Turning Down the Noise in the Blogosphere

Khalid El-Arini[*] **Gaurav Veda**[*] **Dafna Shahaf**[*]
Carlos Guestrin[*]

May 2009

CMU-ML-09-103

School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213

Abstract

In recent years, the blogosphere has experienced a substantial increase in the number of posts published daily, forcing users to cope with information overload. The task of guiding users through this flood of information has thus become critical. To address this issue, we present a principled approach for picking a set of posts that best covers the important stories in the blogosphere. We define a simple and elegant notion of coverage and formalize it as a submodular optimization problem, for which we can efficiently compute a near-optimal solution. In addition, since people have varied interests, the ideal coverage algorithm should incorporate user preferences in order to tailor the selected posts to individual tastes. We define the problem of *learning a personalized coverage function* by providing an appropriate user-interaction model and formalizing an online learning framework for this task. We then provide a no-regret algorithm which can quickly learn a user's preferences from limited feedback. We evaluate our coverage and personalization algorithms extensively over real blog data. Results from a user study show that our simple coverage algorithm does as well as most popular blog aggregation sites, including Google Blog Search, Yahoo! Buzz, and Digg. Furthermore, we demonstrate empirically that our algorithm can successfully adapt to user preferences. We believe that our technique, especially with personalization, can dramatically reduce information overload.

[*] School of Computer Science, Carnegie Mellon University, Pittsburgh, PA, USA

Keywords: Blogs, Personalization

1 Introduction

"How many blogs does the world need?" asked TIME Magazine in 2008 (Kinsley, 2008), claiming that there are already too many. Indeed, the blogosphere has experienced a substantial increase in the number of posts published daily. One immediate consequence is that many readers now suffer from information overload.

While the vast majority of blogs are not worth reading for the average user, even the good ones are too many to keep up with. Moreover, there is often significant overlap in content among multiple blogs. To further complicate matters, many stories seem to resonate in the blogosphere to an extent that is largely uncorrelated with their true importance. For example, in the spring of 2007, Politico broke a story about John Edwards' $400 haircut in a blog post (Smith, 2007), which was almost instantly seized upon by the rest of the blogosphere. Over the next two weeks, the haircut story sparked several major online debates. Avoiding this story was difficult for most Web users, and nearly impossible for those interested in politics but not in this particular line of debate.

The goal of this paper is to *turn down the noise* in the blogosphere. We assume that users have very limited time for reading blog posts, and thus our goal is to show them a small set of posts covering the important stories currently being discussed. Furthermore, we allow users to personalize the process; after all, one man's noise may be another man's music.

In this paper, we formally define what it means for a set of posts to cover the blogosphere. One desired property of this notion of coverage is that it must be an efficiently computable function. For instance, due to the large size of our data sets, we cannot use most clustering algorithms, as they require quadratic computation. In addition, the coverage function must be expressive enough so that it can recognize the important stories in the blogosphere while at the same time identify the important features of a particular document. Finally, the notion should be soft, allowing partial (or probabilistic) coverage, as posts rarely offer complete coverage of their stories. We propose a simple and elegant notion that addresses these requirements and formalize a corresponding objective function, which exhibits a natural diminishing returns property known as submodularity. We present a near-optimal efficient algorithm for optimizing this function.

We then extend our notion of coverage to *personalized coverage*. Posts that cover the blogosphere for the average population may not be optimal for a particular user, given her personal preferences. For example, a user may like stories about badminton, irrespective of their prevalence. Learning a personalized coverage function allows us to show the users posts that are better suited to their tastes.

We formalize and address the problem of *learning a personalized coverage function*. First, we define an interaction model for user feedback that takes into account the order in which the posts are read. Using this model, we then define an online learning setting for coverage functions and provide a simple no-regret algorithm that guarantees we can quickly adapt to a user's preferences.

We evaluate our algorithm, Turning Down the Noise (TDN), on real blog data collected over a two week period in January 2009. We compare TDN to popular blog aggregation sites (Google Blog Search[1], Yahoo! Buzz[2], Digg[3], and BlogPulse[4]), measuring topicality and redundancy. Results from a user study show that our simple, fully-automated coverage algorithm performs as well as, or better than, most of these sites, including those based on user voting or human editing.

Perhaps most importantly, we demonstrate TDN's ability to successfully adapt to user preferences. Personalization not only improves user satisfaction, but is also able to simulate users with different interests. We believe that our algorithm, especially with personalization, can dramatically improve the information overload situation.

[1] http://blogsearch.google.com
[2] http://buzz.yahoo.com
[3] http://digg.com
[4] http://blogpulse.com

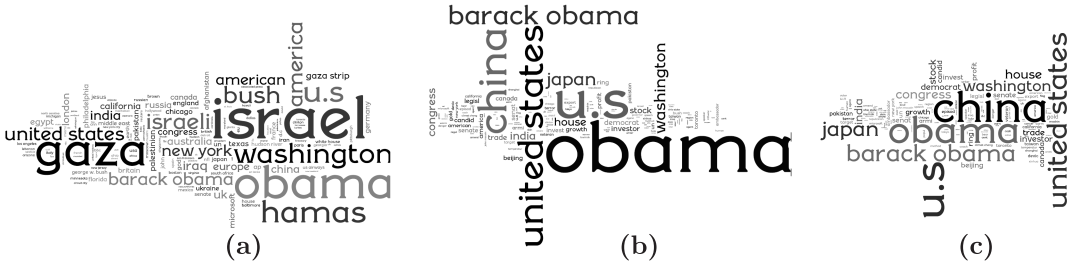


Figure 1: (a) Global word frequency across the blogosphere (January 17, 2009). The size of a word is proportional to its frequency. (b) Coverage vs. (c) incremental coverage of a post about Obama and China, given that we already saw a post about Obama. The incremental coverage of Obama is much smaller than the regular coverage.

In summary, our main contributions are:

- We define the notion of *covering the blogosphere* and formalize it as a submodular optimization problem, for which we provide a near-optimal solution.
- We define and formalize the problem of *learning a personalized coverage function*, and provide a no-regret algorithm for learning user preferences from limited feedback.
- We evaluate our algorithm on real blog data using both user studies and simulations, and compare it to popular blog aggregation sites.

2 Coverage

Figure 1(a) shows a typical day in the blogosphere (January 17, 2009). The size of a word is proportional to its frequency across the blogosphere. Examining the picture, we can spot some of the popular stories for that day: the inauguration of Barack Obama and the Israel-Gaza conflict.

Many posts cover the same story, e.g., the inauguration. Moreover, stories may have a certain degree of overlap. Intuitively, our goal is to select a small set of blog posts that captures the important stories of the day. At the same time, we wish to avoid redundancy. In the following section we formally state the problem of coverage and present an efficient optimization algorithm.

2.1 Documents and Features

We characterize the posts in the blogosphere by features. Features can be any arbitrary collection of objects, high- or low-level, for example: significant words (such as named entities and noun phrases), topics extracted from the corpus, or even higher-level semantic relations. As an example, refer again to Figure 1(a). Here, our features are common named entities. Each document will be about one or more of these features. More formally:

Definition 2.1 (Blogosphere)**.** *A blogosphere is a triplet* $\langle \mathcal{U}, Posts, cover.(\cdot)\rangle$. $\mathcal{U} = \{u_1, u_2, ...\}$ *is a finite set of features, and Posts is a finite set of posts. The relation between posts and features is captured by the* covering *function.* $cover_j(i) : \mathcal{U} \to \mathbb{R}^+$ *quantifies the amount* $post_j \in Posts$ *covers feature* u_i.

In the simplest case, $cover.(\cdot)$ is a binary indicator function, turning posts into subsets of features. Later, we explore other softer notions of coverage functions, e.g., ones with probabilistic interpretations.

2.2 Covering Features

Given our model $\langle \mathcal{U}, \mathit{Posts}, \mathit{cover}_{\cdot}(\cdot)\rangle$, we wish to determine how effectively a given small set of posts can cover the important stories in the blogosphere. More formally, our goal is to pick a set of k posts $\mathcal{A} \subseteq \mathit{Posts}$, in order to maximize some coverage objective. In this section we define desired properties of this objective function, and propose a solution that addresses these requirements.

Perhaps the most natural idea is to first *cluster* the posts, where posts in the same cluster cover the same features. Then, given clusters, we can pick a representative post from each of the k largest clusters. Such clustering approaches are common in the literature (Zhang et al., 2005). However, most clustering methods require us to compute the distance between every pair of posts, which amounts to $O(n^2)$ comparisons for n posts. Due to the sizable amount of posts published daily, methods that require $O(n^2)$ computation are practically infeasible. Our first desirable property for a coverage function is *scalability*, i.e., we should be able to evaluate coverage in time linear in the number of posts.

Another solution, which does not require quadratic complexity, would be to formulate coverage as maximizing the function,

$$F(\mathcal{A}) = \sum_{i \in \mathcal{U}} \mathit{cover}_{\mathcal{A}}(i), \tag{1}$$

where the $\mathit{cover}_{\mathcal{A}}(i)$ function measures the degree to which posts $\mathcal{A}$ cover feature u_i. If posts correspond to a collection of features, and $\mathit{cover}_{\cdot}(\cdot)$ are binary indicator functions, then Eq. 1 reduces to the *Budgeted Maximum Coverage* problem:

Definition 2.2 (Budgeted Maximum Coverage)**.**
Given a set of ground elements $\mathcal{U}$, a collection $\mathbb{S} = \{S_1, ...S_m\}$ of subsets of $\mathcal{U}$, and a budget $k \geq 0$, select $\mathcal{A} \subseteq \mathbb{S}$ of size at most k which maximizes the number of covered elements, $|\bigcup_{S_j \in \mathcal{A}} S_j|$.

In our setting, this coverage can be formalized as maximizing:

$$F(\mathcal{A}) = \sum_{i \in \mathcal{U}} \mathbf{1}(\exists a_j \in \mathcal{A} : \ \mathit{cover}_j(i) = 1).$$

Although max-coverage is an NP-hard problem, there are several efficient and effective approximation algorithms for this task. However, this naïve approach suffers from some serious drawbacks:

- *Feature significance in corpus*: All features in a corpus are treated equally, and thus we cannot emphasize the importance of certain features. For example, covering "Cathedral High School" should not be as valuable as covering "Obama."
- *Feature significance in post*: This objective function does not characterize how relevant a post is to a particular feature, e.g., a post about Obama's speech covers Obama just as much as a post that barely mentions him. As a side effect, this objective rewards "name-dropping" posts (posts that include many features, without being about any of them).
- *Incremental coverage*: This coverage notion is too strong, since after seeing one post that covers a certain feature, we will never gain anything from another post that covers the same feature. This does not correspond to our intuitive notion of coverage, which should be subject to the law of diminishing returns: each additional time we see a feature we get an additional reward, which decreases with the number of occurrences. For example, suppose we show the user a post about Obama's inauguration. The second post we consider showing her is about the effect of Obama's presidency on China. Figure 1(b) shows the raw coverage of the second post, and "Obama" is the top-covered feature. However, if we take into account the fact that we have already covered the feature "Obama" to some extent by the first post, the coverage by the second post changes. Figure 1(c) shows the *incremental coverage* by the second post. As illustrated, the significance of this post towards "Obama" is diminished, and most of our reward would come from covering "China."

We now address each of these three issues. To address *Feature significance in corpus*, we can simply assign (nonnegative) weights w_i to each feature u_i:

$$F(\mathcal{A}) = \sum_{i \in \mathcal{U}} w_i \ \mathbf{1}(\exists a_j \in \mathcal{A} : \ cover_j(i) = 1).$$

If features are words, the weights can correspond to their frequency in the data set.

We now turn our attention to *Feature significance in post*. Each post should exhibit different degrees of coverage for the features it contains, which can be achieved by softening the notion of coverage, $cover_j(i)$. One approach is to use a generative model to estimate the probability of a feature given a post, $P(u_i \mid post_j)$. If, for example, our features are topics discovered by a topic model, then this term is simply the probability that document j is about topic i. More generally, any generative model for the particular set of features can be used to define this probability.

Given such a probabilistic model, we can define the notion of soft coverage more formally. If our features are sufficiently high-level, e.g., topics in a topic model, then a post can be thought of as being about a single feature, in which case $cover_j(i) = P(u_i \mid post_j)$. Alternatively, for lower-level features, such as named entities, we could assume that each post is about ℓ features. If these features are picked, for example, at random with replacement from $P(u_i \mid post_j)$, then our coverage will become $cover_j(i) = 1 - (1 - P(u_i \mid post_j))^{\ell}$. By requiring that all posts cover the same number of features, we alleviate the problem of "name-dropping," since a post cannot cover a large number of features well.

The probabilistic approach allows us to define feature importance in individual posts as well as in the whole corpus. However, if we define coverage as $F(\mathcal{A}) = \sum_{i \in \mathcal{U}} w_i \ \sum_{a_j \in \mathcal{A}} cover_j(i)$, then the *Incremental coverage* problem would persist, as this function does not possess the diminishing returns property. Instead, extending the probabilistic interpretation further, we can view set-coverage as a sampling procedure: each post tries to cover feature i with probability $cover_j(i)$, and the feature is covered if at least one of the posts in $\mathcal{A}$ succeeded. Thus, as $\mathcal{A}$ grows, adding a post provides less and less additional coverage. Formally, we can define the probabilistic coverage of a feature by a set of posts $\mathcal{A}$ as:

$$cover_{\mathcal{A}}(i) = 1 - \prod_{a_j \in \mathcal{A}} (1 - cover_j(i)). \tag{2}$$

Finally, we propose the following objective function for the problem of probabilistic coverage of the blogosphere:

$$F(\mathcal{A}) = \sum_{i \in \mathcal{U}} w_i \, cover_{\mathcal{A}}(i). \tag{3}$$

Our task is to find k posts maximizing the above objective function:

$$\mathcal{A}^* = \operatorname*{argmax}_{\mathcal{A} \subseteq Posts : |\mathcal{A}| \leq k} F(\mathcal{A}). \tag{4}$$

2.3 Optimizing Coverage of the Blogosphere

Using the notion of coverage in Eq. 2, our goal now is to find the set of posts $\mathcal{A}$ that maximizes our objective function in Eq. 3. Unfortunately, we can show by reduction from max-coverage that this objective is NP-complete, suggesting that the exact maximization of this function is intractable. However, our objective function satisfies an intuitive diminishing returns property, *submodularity*, which allows us to find good approximations very efficiently:

Definition 2.3 (Submodularity)**.** *A set function F is* submodular *if,* $\forall \mathcal{A} \subseteq \mathcal{B} \subseteq \mathcal{V}, \forall s \in \mathcal{V} \setminus \mathcal{B}$, $F(\mathcal{A} \cup \{s\}) - F(\mathcal{A}) \geq F(\mathcal{B} \cup \{s\}) - F(\mathcal{B})$.

Claim 2.4. *The probabilistic coverage function for the blogosphere in Eq. 3 is submodular.*

Proof. Let $\mathcal{B} \subseteq \mathcal{V}$ and $s \in \mathcal{V} \setminus \mathcal{B}$.

$$\begin{aligned} cover_{\mathcal{B}\cup\{s\}}(i) - cover_{\mathcal{B}}(i) &= 1 - \prod_{j\in\mathcal{B}\cup\{s\}} (1 - cover_j(i)) - \left(1 - \prod_{j\in\mathcal{B}} (1 - cover_j(i))\right), \\ &= \prod_{j\in\mathcal{B}} (1 - cover_j(i)) - \prod_{j\in\mathcal{B}\cup\{s\}} (1 - cover_j(i)), \\ &= \prod_{j\in\mathcal{B}} (1 - cover_j(i)) \left(1 - (1 - cover_s(i))\right), \\ &= \prod_{j\in\mathcal{B}} (1 - cover_j(i)) \left(cover_s(i)\right). \end{aligned}$$

Because $cover_j(i)$ is defined as a probability, it is in the range $[0, 1]$, and therefore $(1 - cover_j(i)) \in [0, 1]$ (for all j). Thus, for any $\mathcal{A} \subseteq \mathcal{B}$, we have that, $\prod_{j\in\mathcal{B}}(1 - cover_j(i)) \leq \prod_{j\in\mathcal{A}}(1 - cover_j(i))$. Hence,

$$\begin{aligned} \prod_{j\in\mathcal{B}} (1 - cover_j(i)) \left(cover_s(i)\right) &\leq \prod_{j\in\mathcal{A}} (1 - cover_j(i)) \left(cover_s(i)\right), \\ &= \prod_{j\in\mathcal{A}} (1 - cover_j(i)) - \prod_{j\in\mathcal{A}\cup\{s\}} (1 - cover_j(i)), \\ &= cover_{\mathcal{A}\cup\{s\}}(i) - cover_{\mathcal{A}}(i). \end{aligned}$$

Thus, $cover_{\mathcal{A}}(i)$ is submodular. Since submodularity is closed under nonnegative linear combinations, and our weights $w_i \geq 0$, it directly follows that our coverage function $F(\mathcal{A})$ is submodular. $\square$

Intuitively, submodularity characterizes the notion that reading a post s after reading a small set of posts $\mathcal{A}$ provides more coverage than reading s after having already read the larger set $\mathcal{B} \supseteq \mathcal{A}$.

Although maximizing submodular functions is NP-hard (Khuller et al., 1999), by discovering this property in our problem, we can take advantage of several efficient approximation algorithms with theoretical guarantees. For example, the classic result of Nemhauser et al. (Nemhauser et al., 1978) shows that by simply applying a greedy algorithm to maximize our objective function in Eq. 3, we can obtain a $(1 - \frac{1}{e})$ approximation of the optimal value. Thus, a simple greedy optimization can provide us with a near-optimal solution. However, since our set of posts is very large, a naïve greedy approach can be too costly. Therefore, we use CELF (Leskovec et al., 2007), which provides the same approximation guarantees, but uses lazy evaluations, often leading to dramatic speedups.

3 Personalization

Thus far, we have defined a global notion of coverage for the blogosphere. However, each user has different interests, and the selected posts that cover the prevalent stories may contain many topics that do not interest him. Instead, our goal in this section is to utilize user feedback in order to *learn* a personalized notion of coverage for each user.

Recall that, in the previous section, $F(\mathcal{A})$ assigns a fixed weight w_i to every feature, representing its importance. In practice, feature importance varies among different users. One user might care about a feature "NASCAR," while others may be indifferent to it. To address this issue, we augment the fixed weights w_i with personalized preferences π_i for each feature i. In the following, we assume that a user's coverage function is of the form:

$$F_{\pi^*}(\mathcal{A}) = \sum_{i\in\mathcal{U}} \pi_i^* \; w_i \; cover_{\mathcal{A}}(i), \tag{5}$$

for some unknown set of weights $\{\pi_i^*\}$. Our goal now is to learn a user's coverage function $F_{\pi^*}(\mathcal{A})$ by learning this optimal set of preferences $\{\pi_i^*\}$.

3.1 Interaction Models

In order to receive personalized results, users need to communicate their preferences. Since F_π is a set function, the most natural notion of feedback from a machine learning perspective would be for users to provide a single label for the set of posts that they are presented, indicating whether they like or dislike the entire set. However, this approach suffers from two limitations. First, from the point of view of the user, it is not very natural to provide feedback on an entire set of posts. Second, since there are exponentially many such sets, we are likely to need an extensive amount of user feedback (in terms of sets of posts) before we could learn this function. Instead, we assume that users go through a list of posts $\mathcal{A}$ in order, submitting feedback f_j ("liked"$= +1$, "indifferent" $= 0$, "disliked" $=$ -1) for each post $a_j \in \mathcal{A}$. We take no feedback on a post to mean "indifferent."

3.2 Personalization by Minimizing Regret

Our objective function is defined in terms of sets, but our feedback is in terms of individual posts. How should we provide an appropriate credit assignment?

One possible solution would be to assume that the feedback that a user provides for a particular post is independent of the other posts presented in the same set. In this case, one can view the user feedback as being labeled data on which we can train a classifier to determine which posts the user likes. However, this assumption does not fit with our interaction model, as a user might not like a post either because of its content or because previous posts have already covered the story.

To address this issue, we consider the *incremental coverage* of a post, i.e., the advantage it provides over the previous posts. The incremental coverage we receive by adding post a_j to the set $\mathcal{A}$ is:

$$\textit{inc-cover}_j(\mathcal{A}, i) = \textit{cover}_{\mathcal{A} \cup a_j}(i) - \textit{cover}_{\mathcal{A}}(i).$$

Note that if $\textit{cover}_{\mathcal{A}}(i)$ is defined as in Eq. 2, then the incremental coverage is the probability that a_j is the first post to cover feature u_i. Furthermore, if we view the set of documents $\mathcal{A}$ as an ordered set $\mathcal{A} = \{a_1, \ldots, a_k\}$[5], the sum of incremental coverages is a telescoping sum that yields the coverage of a set of documents $\mathcal{A}$:

$$\begin{aligned} \sum_{a_j \in \mathcal{A}} \textit{inc-cover}_j(a_{1:j-1}, i) &= \sum_{a_j \in \mathcal{A}} \textit{cover}_{a_{1:j}}(i) - \textit{cover}_{a_{1:j-1}}(i), \\ &= \textit{cover}_{\mathcal{A}}(i), \end{aligned}$$

where $a_{1:j-1}$ is shorthand for the set of documents $\{a_1, \ldots, a_{j-1}\}$.

Using incremental coverages, we can now define the reward we receive after presenting $\mathcal{A}$ to a user with preferences π and obtaining feedback f:

$$\textit{Rew}(\pi, \mathcal{A}, f) = \sum_{i \in \mathcal{U}} \pi_i \; w_i \sum_{a_j \in \mathcal{A}} f_j \quad \textit{inc-cover}_j(a_{1:j-1}, i).$$

If the user liked all of the documents in $\mathcal{A}$ (i.e., $\forall j, f_j = 1$), this reward becomes exactly the coverage function we are seeking to maximize, $F_\pi(\mathcal{A}) = \sum_{i \in \mathcal{U}} \pi_i \; w_i \; \textit{cover}_{\mathcal{A}}(i)$, as in Eq. 5.

Our algorithm maintains an estimate of the user's preferences at each time step t, $\pi^{(t)}$. Given this estimate, we optimize $F_{\pi^{(t)}}(\mathcal{A})$ and pick a set of documents $\mathcal{A}^{(t)}$ to show the user. After receiving

[5]This ordering could be defined by the order the posts are presented to the user, e.g., the one picked by the greedy algorithm.

feedback $f^{(t)}$, we gain a reward of $Rew(\pi^{(t)}, \mathcal{A}^{(t)}, f^{(t)})$. After T time steps, our average reward is therefore:

$$AvgRew(T) = \frac{1}{T}\sum_{t=1}^{T} Rew(\pi^{(t)}, \mathcal{A}^{(t)}, f^{(t)}).$$

Since our decisions at time t can only take into account the feedback we have received up to time $t-1$, the decisions we made may have been suboptimal. For comparison, consider the reward we would have received if we had made an informed choice for the user's preferences π considering all of the feedback from the T time steps:

$$BestAvgRew(T) = \max_{\pi} \frac{1}{T}\sum_{t=1}^{T} Rew(\pi, \mathcal{A}^{(t)}, f^{(t)}). \tag{6}$$

That is, after seeing all the user feedback, what would have been the right choice for user preference weights π? The difference between our reward and this best choice in retrospect is called the *regret*:

Definition 3.1 (Regret)**.** *Our average regret after T time steps is the difference $BestAvgRew(T) - AvgRew(T)$.*

Positive regret means that we would have preferred to use the weights π that maximize Eq. 6 instead of our actual choice of weights $\pi^{(t)}$. A *no-regret learning algorithm*, such as the one we describe in the next section, will allow us to learn $\pi^{(t)}$ such that, as T goes to infinity, the regret will go to zero at a rapid rate. Intuitively, this no-regret guarantee means that we learn a sequence $\pi^{(t)}$ that does as well as any fixed π–including the true user preferences, π^*–on the sets of posts that the user is presented. By learning the personalized coverage function for a particular user in this manner, the posts we provide will be tailored to his tastes.

A stronger guarantee would be to show that the weights $\pi^{(t)}$ not only do well on the sets of posts from which they were learned, but also on the posts that would have been selected had we used the true π^* as the user preference weights for each day. For example, consider a user who is interested in politics and sports, but is also passionate about bagpiping. We may never show him any bagpiping posts, since they are not likely to be common. Thus, we may never receive feedback that would allow us to accurately model this portion of the user's true preferences. We intend to address this issue in future work.

3.3 Learning a User's Preferences

We now describe our algorithm for learning π^* from repeated user feedback sessions. Like many online algorithms (Cesa-Bianchi & Lugosi, 2006), our approach updates our estimated $\pi^{(t)}$ using a multiplicative update rule. In particular, our approach can be viewed as a special case of Freund and Schapire's multiplicative weights algorithm (Freund & Schapire, 1999).

The algorithm starts by choosing an initial set of weights $\pi^{(1)}$. (WLOG, we assume weights are normalized to sum to 1, since the coverage function is insensitive to scaling.) In the absence of prior knowledge about the user, we can choose the uniform distribution:

$$\pi_i^{(1)} = \frac{1}{|\mathcal{U}|}.$$

If we have prior knowledge about the user, we can start from the corresponding set of weights.

At every round t, we use our current distribution $\pi^{(t)}$ to pick k posts, $\mathcal{A}^{(t)}$, to show the user. After receiving feedback $f^{(t)}$, we would like to increase the weight of features covered by posts the user liked, and decrease the weight of features covered by posts the user disliked. These updates can be achieved by a simple multiplicative update rule:

$$\pi_i^{(t+1)} = \frac{1}{Z}\, \pi_i^{(t)}\, \beta^{-\mathcal{M}(i, f^{(t)})}, \tag{7}$$

where Z is the normalization constant, $\beta \in (0, 1)$ is the learning rate, and, intuitively, $\mathcal{M}(i, f^{(t)})$ measures the contribution (positive or negative) that feature i had on our reward:

$$\mathcal{M}(i, f^{(t)}) := \frac{w_i \ \sum_{a_j \in \mathcal{A}^{(t)}} f_j^{(t)} \ \ \textit{inc-cover}_j(a_{1:j-1}, i)}{2 \max_i w_i}, \tag{8}$$

where the normalization by $2 \max_i w_i$ is simply used to keep this term in the range $[-0.5, 0.5]$.

If the learning rate β is small, we make large moves based on the user feedback. As the learning rate tends to 1, these updates become less significant. Thus, intuitively, we will start with a small value of β and slowly increase it.

Claim 3.2. *If, for number of personalization epochs T, we use a learning rate β_T given by:*

$$\beta_T := \frac{1}{1 + \sqrt{\frac{2 \ln |\mathcal{U}|}{T}}}, \tag{9}$$

then our preference learning procedure will have regret bounded by:

$$\textit{BestAvgRew}(T) - \textit{AvgRew}(T) \leq \mathcal{O}\left(\sqrt{\frac{\ln |\mathcal{U}|}{T}}\right).$$

Since our regret goes to zero as T goes to infinity, our approach is called a no-regret algorithm. The proof follows from Freund and Schapire (Freund & Schapire, 1999), by formalizing our learning process as a two-player repeated matrix game involving our algorithm and the user. (More details can be found in Appendix A.)

4 Evaluation

We evaluate our algorithm on real blog data collected over a two week period in January 2009. These posts come from a diverse set of blogs, including personal blogs, blogs from mainstream news sites, commercial blogs, and many others.

We obtain the data from Spinn3r[6], which indexes and crawls 12 million blogs, collecting approximately 500,000 posts per day. After performing some simple data cleaning steps, such as removing web forums and classifieds, we reduce this number to about 200,000 posts per day in our data set. However, as this is real Web data, it is still invariably noisy even after cleaning. Thus, our algorithm must be robust to content extraction problems.

For each post, we extract named entities and noun phrases using the Stanford Named Entity Recognizer (Finkel et al., 2005) and the LBJ Part of Speech Tagger (Rizzolo & Roth, 2007), respectively. We remove infrequent named entities and uninformative noun phrases (e.g., common nouns such as "year"), leaving us with a total collection size of nearly 3,000. (More details can be found in Appendix B.)

We evaluate an instantiation of our algorithm with high level topic model-based features, which we refer to as TDN+LDA. We define our set of features as topics from a latent Dirichlet allocation (LDA) (Blei et al., 2003) topic model learned on the noun phrases and named entities described above. We take the weight of each feature to be the fraction of words in the corpus assigned to that topic. As described in Section 2.2, we can directly define $\textit{cover}_j(i) = P(u_i \mid \textit{post}_j)$, which in the setting of topic models is the probability that $\textit{post}_j$ is about topic i. We use a Gibbs sampling implementation of LDA (Griffiths & Steyvers, 2004) with 100 topics and the default parameter settings.

[6]http://www.spinn3r.com

Once we have extracted the named entities and noun phrases, LDA is the slowest part of running TDN+LDA. After a 300 iteration burn-in period, we run 2,500 iterations of Gibbs sampling and select 500 samples from them. On a single 3GHz processor, this process takes less than 2GB of RAM and between 1-2 hours to run for an eight hour corpus of blog posts. The submodular function optimization needed to generate posts takes under a minute.

We also evaluate a variant of our algorithm with features consisting of the named entities and noun phrases directly, which we refer to as TDN+NE. As this variant uses a lower-level feature set, it assumes a post can cover multiple features, and thus uses the coverage function for covering ℓ features described in Section 2.2. The value of ℓ is set to be the average number of occurrences of named entities and nouns per document in our corpus, which is approximately 16. In this setting, post selection takes about five minutes.

4.1 Evaluating Coverage

As detailed in Section 2, the main objective of our algorithm is to select a set of posts that best covers the important and prevalent stories currently being discussed in the blogosphere. The major world events that took place during the time corresponding to our data set included the Israel-Gaza conflict, the inauguration of Barack Obama, the gas dispute between Russia and Ukraine, as well as the global financial crisis. As an example, here is the set of posts that our algorithm selects for an eight hour period on January 18, if our budget k is set to five:

1. Israel unilaterally halts fire as rockets persist
2. Downed jet lifted from ice-laden Hudson River
3. Israeli-trained Gaza doctor loses three daughters and niece to IDF tank shell
4. EU wary as Russia and Ukraine reach gas deal
5. Obama's first day as president: prayers, war council, economists, White House reception

The selected five posts all cover important stories from this particular day. The Israel-Gaza conflict appears twice in this set, due to its extensive presence in the blogosphere at the time. It is important to note, however, that these two posts present different aspects of the conflict, each being a prevalent story in its own right. By expanding the budget to fifteen posts, the algorithm makes additional selections related to other major stories of the day (e.g., George W. Bush's legacy), but also selects "lifestyle" posts on religion and cooking, since these represent the large portion of the blogosphere that is not directly related to news and current events.

As another example, here are the top five selected posts from the morning of January 23, the day after the Academy Award nominations were announced:

1. Button is top Oscar nominee
2. Israel rules out opening Gaza border if Hamas gains
3. Paterson chooses Gillibrand for U.S. Senate
4. Fearless Kitchen: Recipe: Medieval Lamb Wrap
5. How Obama avoided a misguided policy blunder

A post describing the Oscar-nominated movie *The Curious Case of Benjamin Button* supplants the Israel-Gaza conflict at the top of the list, while a cooking post makes it up to the fourth position.

We wish to quantitatively evaluate how well a particular post selection technique achieves the notion of coverage we describe above on *real blog data*. However, the standard information retrieval metrics of precision and recall are not directly applicable in our case, since we do not have labels identifying all the prevalent stories in the blogosphere on a given day and assigning them to specific posts. Rather, we measure the *topicality* of individual posts as well as the *redundancy* of a set of posts. We say a post is *topical* with respect to a given time period if its content is related to a major news event from that period. A post r is *redundant* with respect to a previous post p if it

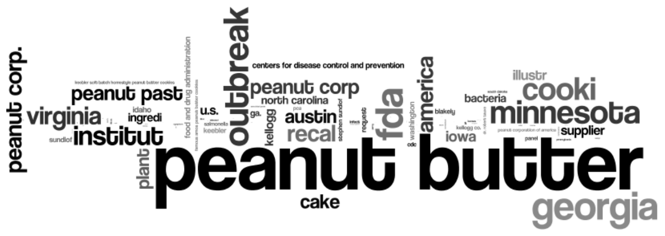


Figure 2: Topic representing the peanut butter recall from January 18, 2009, with the size of a word proportional to its importance in the topic.

contains little or no additional information to post p. An ideal set of posts that covers the major stories discussed in the blogosphere would have high topicality and low redundancy.

We conducted a study on 27 users to obtain labels for topicality and redundancy on our data. We compared TDN+LDA and TDN+NE to four popular blog aggregation sites: the front page of Digg, Google Blog Search, Nielsen BuzzMetrics' BlogPulse, and Yahoo! Buzz. We intended on evaluating Technorati[7] as well, but their RSS feed was unavailable for most days in our evaluation period. Additionally, we also examine the performance of simpler objective functions on the post selection task.

4.1.1 Measuring Topicality

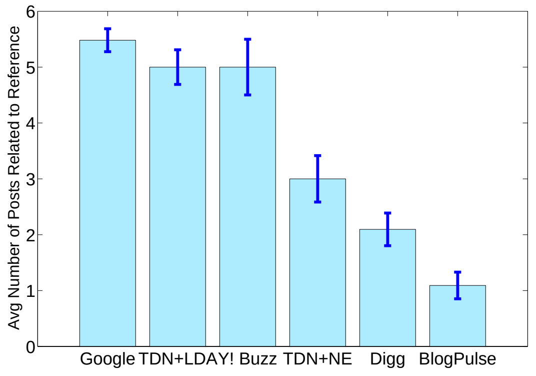
In order for users to measure the topicality of a blog post, they need an idea of what the major news stories are from the same time period. We express this information to our study participants by providing them with headlines gathered from major news sources in five different categories: world news, politics, business, sports, and entertainment. The headlines for each category are aggregated from three different news sources to provide a wider selection for the users and to avoid naming a single source as the definitive news outlet for a category. For instance, for politics we present headlines from Reuters, *USA Today*, and *The Washington Post*. This collection of headlines is akin to a condensed newspaper, and we refer to these stories as *reference stories*.

We present the participants with reference stories gathered at a particular time, e.g., January 18, 2009, 2:00pm EST, which we call the *reference time*. We then show each participant a set of ten posts that was chosen by one of the six post selection techniques, and ask them to mark whether each post is "related" to the reference stories. Each post is presented as a title along with a short description. The users are not made aware of which technique the posts come from, so as not to bias their ratings. The posts selected by TDN+LDA and TDN+NE were chosen from an eight hour window of data ending at the reference time, while the posts selected by the popular blog aggregation sites were retrieved from these sites within fifteen minutes of the reference time.

Figure 3(left) shows the results of the topicality user ratings on the six techniques. On average, the sets of ten posts selected by Google Blog Search, TDN+LDA and Yahoo! Buzz each contain five topical posts out of ten presented. The topicality of these techniques is significantly better than that of TDN+NE, Digg and BlogPulse. BlogPulse selects the most linked-to posts of the day, which does not seem to be a good heuristic for covering the important stories. Many of these posts are technology how-to pages, such as "Help With Social Bookmarking Sites," the highest ranked post from January 18. Digg selects its top posts by user voting, and thus the top selected posts consist of a few prevalent stories and many entertaining or shocking posts, such as "Teen Stabbed But Makes It To Job Interview," the top post from February 6.

TDN+LDA outperforms TDN+NE because high-level features, such as LDA topics, capture stories in a better way than low-level features do. For example, for one eight hour period in our data set,

[7] http://technorati.com



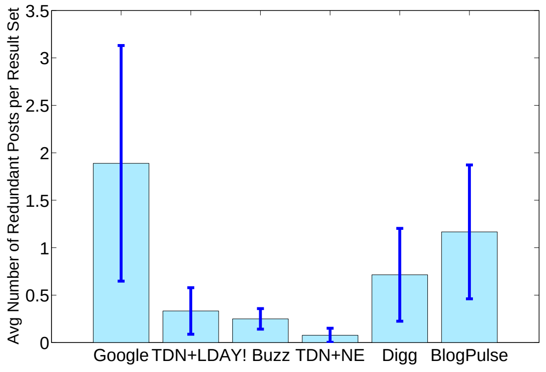


Figure 3: Left: Results from user study measuring topicality. The bars show the average number of posts (out of 10) that users found to be topical with respect to the reference stories. Right: Results of the redundancy user study. Users report the number of redundant posts for each post selection technique they are presented with. Error bars on all plots indicate standard error.

there is a coherent LDA topic about the EU-Russia gas crisis. Therefore, when we cover this topic, we will present a story that is about the crisis. However, the named entity "Russia" may be covered by multiple stories. TDN+NE selects a post about Russia's plan to go ahead with the opening of a pediatric medical center in Moscow despite the current financial crisis, since it contains important named entities and nouns like "Russia," "Putin," "crisis," etc. Hence, if we only cover low-level features, we might select a post that is not topical, yet contains multiple important features.

While topicality captures a major aspect of our notion of coverage, in that important current events are covered by the selected posts, one drawback of this evaluation method is that lifestyle blog posts are not adequately represented. It is difficult to define a set of reference sites that summarize the day's most important recipes or most prevalent do-it-yourself tips, for instance. Furthermore, in our case, we did not want to show our study participants more than five categories of reference stories, so as not to overwhelm them. As a result, a post related to an important technology story would likely not be considered topical, as we left this category out.

4.1.2 Measuring Redundancy

The user study described in the previous section allowed us to measure whether posts were topical or not. However, topicality is not enough to judge the goodness of a set of posts, since they may all be about the same story, and hence not interesting. Instead, we want the posts to be diverse, so that they capture all of the important stories in the blogosphere, as well as appeal to everyone's interests. As part of our user study, we asked users to look at a set of fifteen posts selected by one of the six previously described post selection techniques, and mark any occurrences they thought were redundant. Each of 27 participants was presented with either two or three sets of posts generated by different algorithms over the same time period. The users were not aware of the sources of the posts.

Figure 3(right) shows that both variants of our algorithm outperform Digg, BlogPulse and Google Blog Search on the redundancy metric. In other words, our algorithm selects diverse sets of posts. This diversity is primarily due to the diminishing returns property of our objective function. If we have covered the important features of a story once, covering it again yields only a small reward. Google Blog Search has the highest number of redundant results, and has high variance, suggesting that on some days many of the posts on its front page are similar. In fact, on average, the posts selected by Google Blog Search are nearly six times as redundant as those selected by TDN+LDA.

However, it should be noted that performing well on the redundancy metric alone is not sufficient. For example, it may turn out that all the posts picked by an algorithm are non-redundant, but meaningless, and hence of no interest to a user. Thus, an algorithm needs to perform well on both the topicality and the redundancy metric in order for it to be useful.

TDN+LDA and Yahoo! Buzz were the two techniques that performed well in both metrics. However, while Yahoo! Buzz uses Web search trends, user voting and other features to select its posts, TDN+LDA achieves the same topicality and redundancy performance by selecting posts only using simple text features. Furthermore, TDN+LDA adapts its results to user preferences, as described in Section 4.2.

4.1.3 Alternative Objective Functions

As an alternative to the submodular objective function defined in Eq. 3, we consider two simpler objective functions.

LDA-based Modular Function. A *modular function* is an additive set function where each element is associated with a fixed score, and the value for a set $\mathcal{A}$ is the sum of the scores of the elements of $\mathcal{A}$. Since the score of a post does not depend on the other elements in the set, there is no incentive to select a diverse set of posts. The naïve way of selecting posts using LDA fits under this modular framework. We first pick the top k topics based on their weight in the corpus. For each one, we pick the post that covers it the most. In addition to the potential for redundancy mentioned above, this technique suffers from the fact that it commits to a topic irrespective of the quality of the posts covering it. Furthermore, even if a post covers multiple topics well, it might not be selected as there may be some posts that better cover each individual topic. Using a strictly submodular objective function alleviates these problems.

For example, if we define our features based on a 50-topic LDA model trained on an eight hour data set from January 18, the topic with the lowest weight is about the peanut butter recall, a major news story at this time (*cf.* Figure 2). Thus, if we select fifteen posts following the naïve LDA approach, we do not pick a post from this topic. However, the weight of this topic (0.019) is not much lower than the mean topic weight (0.020). Moreover, since this topic closely corresponds to a prevalent news story, many posts cover it with high probability. TDN selects such a post because, unlike the naïve LDA approach, it simultaneously considers both the topic weights and the post coverage probabilities.

Budgeted Maximum Coverage. Another simple objective function we consider is budgeted maximum coverage, introduced in Definition 2.2, but with each feature (in this case, noun phrases and named entities) weighted by its corpus frequency. Optimizing this objective leads to the aforementioned "name-dropping" posts. For example, on an eight hour data set from January 20, the second post selected announces the schedule of a rock band's upcoming world tour, and thus completely covers the features, "Washington," "Boston," "New York," "London," "Rome," and a few dozen more cities and countries. Once this post has been selected, there is no further incentive to cover these features.

4.2 Personalization

There are two methods by which we evaluate how well our algorithm personalizes the posts it selects in response to user feedback. In one setting, we conduct a user study to directly measure how many of the presented posts a study participant would like to read. In the second setting, we simulate user preferences on a targeted set of blog posts and observe how our objective function $F(\mathcal{A})$ changes with respect to the unpersonalized case.

4.2.1 Preferences of Real Users

We divide our blog data into 33 eight hour segments (epochs), and pick a starting segment at random for a particular user. We present our user with a set of ten posts from his starting segment, selected using TDN+LDA. The posts are displayed as a title and short summary. The user is instructed to read down the list of posts and, one by one, mark each post as "would like to read," "would not like to read," or "indifferent." The user is told to make each decision with respect to the previous posts displayed in that set, so as to capture the notion of incremental coverage. For example, a user might be excited to read a post about Obama's inauguration appearing at the top slot in a particular result set, and thus would mark it as "like to read." However, if four other very similar posts appear below it, by the time he gets to rating the fifth inauguration post in a row, he will likely label it as "not like to read."

After each set of ten posts, our personalization algorithm uses the user ratings to update the weights $\pi^{(t)}$, and selects a personalized set of posts for the next epoch[8]. We also ask the user to mark his preferences on unpersonalized posts presented for the same epochs. The order in which these two conditions are presented is randomized. We repeat this process for a total of five epochs. As this is not a longitudinal study, and we do not wish it to be overly tedious for our participants, we accelerate the personalization process by using a learning rate β of 0.5, corresponding to a short-term learning horizon (i.e., $T \approx 9$ from Eq. 9).

Figure 4(a) shows the result of this study on twenty users. The vertical axis of the plot shows the average number of posts liked by a user in a single epoch. As one would expect, at epoch 0, when the posts are always unpersonalized, the number of liked posts is approximately the same between the personalized and unpersonalized runs. However, in just two epochs, the users already show a preference towards the personalized results.

If a user only prefers sports posts, personalization is easy, as the user's interests are narrow. In our study, however, the participants were simply instructed to rate posts with their own personal preferences. As people are often eclectic and have varied interests, this task is harder, but more realistic. Thus, it is notable that we are still able to successfully adjust to user tastes in very few epochs, showing a significant improvement over the unpersonalized case.

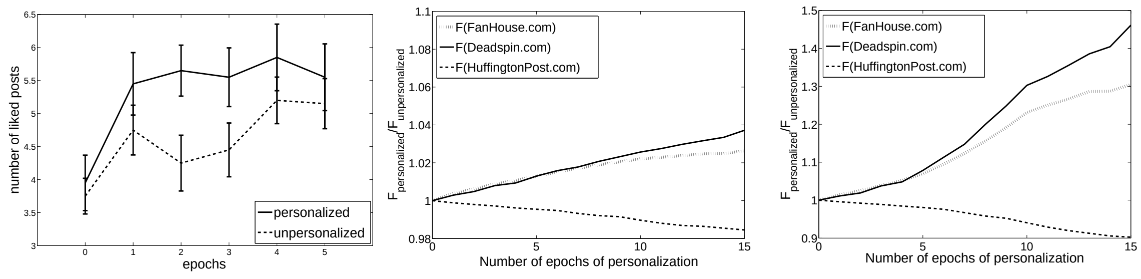
If instead of asking users to rate posts according to their personal tastes, we ask them to pretend that they only want to read posts on a specific subject (e.g., India), we observe interesting qualitative behavior. Initially, the top posts selected are about the main stories of the day, including the Israel-Gaza conflict and the Obama inauguration. After a few epochs of marking any India-related posts as "like" and all others as "dislike," the makeup of the selected posts changes to include more posts about the Indian subcontinent (e.g., "Pakistan flaunts its all-weather ties with China"). This is particularly notable given that these posts appear relatively infrequently in our data set, and thus without personalization, are rarely selected. Also, while after enough epochs, stories about India eventually supplant the other major news stories at the top of the result set, the Israel-Gaza stories do not disappear from the list, due to their high prevalence. We believe this is precisely the behavior one would want from such a personalization setting.

4.2.2 Simulating Preferences

We consider the case of a hypothetical sports fan, who always loves to read any sports-related post. In particular, every day, he is presented with a set of posts from the popular sports blog FanHouse.com, and he marks that he likes all of them. We simulate such a user in order to empirically examine the effect of personalization on the objective function.

Specifically, we simulate this sports fan by marking all FanHouse.com posts as "liked" over a specified number of personalization epochs, updating the personalization weights $\pi^{(t)}$at each epoch.

[8]As topics tend to change from one epoch to the next, we employ a simple bipartite matching algorithm to map personalization weights across epochs. Alternatively, one could use more recent topic models that are designed to work on streaming data (Canini et al., 2009).



(a) **(b)** **(c)**

Figure 4: (a) Results of the personalization user study, measuring how many posts each user liked, out of the ten presented by TDN+LDA in each epoch. The personalized line corresponds to a learning rate $\beta = 0.5$. (b,c) Effect of number epochs spent personalizing to the simulated preferences of a sports fan on the objective function F, with respect to no personalization. F is evaluated on two sports blogs and one politics blog. Learning rate $\beta = 0.5$ (b), 0.1 (c)

On the next epoch, which we call the evaluation epoch, we compute our objective function $F(\mathcal{A})$ on three different sets of posts. First, we compute $F(\mathcal{A})$ on the FanHouse.com posts from this epoch, hypothesizing that the more epochs we spend personalizing prior to the evaluation epoch, the higher this value will be. Second, we compute $F(\mathcal{A})$ on all the posts from DeadSpin.com, another popular sports blog. We also expect to see a higher value of our objective in this case. Finally, we compute $F(\mathcal{A})$ on all the posts from the HuffingtonPost.com Blog, a popular politics blog. The expectation is that by personalizing on sports posts for several days, $F(\mathcal{A})$ for a set $\mathcal{A}$ of politics posts will decrease with respect to the unpersonalized case.

Figure 4(b) shows the results of this experiment with a β value of 0.5, and we observe precisely the hypothesized behavior. The vertical axis of this plot shows the ratio of $F(\mathcal{A})$ computed with the learned personalization weights to that of $F(\mathcal{A})$ with the unpersonalized uniform weights, allowing us to compare across the three blogs. Thus, points on the plot that appear higher along the vertical axis than 1 indicate an improvement over the unpersonalized case, while any value below 1 indicates a decline with respect to the unpersonalized case.

Figure 4(c) shows the same simulation but with $\beta = 0.1$. This is an aggressive setting of the learning rate, and thus, as expected, the plot shows the objective function changing in the same direction but more rapidly when compared to Figure 4(b). These figures capture an important trade off for a deployed system, in that by varying the learning rate β, we trade off the speed of personalization with the variety of selected posts.

5 Related Work

Recently, there has been an increase in the number of websites that index blogs and display a list of the most popular stories. Some examples of such websites are Google Blog Search, Yahoo! Buzz, Digg, Technorati, and Blogpulse. Some of these websites display posts without any manual intervention, e.g., Google Blog Search and Blogpulse. However, most of these websites display posts which have either been handpicked by editors or have been voted for by users of the website. Most websites that pick posts automatically use a combination of features such as link structure (Blogscope), trends in search engine queries (Yahoo! Buzz), and the number of times a post is emailed or shared. Currently, we are only using features derived from the text of the posts, although in the future we hope to incorporate the link structure between posts into our algorithm. Another key difference is that most of these websites lack the personalization functionality we provide.

In a recent paper (Agarwal et al., 2008), Agarwal et. al address a problem similar to ours. Their task is to select four out of a set of sixteen stories to be displayed on the Yahoo! homepage. The sixteen stories are manually picked by human editors; hence, all are of high quality. The authors use click-through rate to learn online models for each article. Their setting differs significantly from ours, since we tackle the problem of selecting ten out of roughly 60,000 posts for each eight hour segment. Moreover, as described in section 4, our data is very noisy, and we do not have access to click-through rates.

Another line of related research is the area of subtopic retrieval (Zhai et al., 2003; Chen & Karger, 2006; Carbonell & Goldstein, 1998). In subtopic retrieval, the task is to retrieve documents that cover many subtopics of the given query. In the traditional information retrieval setting, it is assumed that the relevance of each document is independent of the other documents. However, in subtopic retrieval the utility of a document is contingent on the other retrieved documents. In particular, a newly retrieved document is relevant only if it covers subtopics other than the ones covered by previous documents. Thus, the concept of relevance in subtopic retrieval is similar to our notion of "coverage," which has a diminishing returns characteristic. However, while subtopic retrieval is query-based, we intend to cover all the popular stories being discussed in the blogosphere.

Two common approaches to personalization are *collaborative filtering* (Linden et al., 2003; Das et al., 2007) and *content-based filtering*. In collaborative filtering, user preferences are learned in a content-agnostic manner by correlating the user's past activity with data from the entire user community. In a content-based approach, documents are recommended to a user if they are similar to documents that the user previously liked, where similarity is based on document content. Using a content-based approach, we provide theoretical guarantees for personalization. Moreover, we currently do not have the kind of user base that is needed for collaborative filtering to be effective.

Leskovec et al. propose a solution to the problem of selecting which blogs to read in order to come across all the important stories quickly (Leskovec et al., 2007). Although related to our problem, a fundamental difference is that instead of trying to select which blogs to read, we present the user with a selection of posts from various blogs. Moreover our approach is completely content based, whereas the approach of Leskovec et al. is based only on the links between blogs. In addition, we also incorporate personalization into our algorithm, which they do not.

There has also been extensive work on building models and analyzing the structure of the blogosphere. For example, Finin et al. (Finin et al., 2008) present a model of information flow in the blogosphere. We could potentially leverage such analysis in the future in order to extract better features for our algorithms. Blogscope is intended to be an analysis and visualization tool for the blogosphere. Unlike us, they are not trying to cover the blogosphere. Instead, Blogscope presents the user with a search interface, and suggests some related words based on the search query. They give a preference to words whose frequency increases by a large amount in the past 24 hours (e.g., words with a high "burstiness"). Moreover, they do not employ any personalization.

6 Conclusions

In this paper we describe the problem of *turning down the noise* in the blogosphere. While the vast majority of blog posts are not interesting for the average user, their quantity is truly remarkable. For this reason, many readers suffer from information overload. Our goal is to show them a small set of posts covering only the important stories currently being discussed.

We start by exploring different desired properties of coverage functions. We then formalize the notion of coverage as a submodular optimization problem, and present an efficient algorithm to select the top stories in the blogosphere.

Next, we generalize the coverage notion to the personalized case, where we assume that each user has his own coverage function based on his personal preferences. We introduce the problem of learning these coverage functions from limited user feedback. We formalize the notion of feedback, and illustrate a simple online personalization method based on multiplicative updates of weights. This method achieves no-regret personalization.

We derive two different algorithms based on our general framework, each using different feature instantiations. Both algorithms are efficient enough that they can be run on large, real-world blog feeds. We compare both algorithms against popular blog aggregation websites like Google Blog Search, Yahoo! Buzz, Digg, and BlogPulse. In addition to post content, most of these websites use richer features such as click-through rate, trends in search queries and link structure between posts, or use human intervention to pick posts. We present results based on simulations and a user study. Our TDN algorithm outperforms all others except for Yahoo! Buzz (with which it is comparable), despite having access to text-based features only. Furthermore, our experiments demonstrate that our algorithm can adapt to individual users' preferences.

Our results emphasize that the simple notion of coverage we introduced successfully captures the salient stories of the day. We believe that this combination of coverage and personalization will prove to be a useful tool in the battle against information overload.

Acknowledgments. We thank Spinn3r for providing us access to their data. We are grateful to Geoff Gordon for helpful discussions and to the reviewers for their useful comments. Tag clouds in this paper were generated using wordle.net. This work was partially supported by the ARO under MURI W911NF0710287 and W911NF0810242, by NSF Career IIS-0644225, and by NSF NeTS-NOSS CNS-0625518.

References

Blogpulse, http://blogpulse.com.

Blogscope, http://www.blogscope.net.

Digg, http://digg.com.

Google Blog Search, http://blogsearch.google.com.

Spinn3r, http://spinn3r.com.

Technorati, http://technorati.com.

Yahoo! Buzz, http://buzz.yahoo.com.

Agarwal, D., Chen, B.-C., Elango, P., Ramakrishnan, R., Motgi, N., Roy, S., & Zachariah, J. (2008). Online models for content optimization. *NIPS*.

Blei, D. M., Ng, A. Y., & Jordan, M. I. (2003). Latent Dirichlet allocation. *JMLR*.

Canini, K. R., Shi, L., & Griffiths, T. L. (2009). Online inference of topics with latent Dirichlet allocation. *AISTATS*.

Carbonell, J., & Goldstein, J. (1998). The use of MMR, diversity-based re-ranking for reordering documents and producing summaries. *SIGIR*.

Cesa-Bianchi, N., & Lugosi, G. (2006). *Prediction, learning, and games*. Cambridge University Press.

Chen, H., & Karger, D. (2006). Less is more. *SIGIR*.

Das, A. S., Datar, M., Garg, A., & Rajaram, S. (2007). Google News personalization: scalable online collaborative filtering. *WWW*.

Finin, T., Joshi, A., Kolari, P., Java, A., Kale, A., & Karandikar, A. (2008). The information ecology of social media and online communities. *AI Magazine*.

Finkel, J. R., Grenager, T., & Manning, C. (2005). Incorporating non-local information into information extraction systems by Gibbs sampling. *ACL*.

Freund, Y., & Schapire, R. E. (1999). Adaptive game playing using multiplicative weights. *Games and Economic Behavior*.

Griffiths, T. L., & Steyvers, M. (2004). Finding scientific topics. *PNAS*.

Khuller, S., Moss, A., & Naor, J. (1999). The budgeted maximum coverage problem. *Information Processing Letters*.

Kinsley, M. (2008). How many blogs does the world need? *TIME Magazine*, *172*.

Leskovec, J., Krause, A., Guestrin, C., Faloutsos, C., VanBriesen, J., & Glance, N. (2007). Cost-effective outbreak detection in networks. *KDD*.

Linden, G., Smith, B., & York, J. (2003). Amazon.com recommendations: Item-to-item collaborative filtering. *IEEE Internet Computing*, *7*.

Nemhauser, G., Wolsey, L., & Fisher, M. (1978). An analysis of the approximations for maximizing submodular set functions. *Mathematical Programming*, *14*, 265–294.

Rizzolo, N., & Roth, D. (2007). Modeling discriminative global inference. *ICSC*.

Smith, B. (2007). The hair's still perfect. Politico, April 16.

Zhai, C., Cohen, W. W., & Lafferty, J. (2003). Beyond independent relevance: methods and evaluation metrics for subtopic retrieval. *SIGIR*.

Zhang, B., Li, H., Liu, Y., Ji, L., Xi, W., Fan, W., Chen, Z., & Ma, W.-Y. (2005). Improving web search results using affinity graph. *SIGIR*.

A No-Regret Learning

We cast our problem of learning a user's preferences in the framework of repeated matrix games. Each row i represents a feature. Our goal is to learn a probability distribution $\mathcal{P}$ over the features. Each column $f^{(t)}$ represents the feedback for an ordered set of posts. The only difference from the Freund and Schapire framework (Freund & Schapire, 1999) is that our loss lies in the range $[-0.5, 0.5]$, instead of $[0, 1]$. This is because we define the loss for a cell $(i, f^{(t)})$ in our matrix as,

$$\mathcal{L}(i, f^{(t)}) := -\mathcal{M}(i, f^{(t)}) = \frac{-w_i \ \sum_{a_j \in \mathcal{A}^{(t)}} f_j^{(t)} \quad \textit{inc-cover}_j(a_{1:j-1}, i)}{2 \max_k w_k}.$$

The maximum value of $\sum_{a_j \in \mathcal{A}^{(t)}} f_j^{(t)} \quad \textit{inc-cover}_j(a_{1:j-1}, i)$ is 1, and the minimum value is -1.

Let us denote the above game by $\mathcal{G}$. Consider another matrix game $\mathcal{G}'$ which has the same structure as $\mathcal{G}$. Define the loss function $\mathcal{L}'$ for the game $\mathcal{G}'$ as $\mathcal{L}'(i, f^{(t)}) = \mathcal{L}(i, f^{(t)}) + 0.5$. Thus, by construction, the loss function $\mathcal{L}'$ lies in the range $[0, 1]$. We will show that using multiplicative updates leads to a no-regret algorithm for the game $\mathcal{G}'$, and equivalently for the game $\mathcal{G}$.

Let the initial mixed strategy for game $\mathcal{G}'$ be $\mathcal{P}'_1$, and let $\mathcal{Q}'_t$ be the mixed strategy of the column player (i.e., the environment) at round t. After each round t, we compute a new mixed strategy for the next round using Freund and Schapire's multiplicative update rule[9]:

$$\mathcal{P}'_{t+1}(i) = \mathcal{P}'_t(i)\frac{\beta^{\mathcal{L}'(i,\mathcal{Q}'_t)}}{\mathcal{Z}'_t},$$

where $\mathcal{Z}'_t$ is the normalization factor, and $\beta \in [0,1)$ is a parameter of the algorithm.

Similarly, let the initial mixed strategy for game $\mathcal{G}$ be $\mathcal{P}_1$. After each round t, we compute a new mixed strategy for the next round using the analogous multiplicative update rule:

$$\mathcal{P}_{t+1}(i) = \mathcal{P}_t(i)\frac{\beta^{\mathcal{L}(i,\mathcal{Q}_t)}}{\mathcal{Z}_t},$$

where $\mathcal{Z}_t$ is the normalization factor, and $\beta \in [0,1)$ is a parameter of the algorithm. By definition, we see that,

$$\begin{aligned}\mathcal{Z}_t =& \sum_{i=1}^{n} \mathcal{P}_t(i)\beta^{\mathcal{L}(i,\mathcal{Q}_t)},\\ =& \sum_{i=1}^{n} \mathcal{P}_t(i)\beta^{\mathcal{L}'(i,\mathcal{Q}'_t)-0.5},\\ =& \frac{1}{\sqrt{\beta}}\sum_{i=1}^{n} \mathcal{P}_t(i)\beta^{\mathcal{L}'(i,\mathcal{Q}'_t)}.\end{aligned}$$

Thus, if $\mathcal{P}_t(i) = \mathcal{P}'_t(i)$, then $\mathcal{Z}_t = \frac{1}{\sqrt{\beta}}\mathcal{Z}'_t$. Also, we notice that,

$$\begin{aligned}\mathcal{P}_{t+1}(i) =& \mathcal{P}_t(i)\frac{\beta^{\mathcal{L}(i,\mathcal{Q}'_t)}}{\mathcal{Z}_t},\\ =& \mathcal{P}_t(i)\frac{\beta^{\mathcal{L}'(i,\mathcal{Q}'_t)-0.5}}{\mathcal{Z}_t},\\ =& \frac{1}{\sqrt{\beta}}\mathcal{P}_t(i)\frac{\beta^{\mathcal{L}'(i,\mathcal{Q}'_t)}}{\mathcal{Z}_t}.\end{aligned}$$

If $\mathcal{P}_t(i) = \mathcal{P}'_t(i)$, then,

$$\begin{aligned}\mathcal{P}_{t+1}(i) =& \frac{1}{\sqrt{\beta}}\mathcal{P}'_t(i)\frac{\beta^{\mathcal{L}'(i,\mathcal{Q}'_t)}}{\frac{1}{\sqrt{\beta}}\mathcal{Z}'_t},\\ =& \mathcal{P}'_t(i)\frac{\beta^{\mathcal{L}'(i,\mathcal{Q}'_t)}}{\mathcal{Z}'_t},\\ =& \mathcal{P}'_{t+1}(i).\end{aligned}$$

Therefore, if we set $\mathcal{P}'_1 = \mathcal{P}_1$, then,

$$\forall t, \mathcal{P}'_t = \mathcal{P}_t. \tag{10}$$

We now use the following theorem from Freund and Schapire:

Theorem A.1. *For any matrix $\mathcal{L}'$ with n rows and entries in $[0,1]$, and for any sequence of mixed strategies $\mathcal{Q}'_1, \ldots, \mathcal{Q}'_T$ played by the environment, the sequence of mixed strategies $\mathcal{P}'_1, \ldots, \mathcal{P}'_T$ produced by the multiplicative weights algorithm satisfies:*

$$\sum_{t=1}^{T} \mathcal{L}'(\mathcal{P}'_t, \mathcal{Q}'_t) \leq \min_{\mathcal{P}'}\left[\alpha_\beta \sum_{t=1}^{T} \mathcal{L}'(\mathcal{P}', \mathcal{Q}'_t) + c_\beta KL(\mathcal{P}'||\mathcal{P}'_1)\right]$$

[9]The reader should note that our original update equation (Eq. 7) is defined directly in terms of $\mathcal{M}$, and thus contains a negative sign. We use loss in this proof to match the convention of Freund and Schapire.

$$\textit{where} \;\; \alpha_\beta = \frac{ln(1/\beta)}{1-\beta},\;\; c_\beta = \frac{1}{1-\beta}.$$

Now, suppose we set $\mathcal{P}'_1 = \mathcal{P}_1$. Then, as a corollary of the above theorem, we can say,

$$\sum_{t=1}^{T} (\mathcal{L}(\mathcal{P}_t, \mathcal{Q}_t) + 0.5) \leq \min_{\mathcal{P}} \left[\alpha_\beta \sum_{t=1}^{T} (\mathcal{L}(\mathcal{P}, \mathcal{Q}_t) + 0.5) + c_\beta KL(\mathcal{P}||\mathcal{P}_1) \right]$$

$$\Longrightarrow \sum_{t=1}^{T} \mathcal{L}(\mathcal{P}_t, \mathcal{Q}_t) \leq \min_{\mathcal{P}} \left[\alpha_\beta \sum_{t=1}^{T} \mathcal{L}(\mathcal{P}, \mathcal{Q}_t) + c_\beta KL(\mathcal{P}||\mathcal{P}_1) \right] + 0.5\ T\ (\alpha_\beta - 1).$$

If we set $\mathcal{P}'_1 = \mathcal{P}_1$ to the uniform distribution over the n rows of the matrix, we obtain,

$$\sum_{t=1}^{T} \mathcal{L}(\mathcal{P}_t, \mathcal{Q}_t) \leq \min_{\mathcal{P}} \left[\alpha_\beta \sum_{t=1}^{T} \mathcal{L}(\mathcal{P}, \mathcal{Q}_t) + c_\beta\ ln\ n \right] + 0.5\ T\ (\alpha_\beta - 1). \tag{11}$$

Corollary A.2. *If we set β to,*

$$\frac{1}{1+\sqrt{\frac{2\ ln\ n}{T}}},$$

the average per-trial loss suffered by the learner is,

$$\frac{1}{T}\sum_{t=1}^{T} \mathcal{L}(\mathcal{P}_t, \mathcal{Q}_t) \leq \min_{\mathcal{P}} \frac{1}{T}\sum_{t=1}^{T} \mathcal{L}(\mathcal{P}, \mathcal{Q}_t) + O\left(\sqrt{\frac{ln\ n}{T}}\right).$$

Proof. The right hand side of Eq. 11 consists of three components. We look at each one of them in turn. We first look at α_β.

For $\beta \in (0,1]$, by log series expansion we get,

$$\begin{aligned}
-ln\ \beta &= -2\sum_{n=0}^{\infty} \frac{1}{2n+1}\left(\frac{\beta-1}{\beta+1}\right)^{2n+1}, \\
&= 2\sum_{n=0}^{\infty} \frac{1}{2n+1}\left(\frac{1-\beta}{1+\beta}\right)^{2n+1}, \\
&\leq 2\sum_{n=0}^{\infty} \left(\frac{1-\beta}{1+\beta}\right)^{2n+1}, \\
&= 2\left(\frac{1-\beta}{1+\beta}\right)\frac{1}{1-\left(\frac{1-\beta}{1+\beta}\right)^2}, \\
&= 2\frac{(1-\beta)(1+\beta)}{(1+\beta)^2-(1-\beta)^2}, \\
&= \frac{1-\beta^2}{2\beta}.
\end{aligned}$$

Therefore,

$$\alpha_\beta = \frac{-ln\ \beta}{1-\beta} \leq \frac{1+\beta}{2\beta} = \frac{1}{2\beta} + \frac{1}{2} = 1 + \sqrt{\frac{ln\ n}{2T}}. \tag{12}$$

We now look at the second term in Eq. 11:

$$c_\beta ln\ n = \frac{1}{1-\beta} ln\ n = \left(1+\sqrt{\frac{T}{2ln\ n}}\right) ln\ n = ln\ n + \sqrt{\frac{Tln\ n}{2}}. \tag{13}$$

We now look at the third term in Eq. 11. By Eq. 12,

$$0.5T(\alpha_\beta - 1) \leq 0.5T\sqrt{\frac{ln\ n}{2T}}. \tag{14}$$

From Eq. 11, Eq. 12, Eq. 13, Eq. 14, and the fact that $\mathcal{L}(.,.) \leq 1$, we obtain,

$$\begin{aligned}
\frac{1}{T}\sum_{t=1}^{T}\mathcal{L}(\mathcal{P}_t, \mathcal{Q}_t) &\leq \min_{\mathcal{P}} \frac{1}{T}\left[\alpha_\beta \sum_{t=1}^{T}\mathcal{L}(\mathcal{P}, \mathcal{Q}_t) + c_\beta\ ln\ n\right] + \frac{1}{T}0.5\ T\ (\alpha_\beta - 1), \\
&\leq \min_{\mathcal{P}} \left[\left(\frac{1}{T}\sum_{t=1}^{T}\mathcal{L}(\mathcal{P}, \mathcal{Q}_t) + \sqrt{\frac{ln\ n}{2T}}\right) + \left(\frac{ln\ n}{T} + \frac{1}{T}\sqrt{\frac{Tln\ n}{2}}\right)\right] + 0.5\sqrt{\frac{ln\ n}{2T}}, \\
&= \min_{\mathcal{P}} \frac{1}{T}\sum_{t=1}^{T}\mathcal{L}(\mathcal{P}, \mathcal{Q}_t) + O\left(\sqrt{\frac{ln\ n}{T}}\right).
\end{aligned}$$

□

The learning rate β depends on the number of rounds, T. As we often do not know this number in advance, one common approach, suggested by Freund and Schapire, is to divide the sequence into segments of increasing length, where the kth segment has length $T_k = k^2$. The learning rate for segment k can now be defined based on T_k, rather than the (unknown) total number of rounds, T. Freund and Schapire provide more details on the theoretical guarantees of this technique (Freund & Schapire, 1999).

B Data Preprocessing

In order to use the blog feeds gathered from Spinn3r, a series of preprocessing steps are first employed to clean the data. Spinn3r categorizes all posts as belonging to one of four categories: Weblog, Mainstream News, Forum, and Classified. We remove any posts that are tagged with the Forum or Classified labels, as we assume that users would not wish to be presented with such posts. This step reduces the number of posts by approximately half. Additionally, a short blacklist (121 sites) is used to filter out sites that are known to contain spam or that were misclassified as blogs. Finally, using a standard shingling approach, near-duplicate posts that appear *in the same blog* are removed. This allows us to deal with a common manifestation of post content extraction errors, as well as spam blog sites that often contain many duplicate posts. Note that we do not remove duplicate posts *across different blogs*, as these provide important information regarding the prevalence of a particular story.

At this point, the named entity recognizer and part of speech tagger are used to extract all the named entities and noun phrases, as described in Section 4. After the standard steps of removing stop words and stemming the nouns, there are still over 20,000 total features. Retaining all of these features is wasteful, as most do not add any value, either because they rarely occur in the corpus or because they are uninformative for other reasons. Specifically, we perform the following feature selection steps on each eight hour epoch of data:

1. Select the 2,000 most frequent named entities, and discard the rest.

2. Select the 2,100 most frequent noun phrases, and discard the rest. (The first two steps discard rarely occurring features.)

3. Discard the 200 most frequent noun phrases, as they tend to be uninformative (e.g., “Post” or “Comment”).

4. For each of the remaining noun phrases, compute how often it appears in each post in the corpus. Calculate the mean and variance of these counts for each noun phrase.

5. Use the statistics calculated in the previous step to prune away uninformative noun phrases. For instance, noun phrases with an extremely low variance of occurrence (i.e., whenever they occur in a document, they occur the same number of times), are often indicative of "boilerplate" text incorrectly parsed as post content (e.g., the word "copyright"). As another example, nouns that have a high average frequency of occurrence (i.e., whenever they occur, they occur many times) may indicate spam. Based on cursory observations, we empirically selected cutoff values[10] for these two statistics that allowed us to reduce the number of noun phrases we keep to below 1,000.

After this process, we are left with slightly fewer than 3,000 features per eight hour epoch, upon which we then run LDA.

[10]We keep noun phrases whose average frequency of occurrence in a post is in the range (1.3, 6), and whose variance is in the range (0.25, 12).