and higher feature number are shown in Table **??**. The base model was made by linear regression with features from Phase 1 (equal to flat gmp as in Table **??**). Note the reseach on Phase 1 and Phase 2 was independent. Experiments on Phase 1 and Phase2 used distinct training set and test set. Hence, the numbers shown on Table **??** and Table **??** are not sync-ed. Two GBDT models are tested. One used phase 1 features and the second used Phase 2 features as described in Section **??**. Phase 2 features have 14 features in total. The same algo was applied into three properties: Sports, Finance and News. The results are consistent across the three properties. The linear Base model output the worst results. The GBDT with Phase 2 features derived the best results. The number is AUC.

Intuitively, we would assume better results could be derived if each stream uses its own model. Actually, we found this assumption was right if streams are showing much different in contents such as Sports, Finance and News. These are heteogenerous streams. We built a separate model each for Sport, Finance and News. But this assumption is incorrect for homogeneous steams by our experimentss such as NFL,NBA and MLB. 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 **??**.

We found the model trained on all sports data performed better than dedicated models. We interpreted this as using all data can have rich feature distribution in training data. Feature distribution is biased for dedicated models. Rich feature distribution can't be satisfied by simply increasing training data size of dedicated models. Actually, the train-

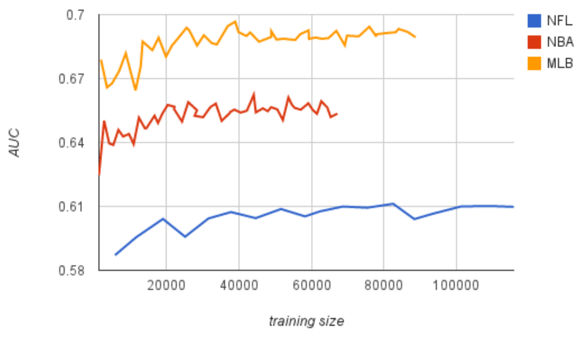


Figure 3: Test data metrics as increasing training data size

ing data sizes we used for dedicated models were sufficient to make maximal performance. As shown in Fig. **??**, it is a training curve which shows the trend of test data AUC change as increasing training data size. The training data size has arrived at the maximal results. Increaing more training data size does not help increasing the test data metric.

While we found dedicated models not the best performer for sports data, adding an indicator feature to distinguish source of sub-stream from one another does help. Using this feature, AUC metric increases for all cases. (last line of Table **??**). srcspaceid is a useful feature for multiple stream ranking.

Recency must be considered in news recommendation. Users prefer fresh contents, especially for frequent users. They want to read fresh contents when they come back again. Recency is not equal to ranking by time. That will present users many very low quality contents. Usually, there are two choices in handling recency. First as used in Phase 1, recency is boosted by applying an exponential age decay factor. Old age documents are punished. Using exponential function proves to have better quality. Related works can be found in [**?**]. Second, as used in Phase 2, we use document age as features directly in building the gbdt model. The age feature is used indiscriminately with other features. The importance of age feature in the model is completely determined by the training data used and objective of CTR optimization. We show the results and compare the two methods in table **??**.

AUC is chosen to evaluate of ranking correctness. We use ”top 1 average age” as recency metric. Namely, it averages document age by aggregating over the top 1 documents in all sessions. We found that using document age as a gbdt feature achieves better results than using expontial age decay: both AUC and top 1 average age are improved. Expontial age decay hurts AUC a lot although top 1 average age is improved. Based on online user experience, user clicks are more helpful to AUC than recency target.

Overall, GBDT gives higher ranking results than Phase 1 ranking. Our offline experiments sync with online results. We see higher CTR and dwell time increase in online bucket test **??**.

test data / 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 srcspaceid	0.642	0.635	0.689	0.725

Table 3: Substream evaluation under various setting

	AUC	top1 average age
Sports Base		
Sports (expontial age decay)		
Sports (age as gbdt feature)		
Finance Base		
Finance (expontial age decay)		
Finance (age as gbdt feature)		

Table 4: Recency evaluation

	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%

Table 5: bucket test