Click-Based Content Recommendation for Hierarchical Semantic Streams

Yunjiang Jiang jyj@yahoo-inc.com
Ruiqiang Zhang ruiqiang@yahoo-inc.com
Rao Shen raoshen@yahoo-inc.com
Karolina Buchner karolina@yahoo-inc.com
Tina Liu tliu@yahoo-inc.com
Sudharshan Lamkhede lamkhede@yahoo-inc.com
Youssef Billawala billawal@yahoo-inc.com

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. 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 (phase 0).

To solve the ranking problem under stringent runtime constraints, we propose a system design with 2 components: a coarse linear ranking step followed by a fine gbdt based step. 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 (**??**, stream classifier(section 3.2), Phase 1 ranking (3.3) and Phase 2 ranking (3.4). 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 [13]. For instance, web document link structures were exploited to predict document categories in the work of [2]. 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 ([1, 9]). Some recent work give more up-to-date improvment on using click [7].

The logistic regression model used in Phase 0 is a classical supervised approach[6], 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 [14] and LibLinear [4], the latter implemented in Weka.

We also could not find any previous work on the twisted

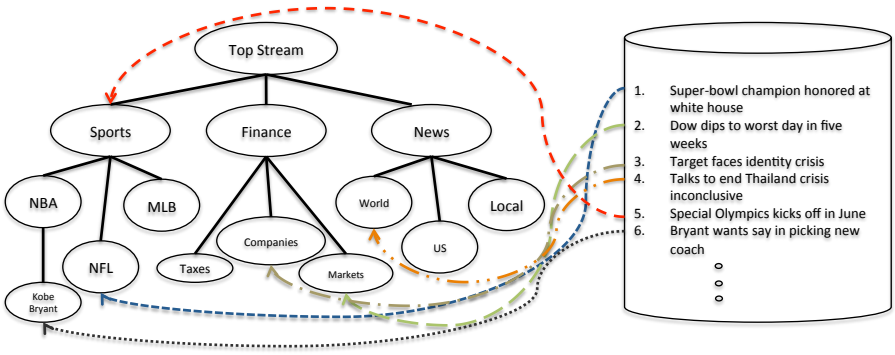

Figure 1: Semantic Stream Hierarchy

dot-product idea behind Phase 1. However it is closely related to tensor factorization models [12], [16]. 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([3]) 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.

3. SYSTEM DESIGN

3.1 Motivations

Designing an actionable system for thousands of article streams with millions of contents 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, 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 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 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 features such as wikis and ycts aboutness scores. One could argue that not all classification criteria can be compactly summarized by a handful (certainly fewer than 100) of logical conjunctives and disjunctives. As we will show in the experimental results section below, 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, which lead to hundreds of terminal decision nodes.

As a baseline, we also consider the term-frequency inverse-document-frequency (TFIDF) approach [11], 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:

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

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.

3.2.1 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 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, positive examples consist of articles with clicks from at least 5 unique users. 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.

Because of the low-recall concern with the above approach, we want to avoid false negatives at the expense of true negatives. To ensure scalability, we use the cold-start rule based filter to select negative examples, but exclude the low click count articles from training since they could be unpopular due to either irrelevance or low quality.

For streams not yet launched, 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. To avoid flooding a specialized stream with generally popular topics, editorial judgments are periodically injected as training samples.

Other active learning aspects of the model updating will be explored in future work.

3.2.2 Feature and Model Selection

We compared the relative performance of 4 distinct families of features: title and body token stems and frequency counts, yct and wiki aboutness score, editorial tags, and publisher ids.

While the topic category and publisher id together achieve high performance on testing, they perform poorly on editorial test. This is likely because we are merely relearning the rule based filter. 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, so we choose them for all streams.

To determine the exact classifier form, we found that the performances of logistic loss and hinge loss (SVM) are comparable. Since our preferred training tool, Vowpal Wabbit [1] 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.

3.3 Phase 1: Twisted Dot Product

The classifier instrumented at Phase 0 is not perfect, in particular we do not anticipate 100% precision. Subsequent phases will remove any embarassing candidate article for each stream. The Stream Relevance Query (SRQ) proposed at Phase 1 can be viewed as an optimization step for the more serious ranking done at Phase 2.

The production linear ranking uses the following formula:

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

Here $\mathbf{u}, \mathbf{d}$ denotes the user and dcument profile vectors respectively. 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. SRQ builds on (1) and takes the following form:

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

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

The main issue is that the new stream-specific user profile $\mathbf{u}'$ can now have hundreds of thousands of nonzero entries, making it non-sparse and computationally intensive. So we instead consider the following practical SRQ model:

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

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.

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

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.

3.4 Phase 2: Boosted Ranking

Phase 1 performs a coarse linear ranking to ensure fast and high recall article selection. The top 200 articles are passed to Phase 2, which performs more delicate ranking. The number 200 was selected according to Phase 2 complexity and system latency constraints.

Phase 2 ranking model integrates various type of features, including recency (approximated by document age), article popularity (gmp), and content based features. The latter includes 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, 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 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 ranking of the 200 articles are completely randomized to eliminate position bias [7]. The articles are presented to a user in the format of a scrollable stream. Only title and a short summary are shown. Among the articles not clicked, only the ones above the last clicked article (skipped ones) are used for training, since users most likely have viewed but decided not to click on them.

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 jth 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

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%

in reality $N \leq 200$ due to discarding non-viewed articles from training.

We employ Gradient Boosted Decision Tree (GBDT) algorithm [5, 15] 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.

4. EXPERIMENT RESULTS

4.1 Test Results

4.1.1 Phase 0

The precision-recall curves for the three streams (Sports, Finance, and NFL) under linear classifier and tfidf approaches are shown in Figures **??**. 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.

4.1.2 Phase 1

Here we present the ROC AUC for Sports and Finance under four sets of training methodologies (Table 3). 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 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 $\mathbf{q}_i \leq 0$.

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

Table 4: Click 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

4.1.3 phase 2

We build ranking models for Sports, Finance and News stream respectively, collecting two weeks of user action data for training and one week for test. Sports, Finance and News are big streams. Under them there are small streams such as NFL,NBA,MLB from Sports (see Fig. 1). 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 (equal to "flat gmp" as in 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 3.4. 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.

Intuitively, one would assume that better results could be derived if each stream uses its own model. Actually,

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

Table 6: Substream evaluations

training data \ test 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

Figure 3: Test data metrics as increasing sub-stream training data size

we found this to be true only for heterogeneous top level streams such as Sports, Finance and News, but 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.

We found that the model trained on all sports data performed better than dedicated models. We interpret this to mean that using all data can have rich feature distributions while feature distributions are biased for sub-streams. Simply increasing training data sizes can't satisfy rich feature distributions. Actually, the training data sizes we used for sub-stream models are sufficient to achieve maximal performance. As shown in Fig. 3, it is a training curve 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 make the maximal results, which proves increaing more training data size does not help.

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. On the other hand, recency can't be over-promoted. Ranking solely by document age is a very bad idea because this will show many low quality and irrelevant articles to users. Therefore, a balance between relevance and recency needs to be considered. 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 [8, 10]. 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, the average document age under the Sports baseline model 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 (3). The model "age as feature" used age simply 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 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 deals with hierarchical content classification and phase 1 and 2 perform efficient ranking. =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 (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.

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

[2] 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.

[3] 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.

[4] 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.

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

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

[7] 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.

[8] 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.

[9] 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.

[10] 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.

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

[12] 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.

[13] 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.

[14] 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.

[15] 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.

[16] 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.