Tracking Changes in User Interests with a Few Relevance Judgments

Dwi H. Widyantoro & Thomas R. Ioerger
Texas A&M University
Department of Computer Sciences
College Station, TX 77843
dhw7942,ioerger@cs.tamu.edu

John Yen
The Pennsylvania State University
School of Information Sciences and Technology
University Park, PA 16802
jyen@ist.psu.edu

ABSTRACT
Keeping track of changes in user interests from a document stream with a few relevance judgments is not an easy task. To tackle this problem, we propose a novel method that integrates (1) pseudo-relevance feedback mechanism, (2) assumption about the persistence of user interests and (3) incremental method for data clustering. This approach has been empirically evaluated using Reuters-21578 corpus in a setting for information filtering. The experiment results reveal that it significantly improves the performances of existing user-interest-tracking systems without requiring additional, actual relevance judgments.

Categories and Subject Descriptors
H.3.3 [**Information Search and Retrieval**]: Clustering, Information Filtering, Relevance Feedback.

General Terms
Algorithms, Management, Experimentation.

Keywords
User Interest Tracking, Persistence Assumption, Pseudo-relevance Feedback, Incremental Data Clustering.

1. INTRODUCTION
Modeling user interests has been an active research area since the past decade. Researchers address the problem of changing user interests by decomposing an interest category into long-term and short-term interest models, relearning examples of recent window, applying decay functions, or employing evolutionary algorithms, among others. Despite their success, most of these approaches implicitly assume that numerous relevance judgments are available to systems. However, users tend to be unwilling to provide the relevance judgments needed [4], making the assumption above not practical. A more realistic setting is to assume that the systems require only a few relevance judgments in order to work with acceptable performance.

CIKM'03, November 3-8, 2003, New Orleans, Louisiana, USA.

Unfortunately, learning changing interests with a few feedback iterations is a difficult task and little effort if ever has been done with it. From the Computational Learning Theory perspective, decreasing the number of examples in a drifting environment corresponds to increasing the rate of drift, which would hurt the system performance [1]. The drift rate is the probability that two consecutive feedback documents with the same topic category disagree on their relevance judgments. Thus, tracking the evolution of user interests is harder if the interest categories change more quickly. Although this problem has been recognized in previous work [5], no solution has been proposed. In addition, methods developed for dealing with a few examples in tasks such as TDT's topic tracking and automatic query expansion are mainly still limited to learning static user interests.

Motivated by this observation, we develop a new method for tracking the dynamics of user interests from a minimal number of relevance judgments.

2. OVERVIEW OF APPROACH
Figure 1 provides the summary of our approach. Like in information filtering setting, the input system is a stream of documents S in which the relevance judgments of some documents in the stream are made available. Two subsystem components independently process each document sequentially. The first component organizes all documents from the stream into a cluster hierarchy H. The clustering process is performed incrementally in unsupervised mode (i.e., ignoring the relevance judgments if any) as it sees a new incoming document in the stream. The second component so-called *context tracker* maintains a sequence of genuine relevance feedback S_F in the order of feedback arrival times.

The context tracker is invoked only when a classifier needs to build a search profile for information retrieval. It takes as input the stream S_F and the cluster hierarchy H generated up to that point, and outputs a new stream S_{PF} containing a subset of S_F and relevant documents retrieved from the cluster hierarchy. The classifier then uses the stream S_{PF} to generate search profile Q by means of *pseudo-relevance* feedback mechanism [6].

The change in notion of *relevance* is inevitable in tracking the evolution of user interests so that relevance judgments for seeding the pseudo-relevance feedback process must be selected from those in S_F that are still truly relevant. The context tracker addresses this problem by associating each document in S_F with its context – that is, an abstraction of interest topic category. It then applies the assumption about the persistence of user interests to infer the relevance of each context. Finally, it retrieves all

Input: a stream of documents S.
Initialization:
$S_F = \langle\varnothing\rangle$, the sequence of relevance judgments.
$H = \varnothing$, the cluster hierarchy.

Incremental Learning
For each document d observed from the stream S,
1: Update H incrementally to incorporate d.
2: If the relevance judgment j of d is available,
Concatenate $\langle(d, j)\rangle$ at the end of S_F.

Search Profile Generation (only when necessary):
Apply the *context tracker* to generate new stream S_{PF} based on current values of S_F and H.
Apply *pseudo-relevance* feedback process on S_{PF}, using a selected classifier, to generate search profile Q.

Figure 1. The summary of approach.

documents relevant to each context from the cluster hierarchy, and uses the retrieved documents to generate the new stream S_{PF}.

3. DOCUMENT-CONTEXT ASSOCIATION

We use a cluster hierarchy, which is a tree structure, to provide document-context associations. Each leaf node represents a singleton cluster containing a single document. A node in the tree also represents a context. In this paper we assume that the most appropriate context that can be associated with a document is one of the document's ancestors in the hierarchy that distinctively partitions documents underneath. The followings are two useful functions needed to access the document-context associations:

1. c = *Context*(d) returns a context c that can be associated with an input document d.
2. **D** = *Context_Extension*(c) is a function that returns a set of documents **D** = $\{d_1, \ldots, d_n\}$ with respect to the input context c. These documents are all leaf nodes that are descendants of a node representing context c in the tree.

We employ a recently developed algorithm for incremental hierarchical clustering [9]. One of the cluster properties generated by the clustering technique is *cluster density*, which is calculated from the average distance to the nearest neighbor among the cluster's members. We identify distinct contexts by thresholding the cluster density information, similar to *proximity dendogram* cutting that identifies clusters according to dissimilarity levels [2].

The density threshold for distinct context identification is empirically determined from a validation set. First, a cluster hierarchy is incrementally built from a stream of documents in the validation set. Because the document topics are known, distinct clusters (i.e., contexts) that correspond to topics in the validation set can be accurately identified. The threshold is then calculated from the average density of these distinct clusters.

Specifically, let H be the cluster hierarchy generated from the validation set containing a set of topics T. Let $c_x \in H$ be a cluster that corresponds to topic $x \in T$. Furthermore, let $\mathcal{E}(c)$ be a set of documents that are members of cluster c, i.e., *Context_Extension*(c). The cluster c_x is identified from H by:

$$c_x = \arg\max_{c \in H} \left\{ \sum_{d \in \mathcal{E}(c)} t_{d,x} - \sum_{y \in T - \{x\}} \sum_{d \in \mathcal{E}(c)} t_{d,y} \right\} \quad (1)$$

where $t_{d,z} = 1$ if the topic of document d is z, or 0 otherwise. Hence, c_x maximizes the difference between the numbers of cluster members that are on x topic and non-x topics. Now let δ_x be the average distance to the nearest neighbor among c_x's members; δ represents the cluster density in the employed clustering technique. Thus, a higher δ value corresponds to a lower-density cluster and vice versa. Let $\delta_{x's\ parent}$ be the density of c_x's parent. Taking δ_x as the threshold runs the risk of overfitting to a specific topic while selecting $\delta_{x's\ parent}$ is completely inappropriate because it also covers the contexts of other topics (over-generalization). Therefore, the threshold is selected at a value between δ_x and $\delta_{x's\ parent}$, averaged over all topics:

$$\theta_k = \frac{1}{|T|} \sum_{x \in T} \max\left\{\delta_x, \delta_x + k \cdot \left(\delta_{x's\ parent} - \delta_x\right)\right\} \quad (2)$$

where $0 \le k \le 1$. We set $k = 0.5$ by default, which maximizes the margins between overfitting and overgeneralization. Given a document d, *Context*(d) returns a cluster c, whose cluster density is δ_c, such that $\delta_c \le \theta_k \le \delta_{c's\ parent}$ and c is one of d's ancestors.

4. TRACKING CONTEXT RELEVANCE

Now we describe the *context tracker* algorithm in detail. Let $S_F = \langle(d_1, j_1), \ldots, (d_n, j_n)\rangle$ be the stream of documents d_i with its relevance judgment j_i. The relevance judgment is either 1 (*relevant* or *positive*) or 0 (*irrelevant* or *negative*). A document at the left side arrives earlier. The *context tracker* processes its input in four steps.

Step 1: *Transforming the problem of tracking multiple interest categories into several sub-problems of tracking single interest category*. The original stream of relevance judgments is partitioned into several, shorter sequences according to their contexts. The relative ordering of documents in the original stream is also maintained in each partition.

Example1. Let $S_F = \langle(d_1,1),(d_2,0),(d_3,0),(d_4,1),(d_5,0),(d_6,0),(d_7,1), d_8,1)\rangle$. Suppose that the sets $\{d_2,d_6\}$, $\{d_1,d_4,d_5\}$ and $\{d_3,d_7,d_8\}$ belong to contexts c_1, c_2 and c_3 respectively (i.e., *Context*(d_2)=*Context*(d_6)=c_1 in the first set, and so on). Then, S_F is partitioned into $S_{F1}=\langle(d_2,0),(d_6,0)\rangle$, $S_{F2}=\langle(d_1,1),(d_4,1),(d_5,0)\rangle$ and $S_{F3}=\langle(d_3,0),(d_7,1),(d_8,1)\rangle$ during the first step.

Step 2: *Normalizing sequence partitions*. Since each partition generated by the first step contains documents of the same context, two or more consecutive documents with the same judgment constitute a redundancy. Current step eliminates such a redundancy by repeatedly dropping a document whose judgment is the same as that of the next document on the sequence.

Example 2. From *Example 1*, d_2 is dropped from S_{F1} because its subsequent document, d_6, has the same relevance judgment. Similarly, d_1 and d_7 are also removed from S_{F2} and S_{F3}, respectively. It then generates normalized sequence partitions $S'_{F1}=\langle(d_6,0)\rangle$, $S'_{F2}=\langle(d_4,1),(d_5,0)\rangle$ and $S'_{F3}=\langle(d_3,0), (d_8,1)\rangle$.

Step 3: *Inferring the relevance of each context*. To do this, we adopt the persistence assumption in temporal reasoning, which

states that once a fact becomes true it remains true thenceforth until the fact is negated [3]. The context relevance thus can be simply inferred from the relevance value given by the last document in the normalized sequence partition. In addition, we drop a context if the last document is not relevant while the preceding document is relevant (i.e., $\langle(d_i,1),(d_j,0)\rangle$).

Example 3. From *Example 2*, context c_1, as represented by S'_{F1}, is irrelevant while context c_3 is relevant. Context c_2 (as represented by S'_{F2}) is no longer counted.

Step 4: *Generating stream for pseudo feedback process*. It first retrieves documents that belong to the remaining contexts from cluster hierarchy, using *Context_Extension*() function. A new stream S_{PF} is then generated from the retrieved documents and is ordered by the document arrival times. The relevance judgment of a document in S_{PF} is set to the relevance value of its context.

Example 4. Let $\{d_2,d_6,d_9,d_{11}\}$=*Context_Extension*(c_1) and $\{d_3,d_7,d_8,d_{10},d_{12}\}$=*Context_Extension*($c_3$) be the sets of documents relevant to contexts c_1 and c_3. The last step will generate $S_{PF}=\langle(d_2,0),(d_3,1),(d_6,0),(d_7,1),(d_8,1),(d_9,0),(d_{10},1),(d_{11},0),(d_{12},1)\rangle$.

5. EXPERIMENT SETUP

Our experiments used a subset of the Reuters-21578 corpus, and selected among documents in the *ModApte* split that had been assigned a single topic category. As a result, the test set contained 2581 documents consisting of 59 topics. The validation set needed for determining the density threshold was randomly selected from the training set, and contained 100 documents of five topics. The rest of the training set (6452 documents) was used to generate document streams. We applied stop word removal, word stemming and *tf-idf* weighting method to all documents.

The main evaluation objective was to observe the system performance over time in adapting to various scenarios of changes in topics of interest. Accordingly, the system was presented with a stream of documents to process sequentially. Its performances were then measured at regular intervals on a fixed test set. Let *tracking cycle* be a period of processing *m*-document sequence and system performance measurement. Referring to the set of processes depicted by Figure 1, the experiments were conducted using the following procedures:

1. **Initialize** S_F and the concept hierarchy H.
2. At each *tracking cycle* $i = \{1 \ldots K\}$
3. Perform **Incremental Learning** process on the i^{th} *m*-document sequence taken from the stream S.
4. Execute **Search Profile Generation** process to create the search profile Q.
5. Rank all documents in the test set, using the same classifier as in Step 4 and the search profile Q, and measure the system accuracy based on the *recall-precision break-even point* performance measure.

In this paper we consider MTDR [8] and Rocchio [7] algorithms as the classifiers needed to perform steps 4 and 5 above.

The document stream S was generated according to a tracking task, a scenario that described the document stream and the evolution of user interests. The changes in topics of interest over time were simulated by alternating among interests in *Trade*, *Coffee*, *Crude*, *Sugar* and *Acq* topics. These five topics are called target topics whose sizes in the test set were 75, 22, 121, 25 and 696, respectively.

Table 1. Description of three tracking tasks for experiments.

	Tracking Cycle				
	1 – 20	21 – 40	41 – 60	61 – 80	81 – 100
S_1	Trade & 9 others	~~Trade~~, Coffee & 8 others	~~Coffee~~, Crude & 8 others	~~Crude~~, Sugar & 8 others	~~Sugar~~, Acq & 8 others
S_2	Trade, Coffee & 8 others	~~Trade~~, Coffee, Crude & 7 others	~~Coffee~~, Crude, Sugar & 7 others	~~Crude~~, Sugar, Acq & 7 others	
S_3	Trade, Coffee, Crude & 7 others	~~Trade~~, Coffee, Crude, Sugar & 6 others	~~Coffee~~, Crude, Sugar, Acq & 6 others		

Table 1 described three tracking tasks that were represented by streams S_1, S_2 and S_3, respectively. A document stream was arranged into tracking cycles each of which consisted of 10-document sequence. The table indicated the presence of document topics on the corresponding tracking cycles. The *other* and ~~strikethrough~~ topics represented irrelevant documents. Relevance judgments are never given to documents with the *other* topics.

Positive feedback was used for establishing a new (or emphasizing the interest in a) topic category. Negative feedback was given for expressing the disinterest in topics previously considered relevant. These relevance judgments were made available at tracking cycles that marked the beginning of change in current target topics, and formed the stream known as S_F. The current target topics were changed after periods of twenty tracking cycles. Starting from the first tracking cycles, the next relevance judgments were given to some documents at tracking cycles 21, 41 and so on. When relevance judgments were made available, the underlined (~~strikethrough~~) topics indicated that the corresponding documents were considered relevant (irrelevant).

6. RESULTS AND CONCLUSION

Figures 2 and 3 showed the performances of systems over time, averaged over ten runs, on the three tracking tasks. PSEUDO-FEEDBACK performances in these figures were obtained by applying the *pseudo-relevance* feedback mechanism on the stream S_{PF} generated by the context tracker. For comparison, we also presented the performances of FULL-FEEDBACK and PARTIAL-FEEDBACK systems that did not use the context tracker's outputs. The FULL-FEEDBACK performances were produced by providing the stream S, making the relevance judgments of all target topic documents in the stream available while ignoring all non-target topic documents. The FULL-FEEDBACK systems provided the empirical upper bound performances. The PARTIAL-FEEDBACK results were generated from processing only the genuine relevance feedback stream S_F, which represented only 5% of the numbers of feedback documents given to the FULL-FEEDBACK systems. It served as the empirical lower bound (baseline) performances.

The PARTIAL-FEEDBACK systems were given relevance judgments only at the first tracking cycles during the twenty-tracking-cycle periods. Consequently, their accuracies were constant, because the

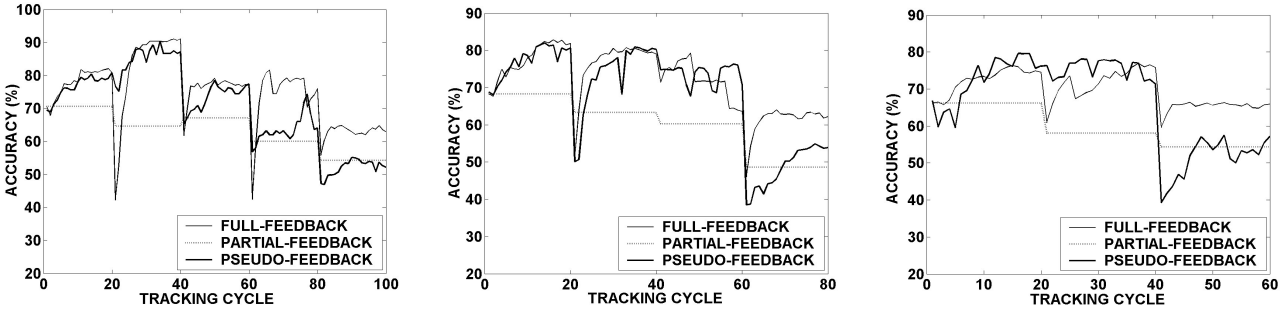

Figure 2. The performance over time of MTDR classifier on tracking tasks S_1 (left), S_2 (middle) and S_3 (right).

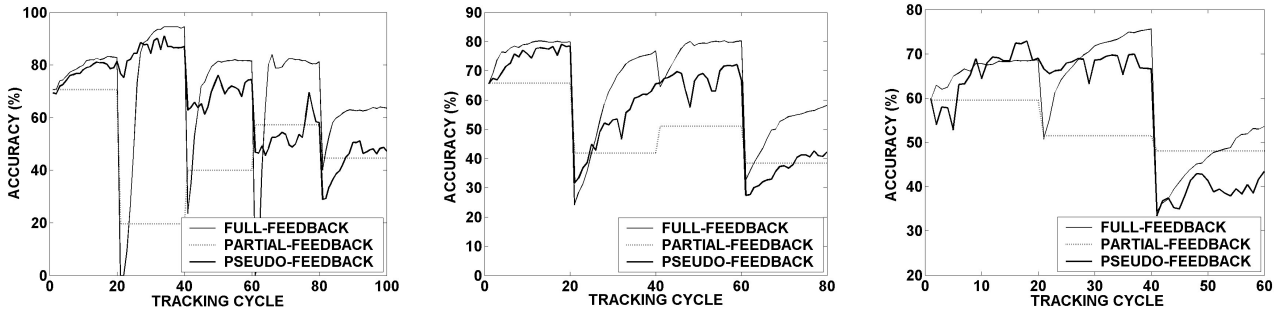

Figure 3. The performance over time of Rocchio classifier on tracking tasks S_1 (left), S_2 (middle) and S_3 (right).

search profiles were evaluated on the same test set, until the next relevance judgments were received. By contrast, the trends of FULL-FEEDBACK performances were increasing over time during which the target topics were stable. These were not surprising because the systems were able to continuously revise the search profiles at each tracking cycle.

Regardless of the classifier employed for implementing the relevance feedback process, the performances of PSEUDO-FEEDBACK systems improved over the baseline performances, as shown in Figures 2 and 3. Using the same sequences of relevance judgments as those given to the PARTIAL-FEEDBACK systems, the PSEUDO-FEEDBACK systems gained their performances as more relevant documents became available. Except during the last twenty tracking cycles, most of the performance gains achieved over the baseline performances were significant, and in some cases were even better than the performances of the FULL-FEEDBACK systems. Nonetheless, the PSEUDO-FEEDBACK systems during the last twenty tracking cycles were still able to improve their performances automatically, although rather slower. This tendency was very encouraging.

In conclusion, we have presented and evaluated the method for tracking the dynamics of user preference with minimal relevance judgments. Its evaluation on three tracking tasks indicates the feasibility and the effectiveness of the method.

7. REFERENCES

[1] Bartlett, P.L., David, S.B. and Kulkarni, S.R. (1996) Learning Changing Concepts by Exploiting the Structure of Change, *Computational Learning Theory*, pp. 131-139.

[2] Jain, A.K. and Dubes, R.C. (1988) *Algorithm for Clustering Data*. Prentice Hall.

[3] Gabbay, D.M, Hogger, C. J. and Robinson, J.A. (1995) *Handbook of Logic in AI and Logic Programming: V4. Epistemic and Temporal Reasoning*.

[4] Jansen, B. J., Spink, A. and Saracevic, T. (2000) Real Life, Real Users and Real Needs: a Study and Analysis of Users Queries on the Web. *Information Processing & Management, 36*(2): 207-227.

[5] Klinkenberg, R. (1999) Learning Drifting Concepts with Partial User Feedback, Beiträge zum Treffen der GI-Fachgruppe 1.1.3 Maschinelles Lernen (FGML-99), Perner, Petra and Fink, Volkmar (ed.).

[6] Mitra, M., Singhal, A. and Buckley, C. (1998) Improving Automatic Query Expansion. In *Proc. of the 21st ACM SIGIR Conference,* pp. 206 - 214.

[7] Rocchio, J.J. (1971) Relevance Feedback in Information Retrieval. In G. Salton, *The SMART Retrieval System: Experiments in Automatic Doc. Processing*, pp. 313-323.

[8] Widyantoro, D.H., Ioerger, T.R. and Yen, J. (2001) Learning User Interest Dynamics with a Three-Descriptor Representation. *Journal of the American Society for Information Science*, 52(3): 212-225.

[9] Widyantoro, D.H., Ioerger, T.R., and Yen, J. (2002) An Incremental Approach to Building a Cluster Hierarchy. In *Proc. of the 2nd IEEE International Conference on Data Mining,* pp. 705-708.