Novelty-aware Event Delivery

Dimitris Souravlias

MASTER THESIS

— ♦ —

Ioannina, June 2011

ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ
ΠΑΝΕΠΙΣΤΗΜΙΟ ΙΩΑΝΝΙΝΩΝ

DEPARTMENT OF COMPUTER SCIENCE
UNIVERSITY OF IOANNINA

Novelty-aware Event Delivery

Η ΜΕΤΑΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ ΕΞΕΙΔΙΚΕΥΣΗΣ

υποβάλλεται στην

ορισθείσα από την Γενική Συνέλευση Ειδικής Σύνθεσης

του Τμήματος Πληροφορικής Εξεταστική Επιτροπή

από τον

Δημήτριο Σουραβλιά

ως μέρος των Υποχρεώσεων για τη λήψη του

ΜΕΤΑΠΤΥΧΙΑΚΟΥ ΔΙΠΛΩΜΑΤΟΣ ΣΤΗΝ ΠΛΗΡΟΦΟΡΙΚΗ

ΜΕ ΕΞΕΙΔΙΚΕΥΣΗ

ΣΤΟ ΛΟΓΙΣΜΙΚΟ

Ιούνιος 2011

DEDICATION

To my family.

ACKNOWLEDGEMENTS

I would first like to thank my supervisor Professor Evaggelia Pitoura, for guiding, encouranging, motivating me, for the time spent and especially, the patience she has shown until this thesis is completed. I would like to thank Kostas Stefanidis and Marina Drosou for our great collaboration. With Kostas and Marina, we have worked on the initial approach of exploring novelty as a ranking criterion in publish/subscribe systems. Many thanks to Georgia Koloniari and Nikos Ntarmos. With Georgia and Nikos, we have worked together during the last year on distributed filtering approaches for duplicate-free event delivery. Special thanks to my colleague and friend Vasilis Bourgos. With Vasilis, we have co-operated not only during our undergraduate but also during our postgraduate studies. Finally, I would like to thank my family and my friends for their continuous support throughout the years of my studies.

TABLE OF CONTENTS

List of Figures

List of Tables

Algorithm Index

ABSTRACT

Dimitris N. Souravlias. MSc, Computer Science Department, University of Ioannina, Greece. June, 2011. Novelty-aware Event Delivery. Thesis Supervisor: Evaggelia Pitoura.

In publish/subscribe systems users express their interests by submitting long standing queries, called subscriptions, and get notified whenever new events that match their interests become available.

Traditional publish/subscribe systems forward to users all available pieces of information that are relevant to one of their interests. In an effort to avoid overwhelming the users with this ever-growing ocean of relevant information, in this work we propose a new notion of relevance that is called novelty-aware relevance. An event is considered novelty-aware relevant, if it matches a subscription whose previously matching events were rarely delivered to the user. Novelty-aware relevance is used in a per user filtering mechanism, called novelty-based event filtering, that filters out relevant, but less novelty-aware relevant events in an effort to maximize the information gain received by the user.

We have fully implemented our approach and present our extensive experimental results which show that user-perceived novelty maintains a per user steady load and leads to efficient and effective event pruning.

Εκτεταμενη Περιληψη στα Ελληνικα

Δημήτριος Σουραβλιάς του Νικολάου και της Αικατερίνης. MSc, Τμήμα Πληροφορικής, Πανεπιστήμιο Ιωαννίνων, Ιούνιος, 2011. Διάταξη Γεγονότων βάσει του Χρόνου Δημιουργίας τους. Επιβλέπουσα: Ευαγγελία Πιτουρά.

Στα συστήματα έκδοσης/συνδρομής, οι χρήστες εκφράζουν τα ενδιαφέροντά τους υποβάλλοντας συνεχή ερωτήματα, τα οποία ονομάζονται συνδρομές, και ενημερώνονται όταν νέα γεγονότα που ταιριάζουν στα ενδιαφέροντά τους γίνονται διαθέσιμα.

Τα παραδοσιακά συστήματα έκδοσης/συνδρομής προωθούν στους χρήστες όλες τις διαθέσιμες πληροφορίες που είναι σχετικές με τα ενδιαφέροντά τους. Σε μια προσπάθεια να αποφύγουμε να κατακλύσουμε το χρήστη με αυτόν τον τεράστιο όγκο πληροφορίας, προτείνουμε μια νέα θεώρηση της συσχέτισης μεταξύ συνδρομής και γεγονότος, την οποία καλούμε συσχέτιση με βάση το **novelty**. Ένα γεγονός είναι σχετικό με βάση το **novelty**, αν ταιριάζει με μια συνδρομή, της οποίας τα γεγονότα στο παρελθόν παραδίδονται σπάνια στο χρήστη. Η συσχέτιση με βάση το **novelty** χρησιμοποιείται για κάθε χρήστη ως ένας μηχανισμός φιλτραρίσματος, που ονομάζεται φιλτράρισμα με βάση το **novelty**. Ο μηχανισμός αυτός απαλείφει τα λιγότερο σχετικά γεγονότα, με στόχο την αύξηση του κέρδους της πληροφορίας που λαμβάνει ο χρήστης.

Στην παρούσα εργασία, αρχικά προτείνουμε ένα θεωρητικό μοντέλο, όπου ορίζουμε την έννοια της **novel** συνδρομής και βάσει αυτής ορίζουμε την έννοια της συσχέτισης με βάση το **novelty**. Για το φιλτράρισμα γεγονότων προτείνουμε δύο νέους αλγορίθμους· έναν αλγόριθμο δειγματοληψίας και έναν αλγόριθμο κατωφλίου. Και οι δύο αλγόριθμοι απαλείφουν σε πραγματικό χρόνο τα λιγότερο σχετικά γεγονότα, με στόχο την αύξηση του κέρδους της πληροφορία που λαμβάνει ο χρήστης. Έχουμε υλοποιήσει πλήρως την προσέγγισή μας και παρουσιάζουμε τα εκτενή αποτελέσματα των πειραμάτων μας.

CHAPTER 1

INTRODUCTION

1.1 Scope of Thesis

Publish/Subscribe systems offer an attractive alternative to search by providing a proactive model of information supply that disburdens the users of the hard task of explicit search. In such systems, users (or subscribers) express their interest in specific pieces of data (or events) through long standing queries called subscriptions. Then, they get notified whenever an information source (or publisher) generates an event that is relevant (or matches) one of their subscriptions. Examples of such proactive delivery include news aggregators, RSS feeds and notification services in social networks such as Facebook and Twitter.

Traditional publish/subscribe systems forward to the users all events that are relevant to one of their subscriptions. In an effort to avoid overwhelming the users with this huge volume of relevant data, we introduce a new notion of relevance between subscriptions and events that is called novelty-aware relevance.

Novelty is gaining increasing interest in information retrieval as evaluation measure [10], [11]. As there is not a formal definition, one can define novelty as the need to limit redundancy by avoiding results with overlapping content. In our previous work [1], we have explored novelty as a ranking criterion in publish/subscribe delivery. Our interpretation of novelty is that an event is considered novelty-aware relevant, if it matches a subscription whose previously matching events were rarely delivered to the user.

This streaming notion of novelty is desirable for a major reason; making rare events visible and thus increasing the information gain of the user. Consider a user that submits subscriptions with varying rates of matching events. As an example, take a user in a social

networking application like Twitter, that follows both publishers that are very productive in terms of content generation and publishers that generate information seldom. Novelty-aware relevant events (i.e. events that match subscriptions which have been matched by events rarely delivered in the past) will be forwarded to the user whereas less relevant will be discarded, targeting that all subscriptions of the user will be equally represented in the stream of delivered events.

The focus of this work is on incorporating novelty-aware relevance in a per user filtering mechanism, called novelty-based event filtering, that filters out relevant, but less novelty-aware relevant events. To this end, we propose two algorithms than work in on-line mode; a threshold-based and a novelty-biased sampling algorithm. The first forwards to the user events that have novelty scores above a per user threshold. The second forwards to the user with high probability events that their matching subscriptions have been rarely matched in the past.

1.2 Thesis Outline

The rest of this thesis is structured as follows. In Chapter 2, we present the model and define the notion of novelty-aware relevance. Also, we present the relation between novelty and entropy. In Chapter 3, we explore novelty-based event filtering and present our threshold-based and novelty-biased sampling algorithms along with their variations. In Chapter 4, we present our evaluation setup and experimental results. Chapter 5 describes related work and finally Chapter 6 concludes this thesis with a summary of our contributions and outlines future work.

CHAPTER 2

MODEL

2.1 Event-Delivery Model

We consider a generic event delivery approach based on a typical publish/subscribe (pub/sub) model. In this model, users express their interests in certain events via long-standing queries, called *subscriptions*. Whenever an event is generated or *published*, it is matched against the current subscriptions. Matching events are then delivered to the corresponding users.

There are two broad types of publish/subscribe systems that differ on the expressive power of the subscription language: (a) *topic-based* and (b) *content-based* ones. In topic-based systems, users subscribe to topics and get notified when events on matching topics are generated. In content-based systems, subscriptions specify conditions that the content associated with the event must satisfy. Our novelty model is applicable to both types. Many popular web applications fall under this generic pub/sub paradigm. For example, many social networking sites such as Twitter and Facebook allow their users to "follow" the content generated by their friends. RSS feeds and news aggregators are other examples.

Let us assume that event matching is exact that is an event e either matches or does not match a subscription s. We denote this by $match(e, s)$ which is equal to 1 if subscription s matches event e and 0 otherwise.

2.2 Novelty-Aware Relevance

Let U be the set of all users, S be the set of all subscriptions, and I be the (potentially infinite) sequence of input events in the system. Further, let $M \subseteq I$ be the set of all input events that match at least one subscription in the system, and $O \subseteq M$ the set of all such events that were delivered to subscribers. For a specific user $u \in U$, we denote as $S(u) \subseteq S$ the set of the subscriptions of u, $M_s(u) \subseteq M$ the set of all input events matching a subscription $s \in S(u)$, and as $O_s(u) \subseteq M_s(u)$ the set of all events being delivered to u as a result of matching s.

Further, let $M(u) = e_1, \ldots e_i \ldots$ be the sequence of all input events matching at least one of u's subscriptions. We denote with $M_j(u)$ the subsequence of $M(u)$ up to event e_j, and by $M_{s,j}(u) \subseteq M_j(u)$ the sequence of events up to event e_j matching subscription s. Similarly, let $O_j(u)$ be the subsequence of $M_j(u)$ that includes only the events delivered to u, and $O_{s,j}(u) \subseteq O_j(u)$ be the sequence of events delivered to u as a result of matching s.

Typically, an event e is considered relevant to the interests of a user u and thus delivered to the user if it matches at least one of the subscriptions $s \in S(u)$. In this work, we make the case that the relevance of events does not depend solely on the user interests but also on the events previously delivered to that user, that is on how much novel an event is with respect to what the user has seen in the past. We call this new notion of relevance *novelty-aware event relevance*. To define the novelty-aware event relevance, let us first define the novelty of a subscription s. We discern two major ways of defining the novelty of a subscription; one according to the delivery rate model and the other according to the delivery interval model.

2.2.1 Delivery Rate Model

Let $R_{s,t}(u)$ be an indicator variable associated with subscription $s \in S(u)$, counting the number of events in $O_{s,t-1}(u)$; that is, $R_{s,t}(u) = |O_{s,t-1}(u)| = f$, if exactly f of the events in $\mathcal{I}_{t-1}$ were delivered to u as a result of s being matched *and* being the most novel among other matched subscriptions. Then, the novelty score of s for u at time t is given by:

$$novel_{rate}(s, u, t) = 1 - \frac{R_{s,t}(u)}{|I_{t-1}|}. \tag{2.1}$$

Note that the fraction in eq. 2.1 is equal to the delivery rate of s at time i; that is, the fraction of all input events in I_{t-1} having been delivered as a result of matching s.

2.2.2 Delivery Interval Model

Let $L_{s,t}$ be an indicator variable associated with subscription $s \in S(u)$, recording the last event before t which was delivered as a result of subscription s; that is:

$$L_{s,t} = \begin{cases} \max\{j : e_j \in O_{s,t-1}(u)\}, \text{ if } O_{s,t}(u) \neq \emptyset \\ t - 1, \text{ otherwise} \end{cases}$$

Then:

$$novel_{inter}(s, u, t) = t - L_{s,t}. \tag{2.2}$$

Note that $novel_{inter}(\cdot)$ is in essence the time between consecutive deliveries of events matching s. Naturally, the mean value of this metric, that is, the inter-delivery time for s, is equal to the inverse of the delivery frequency for s.

Given the above ways of computing subscription novelty, a subscription s is novel for user u at time t if and only if its score is above an appropriate per user threshold $th(u)$:

Definition 2.1. Subscription s is novel for user u at time t iff $novel(s, O_{s,t-1}(u)) > th(u)$

Also, the following theorem holds:

Theorem 2.1. *Both $novel_{rate}(\cdot)$ and $novel_{inter}(\cdot)$ result in equivalent rankings.*

Proof can be found in the Appendix.

2.2.3 Novelty-aware Event Relevance

In our model, each event e is delivered to user u if and only if it is novelty-aware relevant to at least one of the subscriptions of the user u. An event e is novelty-aware relevant if and only if both of the following conditions hold:

(a) The event e matches at least one of the subscriptions $s \in S(u)$.
(b) At least one of the subscription s that is matched by e has a novelty score above user threshold $th(u)$.

Each event e delivered to the user u is associated with a novelty score that indicates how much novel it is for user u. This score is computed in accordance with the score of the subscriptions that cover it. In the simple case, in which event e matches exactly one subscription $s \in S(u)$, the novelty score of the event is assigned the novelty score of the subscription that covers it. In the case that we have multiple matches, the novelty score of the event is assigned the score of the *most novel* of the subscriptions (i.e. that has the highest novelty score) of user u that cover e.

For example, take two subscriptions $s_1 = \{director = S.Spielberg, genre = sci - fi\}$ and $s_2 = \{genre = sci - fi, release\_year > 1999\}$. Note that none of the subscriptions covers the other. An event that matches both subscriptions will get the score of the most novel one. For example, if many events that match subscription s_1 have been delivered to the user and only a few that match subscription s_2, then the event is assigned the score of s_2 (that is the most novel one) so that rare events are noticed.

2.3 Novelty and Entropy

2.3.1 Entropy

In information theory, entropy is a measure of the uncertainty associated with a random variable. It was first introduced by Shannon in [2]. Entropy quantifies the expected value of the information that is contained in a message and is measured in terms of bits. The entropy H of a discrete random variable X with posible values $\{x_1, x_2, \ldots, x_n\}$ is:

$$H(X,n) = E(I(X)) \tag{2.3}$$

Here E denotes the expected value and I denotes the information content of X. If p denotes the probability mass function of X, then entropy can be written explicity as:

$$H(X,n) = -\sum_{i=1}^{n} p(x_i) \cdot log_2(p(x_i)) \tag{2.4}$$

Theorem 2.2. *Entropy is maximized when $p(x_i) = 1/n$, for $i = 1, \ldots, n$, that is when the random variable X follows a uniform distribution.*

The value of the maximum entropy is equal to:

$$H_{max}(X,n) = -\sum_{i=1}^{n} 1/n \cdot log_2(1/n) = n \cdot 1/n \cdot log_2(n) = log_2(n) \tag{2.5}$$

In this work, we measure the entropy of events of the output stream, namely the events that are delivered to a user u. In our case, the probability mass function is based on the probability that an event e that is delivered to user, matches one of the subscriptions in $S(u) = \{s_1, s_2, \ldots, s_n\}$ of the user u.

For example, let $S(u) = \{s_1, s_2, s_3\}$ be the subscriptions of the user u and 100 the total number of events that are delivered to him. Also, suppose that 20 of the delivered events match subscription s_1, 30 match s_2 and 50 match s_3. Then the output entropy is:

$$\begin{aligned} H(X,3) &= -\sum_{i=1}^{3} p(x_i) \cdot log_2(p(x_i)) \\ &= -[(20/100) \cdot log_2(20/100) + (30/100) \cdot log_2(30/100) \\ &+ (50/100) \cdot log_2(50/100)] = 1.48548 \end{aligned}$$

Intuitively, high output entropy value indicates a more uniform-like distribution of the output stream and a higher probability that novel events will be finally noticed.

2.3.2 Improving Output Entropy

We now give the definition of the most novel subscription of user u at time t.

Definition 2.2. The most novel subscription of user u at time t is the subscription with the lowest novelty score of the subscriptions of the user.

Let N be the number of events delivered to the user u and e_t be the $N+1$ event that is delivered to the user. Also, let $H(X, N)$ be the entropy of the first N events and $H(X, N+1)$ be the value of the entropy after the delivery of the e_t event.

Theorem 2.3. *The difference between $H(X, N)$ and $H(X, N+1)$ is maximized if the event e_t matches the most novel subscription of the user u.*

Proof can be found in the Appendix. Intuitively, the delivery of an event that matches the most novel subscription of the user, leads to the maximum possible increase of the output entropy.

Let $L(s) = \{s_1, s_2, \ldots, s_m\}$ be the list of the m subscriptions of the user u sorted in a descending order from the most popular subscription to the least popular one. Let r_{s_1}, $r_{s_2}, \ldots, r_{s_m}$ be the corresponding matching rates with $r_{s_1} > r_{s_2} > \ldots > r_{s_m}$ and d_{s_1}, d_{s_2}, $\ldots$, d_{s_m} be the corresponding target delivery rates of events the match each subscription. Also, let $|O_t(u)|$ be the number of events delivered to the user u up to the arrival of the matching event e_t. We aim at improving the entropy of the delivered events, namely we want the target delivery rates of the subscriptions to be equal and independent from the corresponding matching rates. Next, we present the minimum r_{min} and the maximum filtering rate r_{max} per user u in order to maximize the entropy of delivered events.

Observation 2.1. *The minimum target delivery rate per subscription is equal to $\frac{1}{|O_t(u)|}$, namely one matching event per subscription is delivered. Consequently, the minimum filtering rate r_{min} per user u is equal to $\frac{m}{|O_t(u)|}$.*

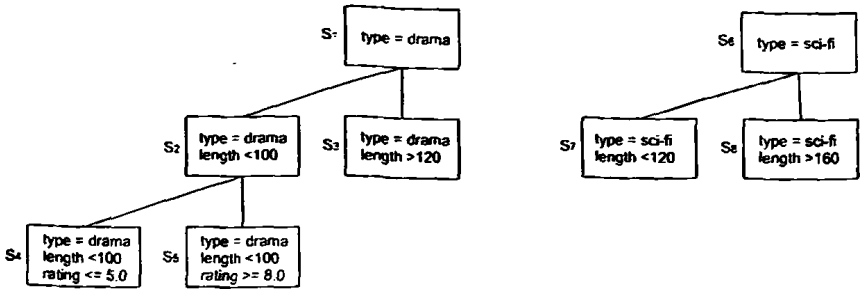
Observation 2.2. *The maximum target delivery rate per subscription is equal to r_{s_m}, namely each subscription has a delivery rate equal to the matching rate of the least popular one (i.e. the most novel). Consequently, the maximum filtering rate r_{max} per user u is equal to $m \cdot r_{s_m}$.*

Note that if the filtering rate r per user u is less than r_{min} or more than r_{max} the subscriptions will not achieve an equal delivery share and thus the entropy will not be maximized.

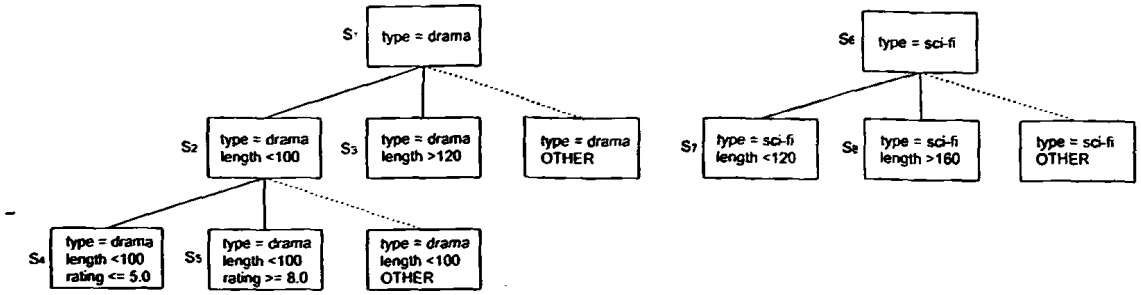
2.3.3 Other Measures

The fairness measure [3] is used in network engineering to determine whether users or applications receive a fair share of system resources. In our work, we use fairness to determine whether users receive a fair share of events per matching subscription.

The fairness F of a stream of events delivered to the user u that match one of the subscriptions $S(u) = \{s_1, s_2, \ldots, s_n\}$ of u is equal to:



(a) Initial Subscription Graph



(b) Subscription Graph with dummy subscriptions

Figure 2.1: Subscription graph

$$F(s_1, s_2, ..., s_n) = \frac{(\sum_{i=1}^{n} O_{s_i}(u))^2}{n \cdot \sum_{i=1}^{n} O_{s_i}^2(u)} \tag{2.6}$$

The value of the fairness ranges from $\frac{1}{n}$, which is the worst case to 1, which is the best case and it is maximized when all subscriptions of the user match the same number of delivered events. Using the above scenario for calculating entropy, the corresponding value of fairness is:

$$F(s_1, s_2, s_3) = \frac{(20 + 30 + 50)^2}{3 \cdot (20^2 + 30^2 + 50^2)} = 0.87719$$

2.4 Subscription Subsumption

Our model supports subscription subsumption or coverage. We say that subscription s_2 is more specific than subscription s_1 if and only if, $\forall$ event e such that $match(e, s_2) = 1$, it holds that $match(e, s_1) = 1$. This may happen for example when s_1 is a subscription expressing interest in a general topic (e.g., a drama movie) and s_2 is a subscription expressing interest in a most specific one (e.g., a drama movie with duration less than 120

min).

Suppose that a user u has submitted to the publish/subscribe system the subscriptions that Figure 2.1(a) shows. Let e_t be the event {type = drama, length = 90, rating = 5.0} that matches subscription s_4. Clearly, e_t matches subscriptions s_1, s_2 and s_3.

The first question that arrises is the novelty score of which subscription should be updated (reduced), if event e_t is delivered. If e_t is delivered then the novelty scores of all subscriptions that match it are updated. Intuitively, if e_t is delivered the novelty of all subscriptions in the path from the root to the most specific matching subscription is reduced, as the part {type = drama} is common in all matching subscriptions.

The next question that arrises is the way that we compute the entropy of the delivered events. We compute the entropy at the most specific subscription that matches an event, if it is delivered. Continuing the previous example, if e_t is delivered, we compute the entropy at the most specific subscription that matches e_t, which is subscription s_3. In case, that the node of the most specific subscription that matches an event has children nodes, then the system adds some dummy subscriptions (linked to the most general subscriptions with dotted lines in Fig. 2.1(b)). For example, if event e_t is {type = drama} then the entropy is computed at the dummy subscription {type = drama, OTHER}.

When a delivered event matches subscriptions that each is not more specific than the other then the novelty score of each subscription is updated. For example, let e_t be the event {type = drama AND type = sci-fi}. Notice that the subscriptions are disjoint, namely {type = drama} $\cap$ {type = sci-fi} = $\emptyset$. In this case, the novelty scores of both subscriptions are updated.

CHAPTER 3

NOVELTY-BASED EVENT FILTERING

3.1 Relevance Filtering

In this section, we focus on the use of novelty for enhancing event delivery. Since the input stream of events can be large, instead of delivering all matching events to all users, relevance filtering can be used to select and deliver the most relevant events to each user. The reason for this is twofold: (a) user satisfaction and (b) system performance. From a user perspective, with relevance filtering, each user receives a subset of the most representative events, instead of being overwhelmed with all matching ones. From a system perspective, filtering events based on novelty decreases the system load and network traffic.

In this work, we introduce a new way of filtering out non-relevant events that is called *novelty-based event filtering*. We consider novelty-based event filtering as a second-stage step that follows the relevance filtering of the event matching process. This new notion of filtering is both history-based and user-based; only events that were delivered to the user u in the past affect the decision of delivering new events to u.

Suppose that the publish/subscribe system limits the subset of events that each user receives with a maximum delivery rate per user threshold $r_{TH}(u) \in [0, 1]$. Our objective is to maximize the per user information gain by delivering events that cover fairly the subscriptions of the user, that is all subscriptions achieve equal number of delivered events. In this work, we quantify the quality of the information content that user receives by using entropy. High entropy values express high information gains received by the user as they

indicate a more uniform-like distribution of delivered events. We now formally state the *Subsequence Selection Problem*.

Subsequence Selection Problem Given user u, the set $S(u)$ of the subscriptions of u, the sequence $M(u)$ of matching events, a maximum delivery threshold $r_{TH}(u) \in [0,1]$ and a period P of time, select a subsequence $O(u)$ of events with size $k = r_{TH}(u) \cdot P$ and deliver it to user u such that the entropy H of $O(u)$ is maximized:

$$O(u)_k^* = argmax_{O(u)_k \text{ subsequence of } M(u)} H(O(u)_k) \tag{3.1}$$

3.2 Algorithms

In this section, we present our novelty-aware algorithms. Our algorithms aim at improving the entropy of the input stream of events that match at least one of the subscriptions of the user by filtering out non-novel events. We propose a *threshold-based algorithm* that uses an adaptive per user threshold; a matching event that has a novelty score above this threshold is delivered to the user; else it is discarded. Further, we propose a *novelty-biased sampling algorithm* that uses a sampling rate per subscription of the user and filters out matching events that with high probability are non-novel. Our algorithms work only in an *on-line* mode, as at the time of each event arrival, the decision whether it is novel or not is made without knowing about the complete event stream.

3.2.1 Lazy vs Eager Mode

We propose a lazy and an eager mode that are both applicable to the threshold-based and the novelty-biased sampling algorithm. The major difference between the lazy mode and eager one is the time the threshold/sampling rate is adjusted. In the first case, we assume that the stream of matching events $M(u)$ is partitioned in disjoint periods of P events. The threshold/sampling rate are adjusted at the end of each period. In the second case, when a matching event arrives, it is inserted in a sliding window of size W. The sliding window is tuple-based; when a new event e_t arrives, the least recent event (i.e. the event that has been inserted in the window W units before e_t) slides out from the window. In the eager update case, the threshold/sampling rate are adjusted when a new matching event is inserted in the window.

When the threshold-based algorithm uses the eager mode, a considerable computational overhead is induced. In this mode, the computation of the threshold is based on the last W matching events of the user u. As we will shortly see, when a new event arrives, the threshold is updated. This update requires sorting the novelty scores of the events in the window with each arrival of a matching event.

3.2.2 Rate-based vs Interval-based Scoring Mode

In section 2.2, we have discerned two major ways of computing the novelty of a subscription; a rate-based scoring mode according to the delivery rate model and an interval-based one according to the delivery interval model. We remind that the rate-based scoring mode is based on a per subscription delivery rate; the more events are delivered for a subscription, the less novel the subscription is. The interval-based scoring mode is based on the distance of the current event that matches subscription s, from the previous event in the stream that matches subscription s. The bigger the distance, the more novel subscription s is. Our scoring modes are combined with both the threshold-based and the the novelty-biased sampling algorithm.

3.2.3 Threshold-based Algorithm

Algorithm 1 Threshold-based Algorithm - Lazy Mode

Input: A sequence of matching events $M(u)$, the set $S(u)$ of the subscriptions of u, a maximum delivery rate $r_{TH}(u)$ and a period length P

Output: A sequence of delivered events $O(u)$

1: **begin**
2: $thr(u) \leftarrow 0$
3: $delivered \leftarrow 0$
4: $k \leftarrow r_{TH}(u) \cdot P$
5: $L \leftarrow$ empty list
6: **for all** $e_t \in M(u)$ **do**
7: $s \leftarrow$ the subscription matched by e_t
8: **if** $novel(s, u, t) \geq thr(u)$ and $delivered \leq k$ **then**
9: deliver e_t to u
10: $delivered \leftarrow delivered + 1$
11: **end if**
12: insert $novel(s, u, t)$ to L
13: **if** $(t \ \% \ P) = 0$ **then**
14: sort L in descending order
15: $thr(u) \leftarrow k^{th}$ element of L
16: clear L
17: **end if**
18: **end for**
19: **end**

Now, we present the threshold-based algorithm. When a new event e_t that arrives at the system matches at least one of the subscriptions of the user u, the algorithm reaches a binary decision whether the event e_t is either novel or not for u. More specifically, the score of the most novel of the matching subscriptions (i.e. that has the highest novelty

score) is compared against a novelty-aware per user threshold. In case that the score is above the threshold, the event is delivered to user along with a score that indicates how much novel the event e_t for user u is; else it is discarded. The score of the delivered event is equal to the score of the most novel subscription that it matches.

The total number of delivered events per period/window is limited by a maximum delivery rate $r_{TH}(u)$. In case the rate of the total delivered events becomes equal to $r_{TH}(u)$, then no more events are delivered in the current period/window. The threshold is adjusted at the end of each period/window and is equal to the k^{th} highest novelty score of the previous period/window, where $k = r_{TH}(u) \cdot P$ and $k = r_{TH}(u) \cdot W$ respectively. Algorithm 1 shows the pseudocode of the algorithm with the lazy mode.

3.2.4 Novelty-biased Sampling Algorithm

We propose a novelty-biased sampling algorithm that *online* filters out non-novel events for the user u. For each subscription of the user a sampling rate is computed. Let e_t be the current event that matches at least one of the subscriptions of user u. The more novel the subscription s that matches the event e_t is, the highest the probability that e_t will be delivered to the user. Intuitively, the highest the value of the sampling rate of subscription S of user u is, the highest the probability that the event e_t that matches s will be delivered to the user.

The computation of the sampling probability, when the rate-based scoring is used, is based on the following: Let $r_{TH}(u)$ be the maximum delivery rate per user u, $S(u)$ be the set of the subscriptions of the user u, s be a subscription in $S(u)$. Also, let e be an event and $P_{match}(s)$ be the probability that event e matches subscription s. Intuitively, we want each subscription of the user to achieve the same delivery rate, that is an equal number of delivered events, thus $\frac{r_{TH}(u)}{|S(u)|}$, where $|S(u)|$ is the number of the subscriptions of the user.

Consequently, the sampling probability per subscription s is equal to:

$$P_{sampl}(s) = \begin{cases} 1, \text{ if } P_{match}(s) < \frac{r_{TH}(u)}{|S(u)|} \\ \frac{r_{TH}(u)}{|S(u)|} \cdot \frac{|M(u)|}{|M_s(u)|}, \text{ otherwise} \end{cases} \tag{3.2}$$

Proof can be found in the Appendix.

When the interval-based scoring is used, the sampling rate is equal to:

$$P_{sampl}(s) = \begin{cases} 1, \text{ if } P_{match}(s) < \frac{r_{TH}(u)}{|S(u)|} \\ \frac{r_{TH}(u) \cdot \sum_{i=1}^{|M_s(u)|} Dist(e_n^s, e_m^s)}{|S(u)| \cdot |M_s(u)|}, \text{ otherwise} \end{cases} \tag{3.3}$$

where $\sum_{i=1}^{M(u)} Dist(e_n^s, e_m^s)$ is the addition of the distances of events e_n and e_m that match the same subscription s and $|M_s(u)|$ is the number of events that match subscription s. The distance is measured in terms of matching events. n denotes the sequence number of matching event e_n, that is $n-1$ events were matched before e_n, and m denotes the sequence

number of matching event e_m. Then, the distance $Dist(e_n^s, e_m^s)$ is equal to $m - n$, with $m > n$. Note that $\frac{\sum_{i=1}^{|M_s(u)|} Dist(e_n^s, e_m^s)}{|M_s(u)|}$ is equal to the average interarrival distance of events that match subscription s. Intuitively, the bigger the interarrival distance of s, the highest the probability that an event that matches subscription s will be delivered.

Algorithm 2 Novelty-biased Sampling Algorithm - Lazy Mode

Input: A sequence of matching events $M(u)$, the set $S(u)$ of the subscriptions of user u, a maximum delivery rate $r_{TH}(u)$ and a period length P

Output: A sequence of delivered events $O(u)$

```
1: begin
2: delivered ← 0
3: k ← r_TH(u) · P
4: for all e_t ∈ M(u) do
5:    choose a number x uniformly at random in [0,1]
6:    s ← the subscription matched by e_t
7:    if P_sampl(s) ≤ x and delivered ≤ k then
8:       deliver e_t to u
9:       delivered ← delivered + 1
10:   end if
11:   if (t % P) = 0 then
12:      for all s ∈ S(u) do
13:         if |M_s(u)|/|M(u)| ¡ r_TH(u)/|S(u)| then
14:            P_sampl(s) = 1
15:         else
16:            P_sampl(s) ← r_TH(u)/|S(u)| · |M(u)|/|M_s(u)|
17:         end if
18:      end for
19:   end if
20: end for
21: end
```

When a new event e_t arrives that matches at least one subscription of the user u, we generate a number x uniformly at random. If x has a lower value than the sampling rate of the most novel subscription s that matches e_t and the number of delivered events is less than $k = r_{TH}(u) \cdot P$ then it is delivered to the user u. Intuitively, an event that matches a subscription which was rarely matched, has a high probability to reach the user. We present the pseudocode of the novelty-biased sampling algorithm when the rate-based scoring mode and the lazy update mode is used (Algorithm 2). We omit the pseudocode when the interval-based scoring mode of the algorithm is used as it differs from the rate-based scoring one in the way that the subscription sampling rate is computed. Also, we omit the pseudocode when the eager mode is used as it differs from the lazy one only in the

time that the subscription sampling rate is computed as we have previously illustrated.

Our sampling algorithm is both space and time efficient compared to our threshold-based algorithm. It does not require maintaining and sorting a list of scores of events per period/window. However, it has some drawbacks. Consider a scenario where a popular (i.e. highly matched) subscription s is not matched in certain periods of time. Recall that the computation of $P_{sampl}(s)$ is based on sharing equally the maximum total delivery rate $r_{TH}(u)$ among the subscriptions of the user. Assigning a fraction of $r_{TH}(u)$ to subscription s in periods when not matched, results in limiting the number of events delivered to the user, as this fraction will not be used. Further, the probability $P_{sampl}(s)$ in periods when s not matched takes high values, thus when s again highly matched, a huge number of its matching events will be delivered to the user. Consequently, the entropy of delivered events will deteriorate.

Due to the above reasons, the threshold-based algorithm is a) resistant in radical changes of the distribution of the input stream and b) achieves a higher total delivery rate than the sampling algorithm when changes in the distribution of the input occur as it does not take into account the number of the subscriptions of the user.

3.3 Aging

The computation of the subscription novelty score can be based on the history of previous delivered events. To incorporate the former delivery history into the novelty score of a subscription, we consider an aging factor γ (i.e. a weight factor) that takes values in [0,1].

Let $novel(s, u, t)$ be the novelty score of subscription s of user u when event e_t is delivered and $novel(s, u, t')$ be the novelty score of subscription s of user u when event e'_t is delivered with $t' < t$. The novelty score $novel(s, u, t)$ of subscription when event e_t is delivered, when the aging factor is used, is computing according to the following formula:

$$novel(s, u, t) = \gamma \cdot novel(s, u, t) + (1 - \gamma) \cdot novel(s, u, t') \tag{3.4}$$

Note that when γ is equal to 1, no aging is introduced. Also, when γ is equal to 0, the current novelty score is equal to the previous one, that is the score is not updated.

3.4 Distributed Setting

We have presented the algorithms with the assumption that we have a centralized setting, namely each subscription of the user is at the same node of the publish/subscribe system. Now, we consider the case that the subscriptions of the user are distributed across a number of nodes of the publish/subscribe system. We discuss the effect of the distribution of the subscriptions on the threshold-based and the novelty-biased sampling algorithm.

3.4.1 Threshold-based Algorithm

In the rate-based scoring mode, the computation of the novelty score of the subscription s is based on the number of events that match s and are delivered to the user and the number of events that match at least one of the subscriptions of the user. In the interval-based scoring mode, the novelty score is based on the time measured in terms of matching events between two consecutive deliveries of events that match s. Both variations are based on the total number of events that match at least one subscription of the user. Each subscription maintains the number of its matching events. Consequently, the computation of the total matching events per user requires the communication of the nodes that maintain the subscriptions of the user with each update of the subscription novelty score.

Further, the adjustment of the threshold requires sorting the novelty scores of the events of the previous period/window. Since each event score is stored locally (i.e. in the node of the matching subscription), a distributed top-k algorithm is needed for computing the threshold.

3.4.2 Novelty-biased Sampling Algorithm

Both the rate-based and the interval-based scoring mode require the total number of matching events of the user and the number of the active subscriptions of the user. A subscription is active if it has been matched by at least one event. As explained earlier, the total number of matching events per user induces a considerable communication overhead. Also, the number of active subscriptions of the user, requires the communication between the nodes that maintain the subscriptions of the user. The more often is the communication, the more accurate is the sampling algorithm. In our work, the number of active subscriptions of the user is updated periodically.

CHAPTER 4

EXPERIMENTAL EVALUATION

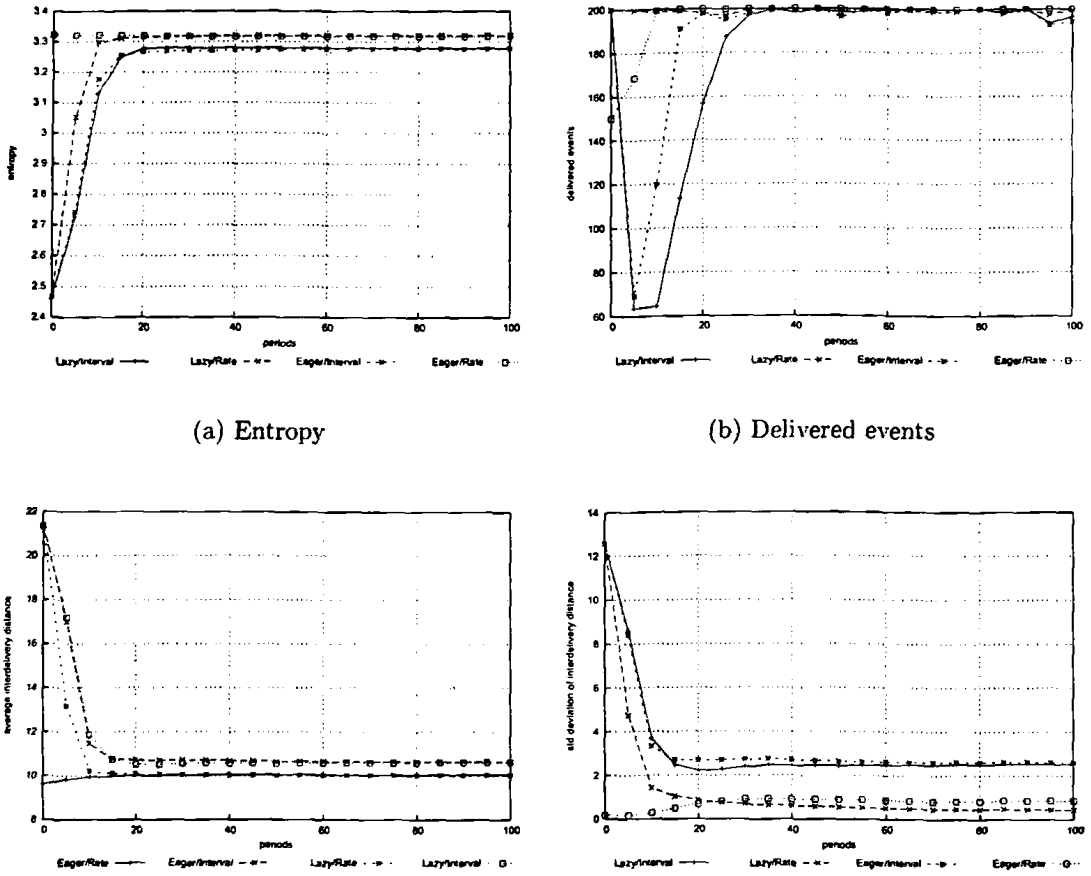
In this section, we present the results of the experimental evaluation of the performance of our novelty-aware algorithms. We present results for both synthetic and real data. To evaluate our approach, we have extended the SIENA notification service with our novelty functionality.

4.1 Synthetic Data

For the following experiments, we generate different input streams, each of which consists of 100000 events. The popularity of the events follows a zipf distribution to mimic the distribution that follow real web data [4]. For each user, we generate a number of 10 mutually exclusive subscriptions, such that all published events of our scenarios match one of the subscriptions of the user. Also, we use an aging factor that influences the novelty subscription score. The aging factor that is equal to 1 indicates no aging. Our input parameters are summarized in Table 4.1.

Table 4.1: Input parameters

Description	Range	Default
input stream (S)		100000
# size of period/window (P)	100,1000	1000
maximum rate (max_{rate})		0.2
aging factor (γ)	0.1,0.5,1.0	1.0
# of subscriptions (N)		10
# of users (U)		1
event distribution	zipf, α = 0.75,1.25	1.25



(a) Entropy

(b) Delivered events

(c) Average interdelivery distance

(d) Standard deviation of interdelivery distance

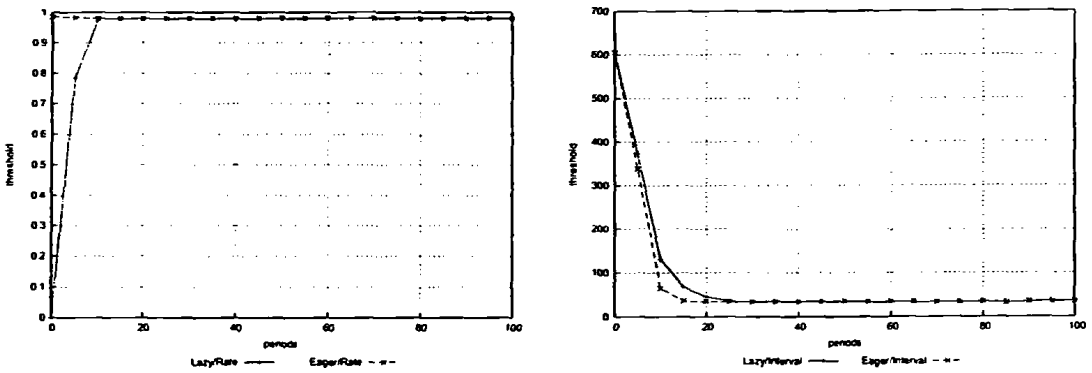
Figure 4.1: Threshold-based Algorithm

4.1.1 Threshold-based Algorithm

First, the goal is to compare the effectiveness of the four variants of the threshold-based algorithm in terms of the achieved output entropy, the average interdelivery distance, the standard deviation of the interdelivery distance and the number of events delivered to the user (Figure 4.1). The events follow a zipf distribution with skew $\alpha = 1.25$.

Eager vs Lazy Mode. Figure 4.1(a) shows the output entropy versus the time measured in periods of events. We see that both the eager and the lazy update mode achieve the same output entropy. Recall, however, that the eager mode induces a considerable computational overhead. Consequently, both update modes are equivalently effective, but it is more efficient to apply the lazy mode to the threshold-based algorithm. Figure 4.1(b) shows the number of events delivered to the user versus the time measured in periods of events. We see that in all approaches, the number of events delivered to the user converges to the maximum delivery rate per user threshold r that is equal to 0.2.

Rate-based vs Interval-based Scoring Mode. In Figure 4.1(a), we see that the rate-



(a) Rate-based Scoring

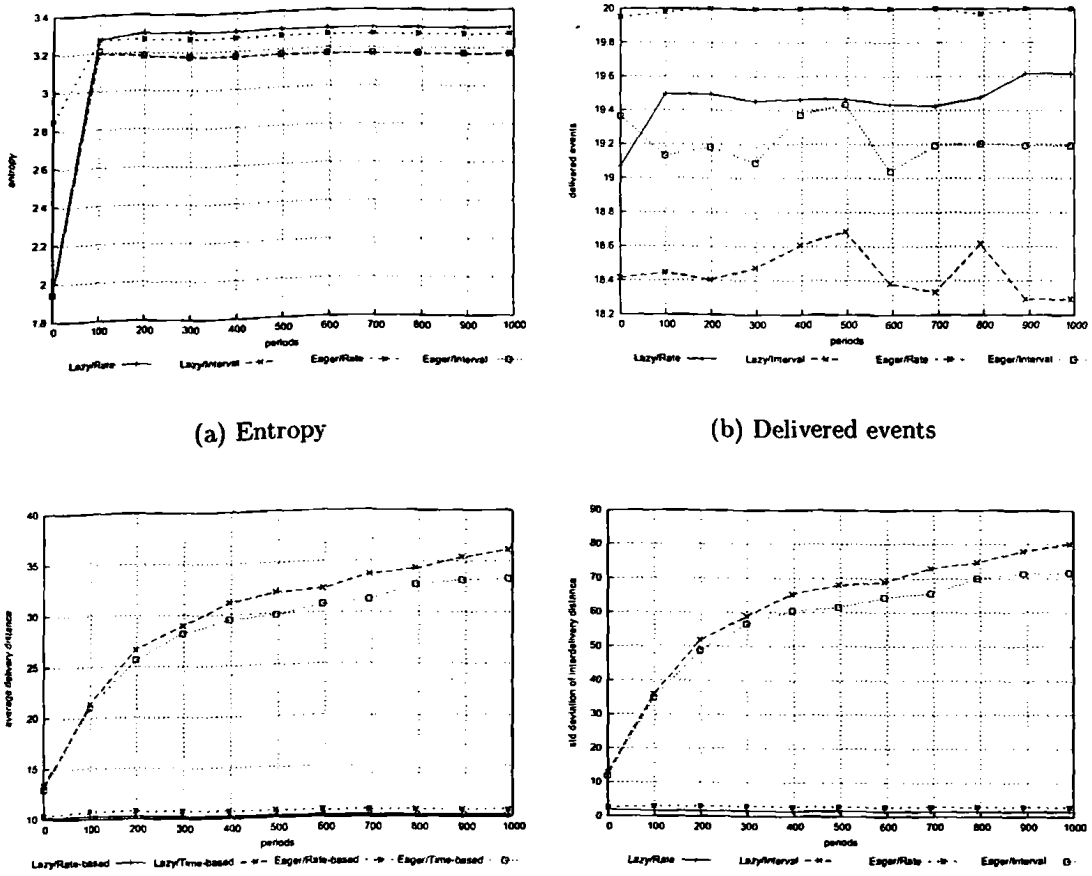
(b) Interval-based Scoring

Figure 4.2: Threshold

based scoring mode outperforms the interval-based one in terms of the achieved entropy for both the eager and the lazy mode. This occurs because the rate-based scoring results in a more accurate ranking that deploys the history of delivered events and adapts more effectively to the distribution of the input stream. This becomes more clear, when we experiment with the average distance of events that match the some subscription and are delivered to the user (i.e. average interdelivery distance) that Figure 4.1(c) depicts. Intuitively, the optimal average interdelivery distance is equal to the number of the subscriptions of the user (here 10), namely each event is delivered to the user after the arrival of 10 matching events. Figure 4.1(d) shows the standard deviation of the interdelivery distance. We see that the rate-based scoring has a value around 0.0, which is the optimal value, for both the lazy and the eager mode. This indicates that the interdelivery distance for events that match the same subscription has small deviations around the average interdelivery distance, namely we have the ideal interdelivery distance per subscription which is equal to 10.

Threshold. Then, we experiment with the behavior of the threshold. Figure 4.2(a) shows the threshold with the time measured in periods of events when the rate-based scoring mode is used. When the eager mode is used, we witness a fast threshold conversion, whereas when the lazy mode is used, the threshold needs a "warm-up" period to become stable. Figure 4.2(b) shows the behavior of the threshold, when the interval-based scoring is used. Again, we witness faster threshold conversion when the eager mode is used in comparison with the lazy mode.

Period/Window. Finally, we explore the role of the period/window of size P used by the threshold-based algorithm. Recall, that in the eager mode the calculation of the threshold is based on the last P matching events (sliding window), whereas in the lazy mode the threshold is updated over the P matching events of the previous period. We set $P = 100$ events and evaluate the algorithm in terms of the output entropy, the average interdelivery distance, the standard deviation of the interdelivery distance and the



(a) Entropy

(b) Delivered events

(c) Average interdelivery distance

(d) Standard deviation of interdelivery distance

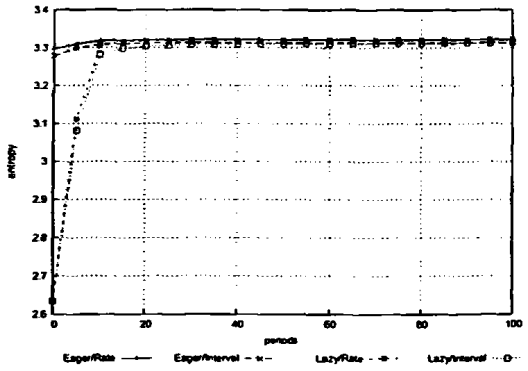
Figure 4.3: Period

number of delivered events. In Figure 4.3(a), we see that a smaller period size results in a lower value of the output entropy for the interval-based scoring mode. Further, we see that the rate-based scoring mode outperforms the interval-based one in terms of the average interdelivery distance (Figure 4.3(c)), the standard deviation of the interdelivery distance (Figure 4.3(d))and the number of delivered events (Figure 4.3(b)).

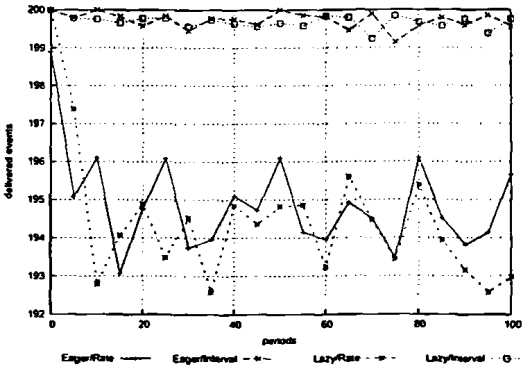
4.1.2 Novelty-biased Sampling Algorithm

We evaluate the effectiveness of the four variants of the novelty-biased sampling algorithm in terms of the achieved output entropy, the average interdelivery distance, the standard deviation of the interdelivery distance and the number of events delivered to the user (Figure 4.4).

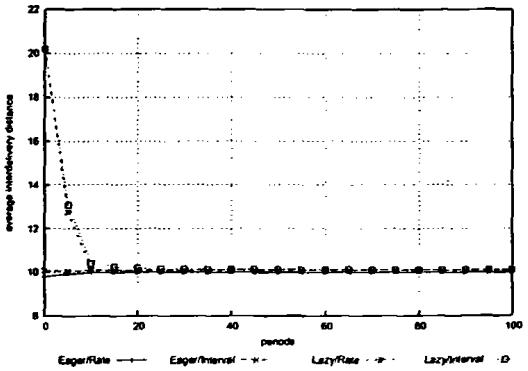
Eager vs Lazy Mode. In Figure 4.4(a), we see that the output entropy converges to the same final value for both modes. We witness the same behavior with the standard deviation of the interdelivery distance in Figure 4.4(d) and the average interdelivery dis-



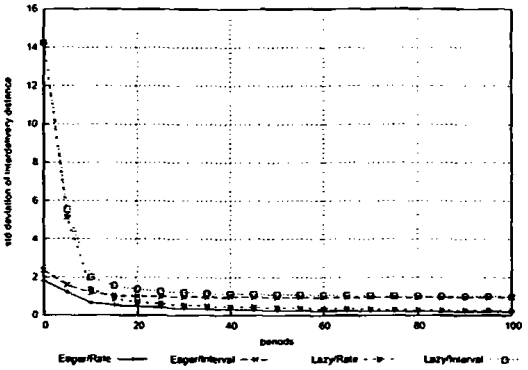
(a) Entropy



(b) Delivered events



(c) Average interdelivery distance

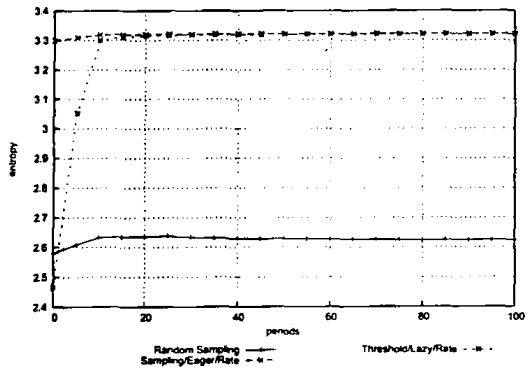


(d) Standard deviation of interdelivery distance

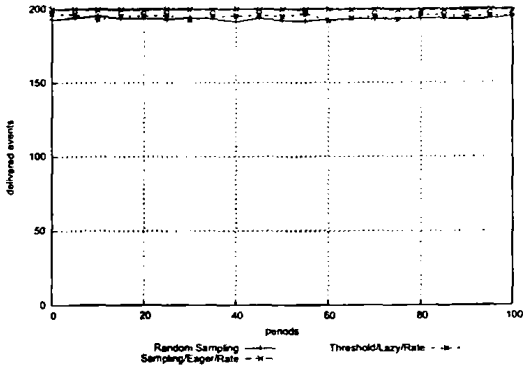
Figure 4.4: Novelty-biased Sampling Algorithm

tance in Figure 4.4(c). The main difference between the two modes is the time that the novelty-biased sampling algorithm converges to the final value of the entropy, the average interdelivery distance and the standard deviation of the interdelivery distance; the usage of the eager update mode results in a faster conversion.

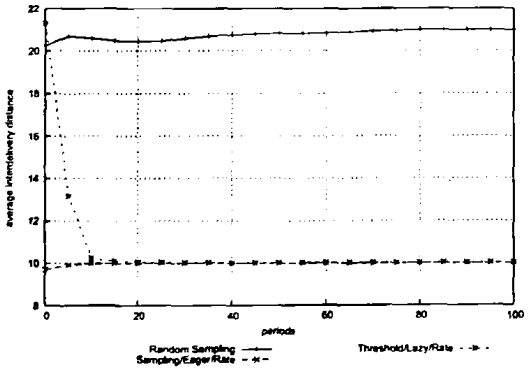
Rate-based vs Interval-based Scoring Mode. In Figure 4.4(c), we see that both modes result in the same values of the average interdelivery distance. In Figure 4.4(a), we witness small variations of the achieved entropy between the two scoring modes; rate-based scoring seems to be a bit more effective. But, in Figure 4.4(d), we clearly see that rate-based scoring mode results in a lower value of the standard deviation than the interval-based one for both the eager and the lazy mode. Regarding the number of delivered events, Figure 4.4(b) shows that when the interval-based scoring is used, more events are delivered to the user.



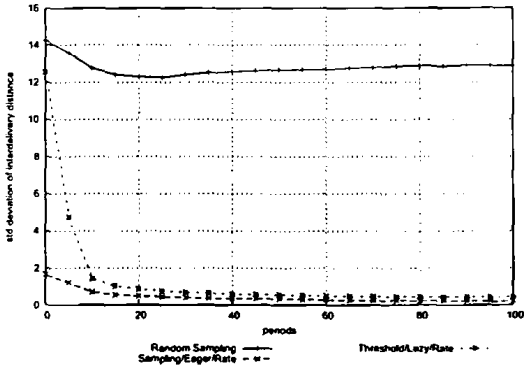
(a) Entropy



(b) Delivered events



(c) Average interdelivery distance

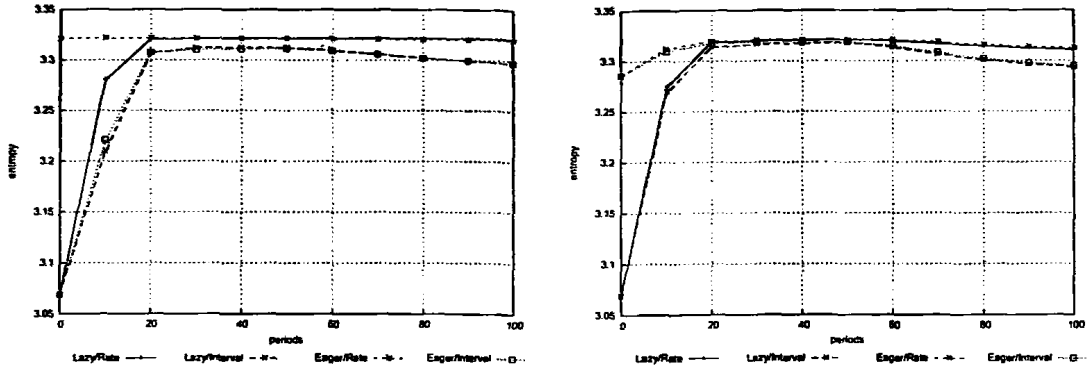


(d) Standard deviation of interdelivery distance

Figure 4.5: Threshold-based vs Novelty-biased Sampling Algorithm

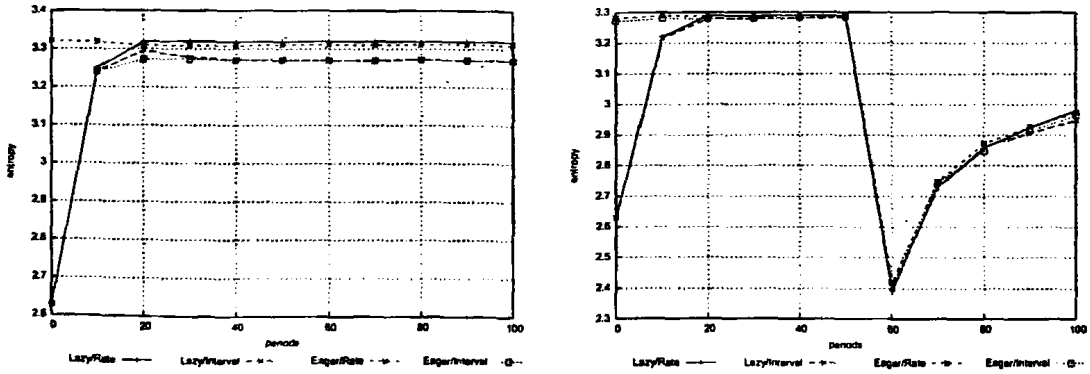
4.1.3 Threshold-based vs Novelty-biased Sampling Algorithm

Next, we compare the most effective variant of the threshold-based algorithm and the one of the novelty-biased sampling algorithm against a baseline approach ("random-sampling"), where a percentage of the matching events per period are randomly selected and delivered to the user. We see that our algorithms outperform the random sampling alternative in terms of the output entropy (Figure 4.5(a)), the average interdelivery distance (Figure 4.5(c)) and the standard deviation of the interdelivery distance (Figure 4.5(d)). Also, we see that the threshold-based and novelty-biased sampling algorithm are equally effective in terms of the achieved entropy and the average interdelivery distance. However, the sampling algorithm outperforms the threshold-based one in terms of the standard deviation of the interdelivery distance. Regarding the number of delivered events, Figure 4.5(b) depicts that when the threshold-based algorithm is used, a bit more events are delivered to the user.



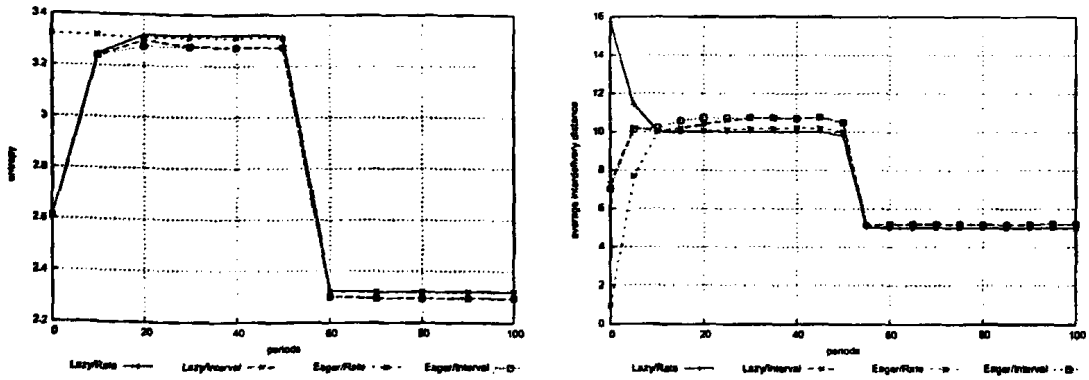
(a) Threshold-based Algorithm (b) Novelty-biased sampling Algorithm

Figure 4.6: "Skew-transient" Scenario



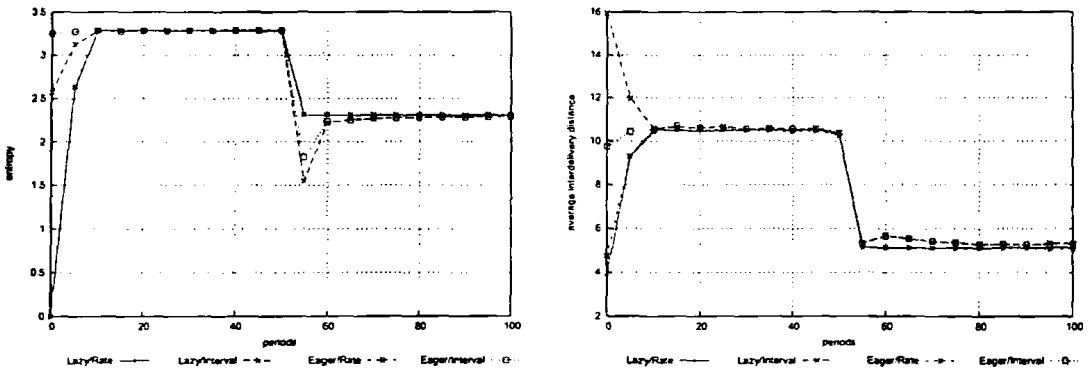
(a) Threshold-based Algorithm (b) Novelty-biased sampling Algorithm

Figure 4.7: "Skew-reverse" Scenario



(a) Entropy (b) Average interdelivery distance

Figure 4.8: "Insertion/Deletion" Scenario - Threshold-based Algorithm



(a) Entropy

(b) Average interdelivery distance

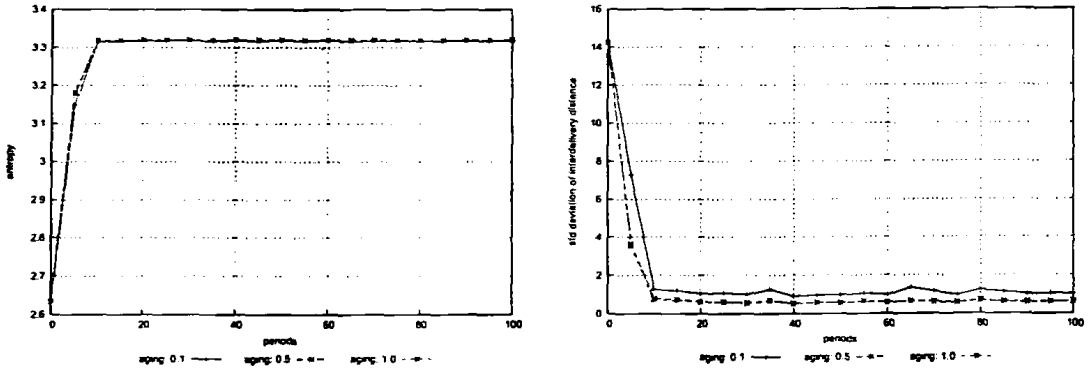
Figure 4.9: "Insertion/Deletion" Scenario - Novelty-biased Sampling Algorithm

4.1.4 Transient data distribution

"Skew-transient" scenario. Next, we evaluate our algorithms when the distribution of the input stream of events changes. We consider two scenarios where the popularity of specific events of the input stream changes over time. In the first "Skew-transient" scenario, we change the skew of the distribution of the input stream. More specifically, the first half of the input stream follows a zipf distribution with $\alpha = 0.75$, whereas the second one a zipf distribution with $\alpha = 1.25$. Figure 4.6 shows that the effectiveness of almost all of our algorithms measured in terms of entropy, remains unaffected as the skew of the distribution changes. However, as the skew increases, we witness small variations of the measured entropy when the time-based scoring mode is used.

"Skew-reverse" scenario. In the second "Skew-reverse" scenario, we reverse the popularity of subscriptions of the user. More specifically, in the second half of the input stream, the most popular subscription becomes the less popular one, the second most popular one becomes the second less popular one and so on. Figure 4.7 depicts the measured entropy with the time in periods of events. We see that the effeciveness of the different variants of the threshold-based algorithm remains unaffected as the distribution changes. However, we see that the novelty-biased sampling algorithm is very sensitive to the changes of the input distribution. To see why, take a subscription that is rarely matched in the past and thus converges to a high sampling rate. As the input distribution changes radically in the middle of the input stream, the less frequently matched subscription becomes the most popular one. Consequently, events that match the most popular subscription are delivered to the user with high probability and thus the value of the output entropy deteriorates.

"Insertion/Deletion" scenario. Finally, we consider an "Insertion/Deletion" scenario, where a user unsubscribes from some of their subscriptions. More specifically, in the first half of the input stream the user has submitted 10 subscriptions, whereas in the second half he has unsubscribed from 5 of them. In Figure 4.8(a), we see that the considering



(a) Entropy

(b) Standard deviation of interdelivery distance

Figure 4.10: Rate-based Scoring Mode - Threshold-based Algorithm

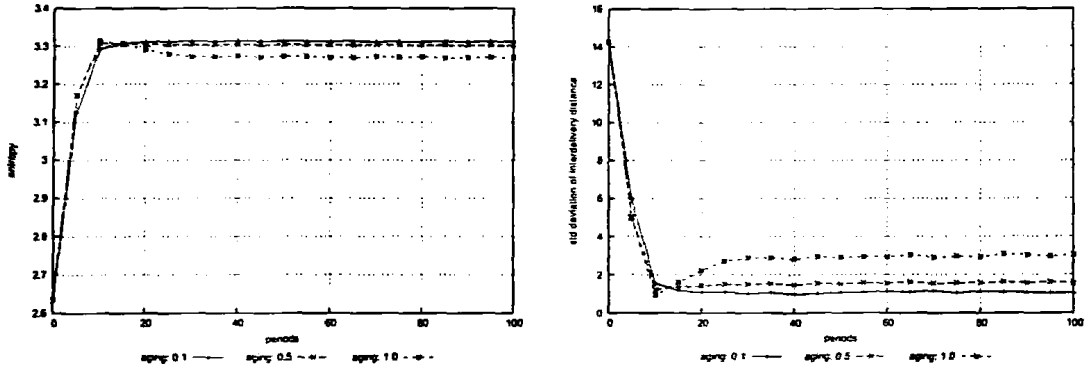
scenario affects the output entropy which deteriorates. Note, however, that the entropy in the first half of the input stream is measured over the 10 matching subscriptions of the user, whereas in the second half over the 5 matching subscriptions of the user. Consequently, we see that in this scenario, the threshold-based algorithm not only retains its effectiveness but also adapts quickly to the change of the number of subscriptions of the user. This is also confirmed in Figure 4.8(b), where we see that after the change in the number of subscriptions, the average interdelivery distance is equal to 5.

As for the novelty-biased sampling algorithm, Figure 4.9 depicts that the both the entropy and the average interdelivery distance are affected by the change in the number of subscriptions, especially when interval-based scoring mode is used. This is due to the fact that the computation of the sampling rate per subscription is based on the number of active subscriptions of the user, namely the subscriptions that have been matched by at least one event. In the second half of the input stream, the number of matching subscriptions is equal to 5. Recall, that the novelty-sampling algorithm, periodically, updates the number of the active subscriptions of the user. This periodic update has an effect on the measured entropy and average interdelivery distance.

4.1.5 Aging

Next, we evaluate our algorithms when an aging factor γ is used that takes values in [0,1]. An aging factor equal to 1 indicates no aging, whereas an aging factor equal to 0 indicates that the novelty score of the subscription is equal to the previous novelty score (i.e. the score is not updated). We report results only for the lazy mode, as in the previous section we have shown that it is equally effective, but more efficient than the eager update mode.

Threshold-based algorithm. First, we report results when the aging factor is used in the threshold-based algorithm.



(a) Entropy

(b) Standard deviation of interdelivery distance

Figure 4.11: Interval-based Scoring Mode - Threshold-based Algorithm

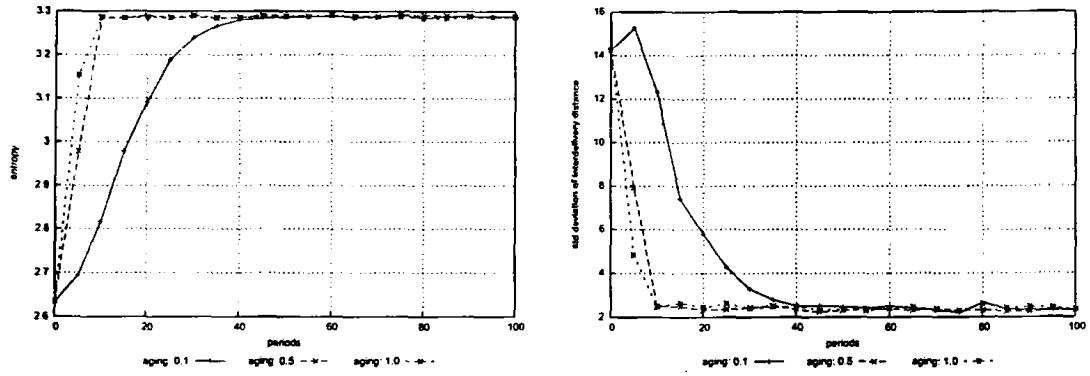
Rate-based Scoring Mode. As expected, using aging with the rate-based scoring mode does not improve the measured entropy as Figure 4.10(a) depicts. This is due to the fact that the rate-based scoring mode is already based on the history of previous delivered events. Also, we see in Figure 4.10(b) that the standard deviation of the interdelivery distance deteriorates a bit when aging is used. Consequently, aging and rate-based scoring mode is not a good combination.

Interval-based Scoring Mode. In the Interval-based scoring mode, as Figure 4.11 depicts, the lower the value of the aging factor γ (i.e. more aging), the higher the improvement of the entropy and the standard deviation of the interdelivery distance. Recall that the novelty score of the event is based on the distance from the previous event that matches the same subscription and is delivered to the user. When aging is used, the computation of the novelty score deploys the former delivery history and leads to better results.

Novelty-biased sampling algorithm. Next, we evaluate the novelty-biased sampling algorithm when aging is used.

Rate-based Scoring Mode. Figure 4.12 depicts the entropy(left) and the standard deviation of the interdelivery distance(right). We see that the more aging is used, the slower is the conversion of the algorithm, namely the slower is the conversion of the sampling rate of each subscription.

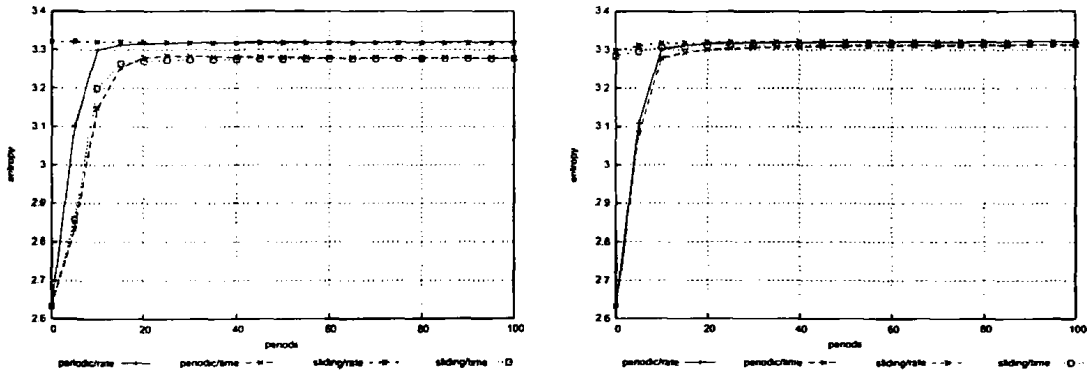
Interval-based Scoring Mode. When the interval-based scoring mode is used, we also see that the usage of aging leads to a slow conversion of the sampling rate of each subscription. We omit the figures that show the effect of the aging when the interval-based scoring mode is used, as the results are equivalent to the rate-based scoring mode.



(a) Entropy

(b) Standard deviation of interdelivery distance

Figure 4.12: Rate-based Scoring Mode - Novelty-biased Sampling Algorithm



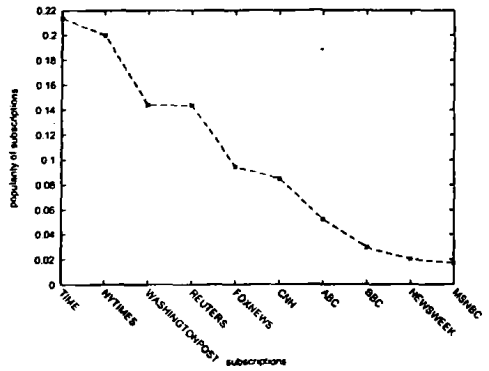
(a) Threshold-based Algorithm

(b) Novelty-biased Sampling Algorithm

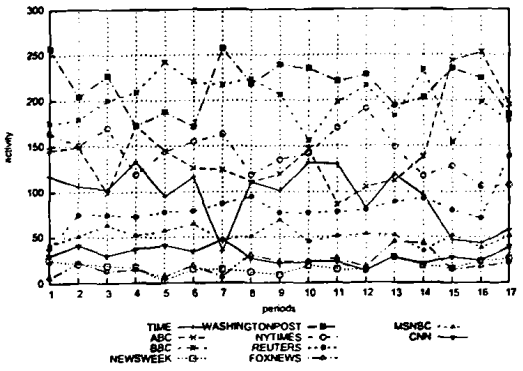
Figure 4.13: Subsumption

4.1.6 Subscription Subsumption

Next, we consider a scenario with subscription subsumption or coverage. The user has submitted 10 subscriptions $\{S_1, S_2, S_3, S_4, S_5, S_6, S_7, S_8, S_9, S_{10}\}$, where S_1 covers S_2 and S_3, S_2 covers S_4 and S_5. Also, S_6 covers S_7 and S_8, S_7 covers S_9 and S_{10}. We have generated events that follow a zipf distribution with skew $\alpha = 1.25$, considering the real-world scenario that most general subscriptions are matched more frequently. Figure 4.13 depicts the high value of the measured entropy when both the threshold-based and the novelty-biased sampling algorithm are used. Consequently, we show that when a subsumption scenario is considered, the choice to use the most novel subscription among other matching subscriptions for threshold/sampling rate comparison leads to effectiveness.

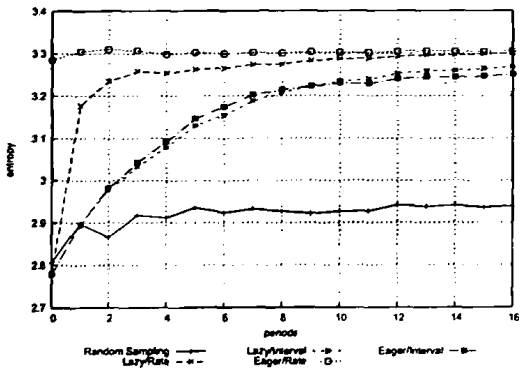


(a) Popularity of subscriptions

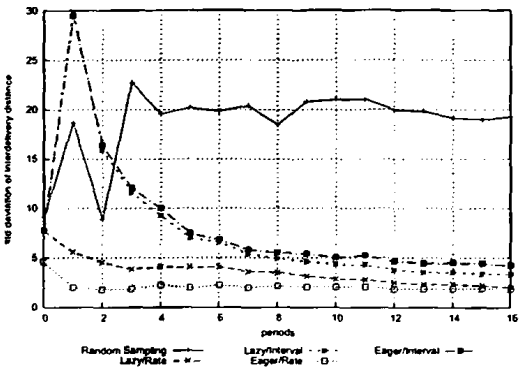


(b) Activity of subscriptions

Figure 4.14: Real dataset



(a) Entropy

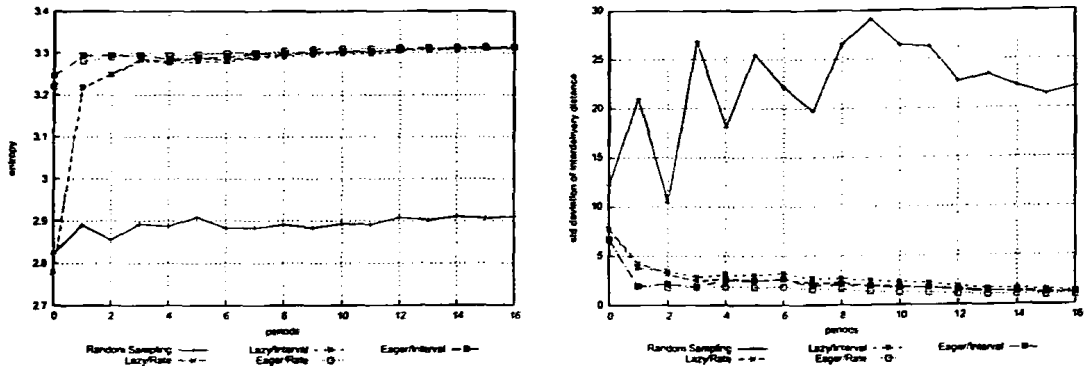


(b) Standard deviation of interdelivery distance

Figure 4.15: Threshold-based Algorithm - Real dataset

4.2 Real Data

In this section, we present results for a real-world dataset. We have used Twitter, a popular networking site, in order to collect a real-time log file. We have followed (i.e. subscribed to) 10 news agencies and collected events (i.e. tweets) that they have generated from 24th January 2011 until 24th March 2011. The log file consists of 17362 entries and its size is approximately 2.0 MB. In this dataset, we have considered the source that generates each event as the subscription that the user has submitted. The popularity of subscriptions (i.e. the percentage of events each source generates) follows a zipf-like distribution as Figure 4.14(a) depicts. Figure 4.14(b) depicts the generation activity of each source versus time measured in periods of events. In the next experiments, we have used an aging factor $\gamma = 0.1$ in the interval scoring mode of both the threshold-based and the novelty-biased sampling algorithm.



(a) Entropy

(b) Standard deviation of interdelivery distance

Figure 4.16: Novelty-biased Sampling Algorithm - Real dataset

Threshold-based algorithm. Figure 4.15(a) shows the entropy with the time in periods of events. We see that the eager mode - rate-based scoring mode variant outperforms all variants in terms of the achieved entropy. This is due to the fact that it adapts more effectively to the nature of the distribution of the real-dataset. Despite the fact that the real dataset follows a zipf-like distribution, we have witnessed that in each period, events do not follow a distribution similar to the distribution of the whole dataset. Some news agencies are in some periods highly productive, whereas in others are not. Also, we see that the interval-based scoring mode achieves the worst entropy values for both the eager and the lazy mode. Figure 4.15(b) that shows the standard deviation of the interdelivery distance illustrates more clearly the differences in the effectiveness between the four variants.

Novelty-biased sampling algorithm. Next, we present results for the novelty-biased sampling algorithm. Figure 4.16(a) shows the entropy, whereas Figure 4.16(b) shows the standard deviation of the interdelivery distance. We see that the lazy mode is less effective than the eager mode. Again, the reason is that the eager mode adapts more accurately to the distribution of the input stream.

CHAPTER 5

RELATED WORK

5.1 Ranked Publish/Subscribe

In this section we present related work about ranked publish/subscribe systems.

The authors of [5] consider the problem of publish/subscribe delivery where a published event is stored or discarded due to limited storage capacity. In their model a subscriber receives the k most relevant publications per subscription within a sliding window of w time units. The relevance between a subscription and a publication is computed using a binary user-defined ranking function.

The proposed solution is based on the fact that at some point in time all publications that belong to the top-k relevant subscriptions of a subscriber will eventually be delivered to her. Publications that belong to the k most relevant publications at the moment of their publishing are called excellent candidates and thus delivered to the subscriber immediately. Publications that can be among the top-k publications at some later time of their publishing with a probability at least Γ are called good candidates and delivered to the subscriber at that time. Publications with probabilities smaller than Γ are discarded.

Each subscription is associated with a sorted queue that stores both excellent and good candidates and is called publications queue. More specifically, excellent candidates are stored at the head of the queue which is of size k and good candidates are stored at the tail of the queue. The minimum size of the tail depends on the value of the given probability Γ, aiming that an adequate number of good candidates (i.e. that have a high probability that will enter the top-k publications) can be stored there.

Another work that focuses on the problem of ranked publish/subscribe is [6]. In this work, the authors propose extending subscriptions to allow users express that some events

are more relevant or interesting to them than others. The proposed model uses preferential subscriptions to compute event scores. Events that match highly preferred subscriptions get higher scores than those that match subscriptions of low preference.

Also, a top-k variant of the problem is introduced where only highly ranked events are delivered to subscribers, along with a number of delivery policies. In periodic delivery, subscribers receive an amount of important events every period of time. In sliding window delivery, the subscriber receives a number of highly ranked events within a tuple-based window of size w. Every time a new event is produced, the top-k event computation restarts. In history-based filtering, the decision of delivering an event is based on previous history, namely whether it is among the last top-k events that the subscriber has already seen.

The model in addition to user preferences takes into account content diversity, where the content of an event differs from other highly ranked events, as a means to increase user satisfaction. Diversity is modelled as the distance between the recently published event and the set of events that have already been delivered to the user.

The authors of [7] consider the problem of ranked publish/subscribe in a reverse way. They aim at recovering the most relevant matching subscriptions for a published event, instead of locating the most relevant events for a subscription of the user. This notion of matching arises naturally in applications related to online-advertising, online-job finding etc. where the stream of incoming users corresponds to events who aim at retrieving only the most important subscriptions based on some predifined criteria.

In the proposed model an event e is represented as a point $(u_1, \ldots, u_d)$ over a d-dimensional space D and a subscription s is represented as a set of intervals $(I_1, \ldots, I_d)$ over space D. A subscription matches exactly an event if the event is fully contained in the hyper-rectangle of the subscription. Relaxed matching can also occur if at least one dimension of the event is contained in the corresponding interval of the subscription. In the case of exact matching, each subscription is associated with a score, whereas in the case of relaxed matching each interval of the subscription is associated with a weight, and the score of the subscription is the sum of the weights of the matching dimensions. The overall goal is to retrieve the top subscriptions ordered by their score.

The authors propose two novel indexing structures to achieve efficient top-k retrieval both in time and space, the Interval R-tree (IR-tree) and the Score-Optimal R-tree (SOPT-R-Tree). The IR-tree is an extension of the typical Interval Tree [8]. Each node of the Interval Tree stores a list of intervals. In the worst case, answering a query may require traversing the entire node list. The IR-tree is based on the idea of replacing this list with an R-tree [9] that indexes the intervals that are stored in a node in order to achieve efficient query times. The SOPT-R-tree data structure is an extension of the scored R-tree. In the scored R-tree, intervals are grouped together by their scores, namely top-scored intervals are grouped together in a tree node, the next lower-scored intervals are grouped together etc. In the worst case, answering a query often leads to visiting all the leaf nodes of the scored R-tree. The SOPT-R-tree tackles this problem with a clever

rearrangement of the indexed intervals.

5.2 Novelty-aware Delivery

Novelty has been used in the context of information retrieval systems as a criterion to rank search results in order to increase user satisfaction. The authors of [10] consider the problem of ranking documents that are relevant to a query submitted by a user. They propose a probabilistic model that considers the relevance of a document in respect with the documents the user has seen before it.

Both information need of the user and information that is present in a document are modelled with information nuggets. Information nuggets are common in the summarization and question answering communities. An information nugget usually represents specific pieces of information. For example a nugget may represent an answer to a question. A document is relevant if it contains a least one nugget that is included in the information need of the user. The computation of relevance depends on the estimation of the probability that the information need of a user u contains nugget n_i, denoted $P(n_i \in u)$ and the probability that a document d contains nugget n_j, denoted $P(n_j \in d)$.

The estimation of $P(n_i \in d)$ is based on a model where a human assesor reaches a binary decision whether a given nugget is included in the document or not. Negative decisions are always condidered correct whereas positive ones may be erroneous with an error probability Γ. The estimation of $P(n_i \in u)$ requires knowledge about user preferences which can be determined impicitly or explicitly by former user behavior and feedback.

The decision whether a new document contains novel information for a user is computed against a list of documents that the user has seen in the past. More specifically, the novelty rank of a document is the probability that the nuggets that it contains are not included in the documents that the user has already seen.

In the context of adapting filtering [11], novelty has been used as a second-stage filtering step that follows the step of relevance filtering. The authors introduce a suite of similarity functions that compare the current document against the content of documents that have been delivered to the user.

The set difference measure assumes that each document is represented as a set of words. The novelty of a new document d is computed by the number of new words that it includes. More specifically, a word w that is frequent in the new document and less frequent in old documents may indicate that d covers novel information. The cosine distance measure assumes that each document is represented by a vector of words. The similarity between a new document d and each of the old documents is computed as a cosine distance between word vectors of documents. High distance indicates high information novelty. Another measure is distributional similarity. It uses the Kullback-Leibler divergence that is a well-known distributional similarity function to compute the novelty of a document against another, given the word distribution of both documents.

Finally, they propose a simple threshold-based technique that discards documents with novelty score below a threshold. The threshold only decreases and when it becomes too low there is not a way of increasing it again.

CHAPTER 6

CONCLUSIONS AND FUTURE WORK

In this thesis, we introduce a new notion of relevance that is based on the former event delivery activity of the user and is called novelty-aware relevance. We define the novelty of a subscription according to the delivery rate model and the delivery interval model and introduce corresponding ways to compute subscription and event novelty scores.

Our overall goal in this work has been to present an efficient and effective filtering mechanism to increase the information gain of the user by forwarding only novelty-aware relevant events. To this end, we introduce two new algorithms that work in on-line mode; a threshold-based algorithm and a novelty-biased sampling algorithm. The first forwards to the user events that have novelty scores above a threshold. The second forwards to the user with high probability events that their matching subscriptions have been rarely matched in the past.

Our focus has been to increase user satisfaction by delivering to the user an equal number of events per subscription. There are many directions for future work. Novelty-aware relevance is only one of the criteria to characterize the importance of an event. Other possible criteria include relevance, source authoritativeness, content diversity and user preferences. How to combine such criteria for effectiveness is a difficult problem. In this work we have explored novelty-aware relevance from a user perspective. Another interesting dimension is exploring novelty from a system perspective mainly regarding system performance. For example, novelty-based filtering can be viewed as a filtering mechanism for groups of users with similar interests mainly targeting in decreasing the overall computational and network overhead by forwarding only novelty-aware relevant events.

BIBLIOGRAPHY

[1] D. Souravlias, M. Drosou, K. Stefanidis, and E. Pitoura. *On Novelty in Publish/Subscribe Delivery*, in DBRank, March 2010.

[2] E.C. Shannon, *Prediction and entropy of printed English*, The Bell System Technical Journal, 30:50-64, January 1951.

[3] R. Jain, D.M. Chiu, and W. Hawe *A Quantitative Measure of Fairness and Discrimination for Resource Allocation in Shared Systems*, DEC Research Report TR-301, 1984.

[4] L. Breslau, P. Cao, L. Fan, G. Phillips, and S. Shenker. *Web caching and zipf-like distributions: evidence and implications*, Proceedings of the Eighteenth Annual Joint Conference of the IEEE Computer and Communications Societes, INFOCOM, 1999.

[5] K. Pripuzik, I.P. Zarko, and K. Aberer, *Top-k/w publish/subscribe: finding k most relevant publications in sliding time window w*, in DEBS, 2008.

[6] M. Drosou, K. Stefanidis, and E. Pitoura, *Preference-Aware Publish/Subscribe Delivery with Diversity*, in DEBS, 2009.

[7] A. Machanavajjhala, E. Vee, M. Garofalakis, and J. Shanmungassundaram *Scalable Ranked Publish/Subscribe*, in PVLDB, 2008.

[8] F. P. Preparata, M. I. Shamos, *Computational Geometry: An Introduction*, Springer-Verlag, 1985.

[9] A. Guttman, *R-Trees: A Dynamic Index Structure for Spatial Searching*, in SIGMOD, 1984.

[10] C. L. A. Clarke, M. Kolla, G. V. Cormack, O. Vechtomova, A. Ashkan, S. Buttcher, and I. MacKinnon, *Novelty and diversity in information retrieval evaluation*, in SIGIR, 2008.

[11] Y. Zhang, J. Callan, and T. Minka, *Novelty and redundancy detection in adaptive filtering*, in SIGIR, 2002.

APPENDIX

We give the proof of Theorem 2.1.

Proof. Both for rate and interval-based novelty ranking, the threshold is defined as the k-th larger score during either the previous period (in lazy mode) or the current window (in eager mode). Let thr_{rate} be the threshold for rate-based ranking, and thr_{inter} be the frequency-based threshold.

For rate-based ranking we have:

$$\begin{aligned} pnovel_{rate}(s,u) &= P\left(novel_{rate}(s,O_{t-1}(u)) > thr_{rate}\right) \\ &= 1 - P\left(novel_{rate}(s,O_{t-1}(u)) \leq thr_{rate}\right) \\ &= 1 - P\left(novel_{rate}(s,O_{t-1}(u)) \leq novel^{(k)}_{rate}(s',O_{t'-1}(u))\right) \\ &= 1 - P\left(1 - \frac{|O_{t-1}(u)|}{|I_{t-1}|} \leq 1 - \frac{|O_{(k)}(u)|}{|I_{(k)}|})\right) \\ &= 1 - P\left(\frac{|O_{t-1}(u)|}{|I_{t-1}|} \geq \frac{|O_{(k)}(u)|}{|I_{(k)}|})\right) \end{aligned}$$

In periodic mode $|I_{t-1}| = |I_{(k)}|$ as subscription scores are recomputed at the end of every period and thus the denominator is the same for all subscriptions and equal to the number of events in the period. Moreover, in eager mode $|I_{t-1}| = |I_{(k)}|$ as well since the denominator in both scores is the number of events in the window. Thus:

$$pnovel_{rate}(s,u) = 1 - P\left(|O_{t-1}(u)| \geq |O_{(k)}(u)|\right) \tag{6.1}$$

On the other hand, for interval-based ranking it holds that:

$$\begin{aligned} pnovel_{inter}(s,u) &= P\left(novel_{inter}(s,O_{t-1}(u)) > thr_{inter}\right) \\ &= 1 - P\left(novel_{inter}(s,O_{t-1}(u)) \leq thr_{inter}\right) \\ &= 1 - P\left(novel_{inter}(s,O_{t-1}(u)) \leq novel^{(k)}_{inter}(s',O_{t'-1}(u))\right) \\ &= 1 - P\left(t - L_{s,t} \leq t' - L_{s',t'}\right) \end{aligned}$$

Intuitively, the mean of $t - L_{s,t}$ is equal to the interdelivery time for subscription s and thus the inverse of the delivery frequency of s. That is: $E(t - L_{s,t}) = \frac{1}{delivery_rate(s)} = \frac{1}{\frac{|O_{t-1}(u)|}{|I_{t-1}|}}$, or $E(t - L_{s,t}) = \frac{|I_{t-1}|}{|O_{t-1}(u)|}$. That makes the above equation:

$$pnovel_{inter}(s,u) = 1 - P\left(\frac{|I_{t-1}|}{|O_{t-1}(u)|} \leq \frac{|I_{(k)}|}{|O_{(k)}(u)|}\right)$$

Last, as in rate-based ranking, $|I_{t-1}| = |I_{(k)}|$ for both period and sliding window modes. Substituting in the above equation we get:

$$pnovel_{inter}(s,u) = 1 - P\left(|O_{t-1}(u)| \geq |O_{(k)}(u)|\right) \tag{6.2}$$

Comparing equations 6.1 and 6.2 concludes the proof. ∎

Then, we give the proof of Theorem 2.3.

Proof. Let u be a user, $S(u) = \{s_1, s_2, \ldots, s_m\}$ be the set of the subscriptions of the user and $|O_u|$ be the number of events delivered to the user u. Also, let $|O_s(u)|$ denote the number of events delivered to the user u that match subscription $s \in S(u)$ and $p(s)$ denote the delivery rate of subscription s that is equal to $|O_s(u)|/|O_u|$. The entropy of the events delivered to the user u is equal to:

$$H(X,m) = -\sum_{i=1}^{m} p(s_i) \cdot log_2(p(s_i)) \tag{6.3}$$

Let $I(s_i) = p(s_i) \cdot log_2(p(s_i))$, with $1 \leq \mathrm{i} \leq m$. From Equation 6.3 we have:

$$H(X,m) = -\sum_{i=1}^{m} I(s_i) \tag{6.4}$$

An event that is delivered and matches subscription $s \in S$ contributes to the total entropy a factor -$I(s)$. Let e_t be a new event that arrives at the system and matches subscription s. In case e_t is delivered, it contributes to the total entropy a factor -$I(s+\delta)$, with $\delta = \frac{|O(u)|-|O_s(u)|}{|O(u)|\cdot(|O(u)|+1)} > 0$.

We define as ΔI a distance function that is equal to the difference between -$I(s)$ and -$I(s+\delta)$, that is:

$$\Delta I(s) = I(s) - I(s+\delta) = p(s) \cdot log_2(p(s)) - p(s+\delta) \cdot log_2(p(s+\delta)) \tag{6.5}$$

We aim at delivering the event that contributes at most to the total entropy, namely that has the highest value of distance function ΔI.

The first derivate of the distance function ΔI is:

$$\Delta I'(s) = log_2(p(s)) - log_2(p(s+\delta)) < 0 \tag{6.6}$$

The first derivative of the distance function is negative, so it is a monotonically decreasing function. The distance function takes its highest value, when an event is delivered that matches a subscription that has the lowest delivery rate, namely the most novel subscription. ∎

We give the proof of Equation 3.2.

Let $S(u)$ be the set of the subscriptions of user u, s be a subscription in $S(u)$ and e be an event. Also, let $|O_s(u)|$ be the number of delivered events that match s, $|M_s(u)|$ be the number of events that match s, $|M(u)|$ be the number of events that match at least

one of the subscriptions in $S(u)$ and $|O(u)|$ be the number of events that are delivered to the user. We define the following probabilities:

$$P(e \text{ delivered } | e \text{ matches } s, u) = \frac{|O_s(u)|}{|M_s(u)|}$$

$$P(e \text{ matches } s) = P_{match}(s) = \frac{|M_s(u)|}{|M(u)|}$$

The sampling probability $P_{sampl}(s)$ of subscription s is equal to $P(e \text{ delivered } | e \text{ matches } s)$.

$$\begin{aligned} P_{sampl}(s) &= P(e \text{ delivered } | e \text{ matches } s) \\ &= \frac{|O_s(u)|}{|M_s(u)|} \\ &= \frac{|O_s(u)|}{|M(u)|} \cdot \frac{|M(u)|}{|M_s(u)|} \end{aligned}$$

Let $r_{TH}(u)$ be the maximum delivery rate of user u. We want each subscription to achieve the same number of delivered events, that is $\frac{r_{TH}(u)}{|S(u)|}$. Consequently, we want $\frac{|O_s(u)|}{|M(u)|}$ to be equal to $\frac{r_{TH}(u)}{|S(u)|}$. Also, note that $\frac{|M(u)|}{|M_s(u)|}$ is equal to inverse matching probability of subscription s, that is $\frac{1}{P_{match}(s)}$. Consequently, the sampling probability $P_{sampl}(s)$ of subscription s is:

$$P_{sampl}(s) = \begin{cases} 1, \text{ if } P_{match}(s) < \frac{r_{TH}(u)}{|S(u)|} \\ \frac{r_{TH}(u)}{|S(u)|} \cdot \frac{|M(u)|}{|M_s(u)|}, \text{ otherwise} \end{cases}$$

AUTHOR'S PUBLICATIONS

D. Souravlias. *Data Stream Summarization to Avoid Overlap*, in Proc. of the 3rd Panhellenic Scientific Student Conference on Informatics, Related Technologies and Applications (EUREKA 2009), Sept 10-12, 2009, Corfu, Greece.

D. Souravlias, M. Drosou, K. Stefanidis and E. Pitoura. *On Novelty in Publish/Subscribe Delivery*, in Proc. of the 4th International Workshop on Ranking in Databases (DBRank - 2010), in conjunction with the ICDE 2010 Conference, Long Beach, California, USA.

Short Vita

Dimitris Souravlias was born in Ioannina in 1987. He was admitted at the Computer Science Department of the University of Ioannina in 2004 and graduated in February 2009. In March 2009 he began his postgraduate studies at the same department. Since 2007, he is a member of the Distributed Management of Data Laboratory (DMOD).