Online Selection of Diverse Results

Debmalya Panigrahi*
CSAIL, MIT
Cambridge, MA
debmalya@mit.edu

Atish Das Sarma, Gagan Aggarwal, Andrew Tomkins
Google Research
Mountain View, CA
{dassarma, gagana, tomkins}@google.com

ABSTRACT

The phenomenal growth in the volume of easily accessible information via various web-based services has made it essential for service providers to provide users with personalized representative summaries of such information. Further, online commercial services including social networking and micro-blogging websites, e-commerce portals, leisure and entertainment websites, etc. recommend interesting content to users that is simultaneously diverse on many different axes such as topic, geographic specificity, etc. The key algorithmic question in all these applications is the generation of a succinct, representative, and relevant summary from a large stream of data coming from a variety of sources. In this paper, we formally model this optimization problem, identify its key structural characteristics, and use these observations to design an extremely scalable and efficient algorithm. We analyze the algorithm using theoretical techniques to show that it always produces a nearly optimal solution. In addition, we perform large-scale experiments on both real-world and synthetically generated datasets, which confirm that our algorithm performs even better than its analytical guarantees in practice, and also outperforms other candidate algorithms for the problem by a wide margin.

Categories and Subject Descriptors

H.3 [**Information Systems**]: Information Storage and Retrieval

General Terms

Algorithms, Performance

Keywords

Result Diversity, Online Algorithm

1. INTRODUCTION

The problem of result-set diversity has been studied in multiple domains, but perhaps the most robust literature exists in web search. In search, it is common to introduce diversity by mixing in different interpretations of a query, as in [11], or by attempting to construct result sets to be pairwise distant, as in [3]. The goal in these instances is largely to cater to the needs of multiple distinct user types, each of whom enter the same query, but with different goals. The system may provide a diverse result set to meet the needs of different types simultaneously, even though a single user might be happiest with a non-diverse result set. An alternate form of the diversity problem, in which each individual user prefers a diverse result set, presents itself in applications such as providing ongoing novelty in a series of entertaining items, or a balance of opinions on a sensitive topic.

Specifically, we consider the problem of selecting subsets of items that are simultaneously diverse in multiple dimensions. This problem arises in a variety of contexts. As a toy example, consider the task of selecting the program committee for this conference. The chairs were faced with the task of producing a committee that offers sufficient expertise in all important topics of the conference, while simultaneously covering different regions of the world, providing a good mix of gender, seniority, industry versus academia, and so forth.

Many other examples exist in natural settings. An e-commerce website may wish to show televisions covering multiple technologies, screen dimensions, manufacturers, and price points. Or a website offering pre-packaged vacation travel may be interested in showing customers a small set of options that cover a wide range of alternatives: domestic or international; cheap or expensive; family-friendly or romantic; and so on. Similarly, as an example we will consider more closely below, consider a system recommending interesting content to a user. The system aims to provide content that is simultaneously diverse along a variety of axes including topic (sports vs. politics vs. technology), geographic specificity (world news vs. national news), voice (humorous, scandalous, contrapuntal), and media type (videos vs. blog articles).

In some of these applications, the itemset presented to a user must be personalized to cater to her individual tastes and preferences. Selecting a distinct itemset for each individual user from a very large corpus of items is particularly challenging from a computational perspective. To alleviate this computational bottleneck, we suggest the following natural two-step process: use a very efficient algorithm to select a single representative itemset of intermediate size that caters to the tastes of all users, and then generate individual (smaller) user summaries from this representative set using more resource-intensive techniques. Since user preferences are typically very diverse, it is important that the intermediate itemset contains a diverse set of items covering the entire range of user interests. Therefore, our techniques for selecting a diverse itemset

*Part of this work was done while the author was an intern at Google Research, Mountain View, CA.

WSDM'12, February 8–12, 2012, Seattle, Washington, USA.

are useful even for applications such as news feeds in social networking websites that involve a high degree of personalization.

Note that it is important to be simultaneously diverse in all dimensions; in our program committee example, for instance, no amount of diversity in national origin will rescue the hapless committee constituted entirely of experts in itemset discovery. Similarly, geographical diversity of news feeds in a news aggregator website will not compensate for lack of diversity in the topics spanned by the articles. Hence, while we will discuss other metrics, our results will focus on maximizing the worst-case coverage over all features.

Further, in most web/internet based applications, the system must make a commitment to certain items before other items are available for consideration. This may occur because candidates truly arrive in an online manner over time, or because a large dataset is to be processed in a streaming manner, or more commonly, because each page of results must be produced at the lowest possible latency. A user should be provided with a diverse experience over many pageviews, and perhaps even many sessions, but the first response must be committed before future requests are known. Hence, an algorithm that operates purely in an offline context may not extend naturally to each successive request.

This variant of the problem arises in content recommendation, news feeds applications on news websites, search-type interfaces for web search, e-commerce search (as at `amazon.com`), item search (as at `ebay.com`), and so forth. We are primarily motivated by the problem of providing a stream of interesting content, where the items need to be presented in a timely manner with consideration for a variety of attribute features, each of which could have varying ranges and densities. Therefore (irrevocable) decisions on whether new items should be shown or skipped need to be made in an online manner.

We abstract this problem as follows. A stream of items, each decorated with features, arrives over time. As each item arrives, the algorithm must accept or decline it. The algorithm must select as many items as specified by a budget, and its score is the coverage of the least populous feature in the final set.

The content recommendation problem by which we are motivated differs from the abstract formulation in few key ways; we recap these differences here. First, the diversifier will typically have access to constant-size batches of input, rather than individual items. However, the asymptotic behavior of our problem is unchanged in this case, so we may focus our attention on the simpler variant. Next, the diversifier is required to "trade off" diversity with the quality of items selected into the result set. There are several natural mechanisms to incorporate item quality into the formulation, which we describe later. Finally, we study coverage of the worst feature, but in certain settings it may be important to study other variants, such as the average feature. We survey the results for other norms, and also present empirical results analyzing our algorithms with respect to low-performing features other than the worst.

Our Contributions. We now give a high-level overview of the contributions of this paper.

1. We formalize the problem of online diversification on multi-dimensional featured items. In our problem formulation, the objective is to maximize the minimum coverage over all features, but we also present simple results for other variants of this formulation, both in the offline and the online settings (cf. Sections 2 and 3).

2. We theoretically analyze our problem formulation by showing hardness results for specific small-coverage and adversarial input instances. We then present a novel algorithm for the diversification problem. As the main theoretical result of our paper (Theorem 6), we prove that the algorithm achieves an approximation ratio of 50% on the objective function, provided the optimum coverage is large enough, and the items are drawn *independently* and *identically* (i.i.d.) from some probability distribution (that is unknown to the algorithm). The theoretical analysis and techniques presented here are novel, and the main algorithm is simple, easy to implement, and lightweight, and therefore potentially applicable to a very wide range of scenarios (cf. Section 4).

3. We perform comprehensive experimental evaluation on real-world data obtained from a commercial news feed generator, as well as several synthetically generated data sets. The experimental results not only conclusively corroborate our theoretical findings but also show that our algorithm performs significantly better than its analytical guarantees on all data sets. In fact, the performance of the algorithm is close to optimal for the minimum coverage feature and very good even for subsequent low-coverage features (cf. Section 5). Further, the algorithm outperforms a set of natural and intuitive algorithms for the diversification problem by a wide margin.

2. DIVERSIFICATION FUNCTIONS

As described in the introduction, our high-level objective is to select a small collection of items that are diverse with respect to their constituent features or dimensions, from a large corpus of multi-dimensional items. In this section, we discuss multiple variants of the problem, all of which represent the high-level goal of diversification, and ultimately converge to a particular problem formulation that we focus on for the rest of the paper.

First, we establish some notation that we will use throughout the paper. Let F be the set of n features. As described in the introduction, the input consists of a set U of m items, where each item $j \in U$ consists of a subset of features $F_j \subseteq F$. The diversification algorithm needs to select a representative subset S of at most B items, where B is a given budget, from the input set U. The *coverage* of feature $i \in F$ in the selected subset S, denoted by C_i, is defined as the number of items in S that have feature i. Each feature also has a *target* T_i which is the desired coverage for the feature. The *fractional coverage* of feature i is the fraction of its target coverage that has been achieved by the selected set of items, i.e. $c_i = C_i/T_i$. Let $\mathbf{c}$ be the vector of fractional coverages, i.e. $\mathbf{c} = (c_i : i \in F)$; then the objective is to select a subset S that maximizes the value of $D(\mathbf{c})$, where D is the diversification function of interest.

We consider two versions of the diversification problem depending on whether the entire set U is available to the algorithm before it starts selecting the items in S. For example, in selecting a program committee from a set of researchers, the entire set of researchers is known to the program committee chairs before any selection decision is taken. We call this the *offline* version of the problem. On the other hand, consider the diversification problem in generating news feeds. In this case, the diversification algorithm needs to select news items as they arrive, i.e. without having access to the entire input set of items. We call this the *online* version of the problem. Thus, in the online model, on the arrival of an item j, the algorithm must immediately either select or discard it, subject to the constraint that the total number of selected items cannot exceed B. We assume that each item in the online input stream is drawn i.i.d. from some probability distribution on a set of features that is *unknown* to the algorithm.

Perhaps the simplest objective function D that one can aim for while selecting a representative subset from a large set of items is to maximize the sum of fractional coverages of all the features, i.e.

$$D(\mathbf{c}) = \sum_{i \in F} c_i.$$

However, observe that this function fails to distinguish between a subset of items that achieves large coverage on a few features but very small coverage on the remaining features, and a different subset of items that achieves uniform moderate coverage on all features. Intuitively, the second subset is clearly more diverse, and hence should be preferred. A diversification function that reflects this intuition is

$$D(\mathbf{c}) = \sum_{i \in F} p_i,$$

where $p_i = 1$ if $c_i > 0$, and $p_i = 0$ otherwise. This function clearly distinguishes between the two subsets of items described above, but has the shortcoming that it treats all non-zero coverage values identically. In fact, these two functions are the extreme ends of a continuum of candidate functions

$$D_\alpha(\mathbf{c}) = \sum_{i \in F} c_i^\alpha, \quad 0 \leq \alpha \leq 1$$

that represents the classical trade-off between maximizing *magnitude* (cf. the first function, i.e. $\alpha = 1$) and ensuring *fairness* (cf. the second function, i.e. $\alpha = 0$).

By a slight abuse of notation, let us also denote the value of the function D_α on the coverage achieved by a set of selected items S as $D_\alpha(S)$. It can be shown that all such functions $D_\alpha(S)$ (for $0 \leq \alpha \leq 1$) are *monotonically increasing submodular*[1] functions. Consider a *greedy* algorithm that repeatedly selects the item that yields the maximum increase in the value of the objective until the entire budget has been used up. It is well-known that this algorithm has an approximation factor of $(1 - 1/e)$ for the problem of maximizing any monotonically increasing submodular function.

THEOREM 1. *For any α between 0 and 1, the greedy algorithm has an approximation factor of $(1-1/e)$ for maximizing D_α in the offline setting.*

In the online setting, if the optimal value of objective function is known, then a standard thresholding technique yields an algorithm with a constant competitive ratio[2]. On the other hand, if the optimum is unknown, then we can guess its value using a standard doubling technique. The key property that we exploit in this guessing scheme is that the input stream is drawn i.i.d., and therefore only a small fraction of the budget is used in obtaining a good estimate of the optimum. (The proof of this theorem is deferred to the full version of the paper.)

THEOREM 2. *For any α between 0 and 1, there exists an algorithm that has a constant competitive ratio for maximizing D_α in the online setting, if the input is drawn i.i.d.*

[1]A function f defined on all subsets of a ground set X is said to be *monotonically increasing* if for any $A \subseteq B$,

$$f(A) \leq f(B),$$

and is said to be *submodular* if for any $A \subseteq B$ and for any $x \in X$,

$$f(A \cup \{x\}) - f(A) \geq f(B \cup \{x\}) - f(B).$$

[2]For a maximization problem, an online algorithm has a competitive ratio of β if the objective value in the solution produced by the algorithm is at least β-times the offline optimum.

We now focus on another natural candidate function for diversification, where the objective is to maximize the minimum fractional coverage all features. That is,

$$D(\mathbf{c}) = \min_{i \in F} c_i$$

Observe that this function achieves the twin objectives of *magnitude* and *fairness* of feature coverage. This function is **not** submodular, and therefore, the techniques described above cannot be used to solve this problem.

Let c_{OPT} be the optimal value of D and

$$\rho_{\text{OPT}} = c_{\text{OPT}} \min_{i \in F} T_i.$$

The next theorem (proof deferred to the full version) shows that the problem does not admit an algorithm with a finite approximation ratio even in the offline setting, if $\rho_{\text{OPT}} = o(\log n)$.

THEOREM 3. *Under standard complexity-theoretic assumptions, there exists no algorithm that obtains a finite approximation ratio for offline instances of the diversification problem where $\rho_{\text{OPT}} = o(\log n)$.*

The above theorem implies that we need to assume that the optimal solution satisfies $\rho_{\text{OPT}} = \Omega(\log n)$, in order to obtain a finite approximation ratio. (In fact, this assumption holds for most real data sets, as verified later in the experimental section.) If this property is satisfied by an offline instance of the problem, then a simple algorithm that employs randomized rounding of the natural linear programming formulation of the problem gives the following theorem. (The proof of this theorem is deferred to the full version of the paper.)

THEOREM 4. *For the problem of maximizing D in the offline setting, there is a PTAS[3] if the optimum is $\Omega(\log n)$.*

Now, we focus on the online version of the problem. The next theorem (proof deferred to the full version) shows that we need to assume that the input stream is not adversarial in order to obtain a sub-polynomial competitive ratio for this problem.

THEOREM 5. *For an adversarial input stream, the competitive ratio of any algorithm for maximizing D in the online setting is $\Omega(n)$.*

To overcome the barrier imposed by the above theorem, we assume that the input is drawn i.i.d. from a probability distribution that is unknown to the algorithm.

3. PROBLEM FORMULATION

As described in the previous section, we focus on the following problem formulation, which we call the DIVERSIFIER problem.

> Let F be a set of features and T_i be the target coverage for feature $i \in F$. An input set U of m items arrives online, where the set of features $F_j \subseteq F$ in each item $j \in U$ is drawn i.i.d. from a probability distribution on subsets of F that is unknown to the algorithm. The algorithm must decide, on the arrival of item j, whether to select or discard it. The overall goal is to select a subset S of at most B items that maximizes $\min_{i \in F} c_i = \min_{i \in F} \frac{C_i}{T_i}$, where C_i is the number of items in S that contain feature i.

[3]A *Polynomial-time Approximation Scheme* (or PTAS) for a maximization problem is an algorithm that has an approximation factor of $(1 - \epsilon)$ for any arbitrarily small constant $\epsilon > 0$. (The running time of the algorithm depends on the choice of ϵ.)

Main Result. We give an online algorithm for the DIVERSIFIER problem which proves the next theorem.

THEOREM 6. *There is a deterministic online algorithm for the* DIVERSIFIER *problem that has a competitive ratio of $\frac{1}{2} - \delta$ for any $\delta > 0$ with probability (over the input distribution) at least $1 - 1/n$, provided the input is drawn i.i.d. from an (unknown) probability distribution on feature sets satisfying the property that the expected value of $\rho_{\text{OPT}} \geq \frac{24 \ln n}{\delta^2}$.*

Our Techniques. Consider the special case where all targets T_i are equal to the budget B. Further, assume that we have the guarantee that the expected value of $\rho_{\text{OPT}} = c_{\text{OPT}} B = \Omega(\log^2 n)$ (which is stronger than that required by Theorem 6). Then, we can partition the input into $\log n$ *epochs*, where in each epoch, the algorithm selects at most $B/\log n$ subsets from an input stream containing $m/\log n$ items, and aims at achieving an expected minimum coverage of $\lambda = \Omega(\frac{\rho_{\text{OPT}}}{\log n})$.

Now, instead of achieving a coverage of λ for each feature, let us change our goal in any epoch to achieving a cumulative coverage of $\Omega(n\lambda)$ over all features, where the contribution of any single feature to this sum is at most λ, i.e. $\sum_{i \in F} \min(C_i, \lambda) = \Omega(n\lambda)$. This can be achieved by using a thresholding algorithm that selects an item *if and only if* it contains at least $\Omega\left(\frac{n\lambda \log n}{B}\right)$ features $i \in F$ with current coverage $C_i < \lambda$. This immediately implies, via an averaging argument, that some constant fraction of features have achieved a coverage of $\Omega(\lambda)$. We discard these features in the next epoch and recurse. Since the number of retained features decreases by a constant factor in every epoch, the coverage on every feature is $\Omega(\lambda)$ at the end of $\log n$ epochs. Therefore, this algorithm yields a competitive ratio of $O(\log n)$.

To transform the algorithm described above to an algorithm that proves Theorem 6, we need to make the following improvements:

- Improve the competitive ratio of the algorithm to a constant.
- Generalize the algorithm to handle arbitrary targets T_i.
- Relax the constraint on the expected value of ρ_{OPT} from $\Omega(\log^2 n)$ to $\Omega(\log n)$.

The previous algorithm can be interpreted in terms of a reward function which gives a reward of 1 every time a feature is covered until the feature has been covered λ times, at which point the reward on covering the feature drops to 0. The algorithm then essentially sets a threshold proportional to the ratio of the remaining rewards to the remaining budget, and selects an item if it meets this reward threshold. However, observe that the algorithm fails to differentiate between covering a feature that already has a large coverage (but less than λ), and a feature that has smaller coverage. To make this distinction, we introduce a smoother reward function in the next section, and show that this simple thresholding algorithm for the new reward function achieves all the three goals outlined above.

4. THE DIVERSIFICATION ALGORITHM

In this section, we describe the diversification algorithm, and use analytical techniques to show that it proves Theorem 6. The algorithm uses the expected optimal value of the objective function denoted by c_{OPT}. If c_{OPT} is not known, we can guess c_{OPT}, and update our guess repeatedly by doubling, as outlined in the previous section. Since the input stream of items is drawn i.i.d. from some (unknown) probability distribution, it can be shown that the competitive ratio of the algorithm remains a constant even if the expected optimal value of the objective function is not known to the algorithm. However, for the sake of simplicity, we assume throughout that we know c_{OPT}.

$$\begin{aligned} \sum_{S \subseteq F : i \in S} w_S p_S &\geq \frac{c_{\text{OPT}} T_i}{m} \quad \forall\, i \in F \\ \sum_{S \subseteq F} w_S p_S &\leq \frac{B}{m} \\ 0 \leq w_S &\leq 1 \quad \forall\, S \subseteq F \end{aligned}$$

Figure 1: A linear program for the DIVERSIFIER problem.

As sketched in the previous section, our algorithm uses a reward function ϕ defined as

$$\phi(k) = n^{-\alpha(k/c_{\text{OPT}})},$$

where α is a constant that we will fix later. We also define

$$\bar{\Phi}(k) = \int_{j=k}^{\infty} \phi(j) dj.$$

If the current collection of selected items has fractional coverage c_i for some feature $i \in F_j$ for the current item j, then the reward r_{ij} of item j due to feature i is defined as $\phi(c_i)$. The overall reward r_j of item j is defined as the sum of rewards of its constituent features, i.e. $r_j = \sum_{i \in F_j} r_{ij}$. At any stage of the algorithm, the remaining reward for feature i is

$$\bar{\Phi}_i = \bar{\Phi}(c_i),$$

and the overall remaining reward is

$$\bar{\Phi} = \sum_{i \in F} \bar{\Phi}_i.$$

The online algorithm selects the current item j *if and only if*

$$r_j \geq \frac{\bar{\Phi} \ln n}{\gamma B},$$

where γ is a constant we will fix later. The algorithm terminates when either the input stream has been exhausted or the algorithm has already selected B items.

Analysis. First, we state a property of the reward function that we will use later in the analysis of the algorithm.

FACT 1. *For any $k \geq 0$, $\bar{\Phi}(k) = \left(\frac{c_{\text{OPT}}}{\alpha \ln n}\right) \phi(k)$.*

Now, consider the linear program (LP) for the DIVERSIFIER problem in Fig. 1. Here, p_S denotes the probability of item j in the input stream having $F_j = S$ for any j, and w_s is the fraction to which such an item is chosen in the optimal fractional (offline) solution. Since the expected optimal value of the objective is c_{OPT}, this LP is feasible.

Recall that $\rho_{\text{OPT}} = c_{\text{OPT}} \min_{i \in F} T_i$. For simplicity of notation, let us also denote the expected value of ρ_{OPT} by ρ_{OPT} itself in the rest of this section. The next lemma lower bounds the probability that an item is chosen by our algorithm if it has not already exhausted its budget.

LEMMA 1. *At any stage of the algorithm, the expected decrease in $\bar{\Phi}$ for the next item in the input stream is at least $\left(\alpha - \frac{1}{\gamma}\right) \frac{\bar{\Phi} \ln n}{m}$. Further, the probability that the next item in the input stream is selected by the algorithm is at least $\left(1 - \frac{1}{\alpha\gamma}\right) \frac{\rho_{\text{OPT}}}{m}$.*

PROOF. Consider a hypothetical algorithm that chooses item j having a feature set $F_j = S$ with probability w_S. The expected decrease of $\bar{\Phi}$ for this algorithm at any stage is

$$\begin{aligned}
&\sum_{S\subseteq F} p_S w_S \sum_{i\in S} \frac{\phi(c_i)}{T_i} \\
&= \left(\frac{\alpha \ln n}{c_{\text{OPT}}}\right) \sum_{i\in F} \frac{\bar{\Phi}_i}{T_i} \sum_{S\subseteq F: i\in S} w_S p_S \\
&\geq \left(\frac{\alpha \ln n}{c_{\text{OPT}}}\right) \sum_{i\in F} \frac{\bar{\Phi}_i}{T_i} \left(\frac{c_{\text{OPT}} T_i}{m}\right) \\
&= \left(\frac{\alpha \ln n}{m}\right) \sum_{i\in F} \bar{\Phi}_i \\
&= \left(\frac{\alpha \ln n}{m}\right) \bar{\Phi}.
\end{aligned}$$

Let

$$y_S = \sum_{i\in S} \frac{\phi(c_i)}{T_i}$$

and

$$z_S = p_S w_S (m/B).$$

Then, we have

$$\begin{aligned}
\sum_{S\subseteq F} y_S z_S &\geq \left(\frac{\alpha \ln n}{B}\right) \bar{\Phi} \\
\sum_{S\subseteq F} z_S &\leq 1.
\end{aligned}$$

By standard convexity arguments, we can conclude that

$$\sum_{S\subseteq F: y_S \geq \frac{\bar{\Phi}\ln n}{\gamma B}} y_S z_S \geq \left(\alpha - \frac{1}{\gamma}\right) \frac{\bar{\Phi}\ln n}{B},$$

which implies that the expected decrease in $\bar{\Phi}$ due to the next item in the input stream is

$$\sum_{S\subseteq F: y_S \geq \frac{\bar{\Phi}\ln n}{\gamma B}} w_S p_S y_S \geq \left(\alpha - \frac{1}{\gamma}\right) \frac{\bar{\Phi}\ln n}{m}.$$

Further, the maximum decrease in $\bar{\Phi}$ due to a single item is

$$\max_{S\subseteq F} \sum_{i\in S} \frac{\phi(c_i)}{T_i} \leq \sum_{i\in F} \frac{\phi(c_i)}{T_i} = \left(\frac{\alpha \ln n}{c_{\text{OPT}}}\right) \sum_{i\in F} \frac{\bar{\Phi}_i}{T_i} \leq \left(\frac{\alpha \ln n}{\rho_{\text{OPT}}}\right) \bar{\Phi}.$$

Since $w_S \leq 1$ for all $S \subseteq F$,

$$\sum_{S\subseteq F: y_S \geq \frac{\bar{\Phi}\ln n}{\gamma B}} p_S \geq \sum_{S\subseteq F: y_S \geq \frac{\bar{\Phi}\ln n}{\gamma B}} p_S w_S \geq \left(1 - \frac{1}{\alpha\gamma}\right) \frac{\rho_{\text{OPT}}}{m}.$$

□

The above lemma implies that if the algorithm has not selected B items already, then the next item in the input stream is selected with probability

$$p \geq \left(1 - \frac{1}{\alpha\gamma}\right) \frac{\rho_{\text{OPT}}}{m},$$

and if the next item is selected, then the value of $\bar{\Phi}$ decreases to at most

$$\left(1 - \frac{1}{p}\left(\alpha - \frac{1}{\gamma}\right) \frac{\ln n}{m}\right) \bar{\Phi} \leq n^{-\frac{\alpha - \frac{1}{\gamma}}{pm}} \bar{\Phi}.$$

The next lemma asserts that if $\rho_{\text{OPT}} = \Omega(\ln n)$ and the algorithm does not select B items, then the value of $\bar{\Phi}$ when the algorithm terminates is small.

LEMMA 2. *Suppose the algorithm does not select B items. Further, let $\rho_{\text{OPT}} \geq \frac{3\ln n}{\epsilon^2\left(1-\frac{1}{\alpha\gamma}\right)}$. Then, the value of $\bar{\Phi}$ when the algorithm terminates is at most $n^{1-(1-\epsilon)\left(\alpha-\frac{1}{\gamma}\right)}$ with probability at least $1 - 1/n$.*

PROOF. The expected number of items selected by the algorithm is

$$pm \geq \left(1 - \frac{1}{\alpha\gamma}\right) \rho_{\text{OPT}} \geq \frac{3\ln n}{\epsilon^2}.$$

Therefore, by Chernoff bounds [10], the value of $\bar{\Phi}$ when the algorithm terminates is at most

$$n^{-\left(\frac{\alpha-\frac{1}{\gamma}}{pm}\right)(1-\epsilon)pm} \cdot n = n^{1-(1-\epsilon)\left(\alpha-\frac{1}{\gamma}\right)}$$

with probability at least $1 - 1/n$. □

Finally, we consider the case when the algorithm uses up its entire budget, i.e. selects B items.

LEMMA 3. *If the algorithm selects B items, then the value of $\bar{\Phi}$ when the algorithm terminates is at most $n^{1-\frac{1}{\gamma}}$.*

PROOF. When the algorithm selects an item, the value of $\bar{\Phi}$ decreases to at most

$$\left(1 - \frac{\ln n}{\gamma B}\right) \bar{\Phi} \leq n^{-\frac{1}{\gamma B}}.$$

Therefore, the value of $\bar{\Phi}$ when the algorithm terminates after selecting B items is at most

$$n^{-1/\gamma} \cdot n = n^{1-\frac{1}{\gamma}}.$$

□

We now set $\gamma = \frac{2-\epsilon}{(1-\epsilon)\alpha}$ which lets us summarize the above two lemmas in the following lemma.

LEMMA 4. *If $\rho_{\text{OPT}} \geq \frac{3(2-\epsilon)\ln n}{\epsilon^2}$, then the value of $\bar{\Phi}$ when the algorithm terminates is at most $n^{1-\alpha\left(\frac{1-\epsilon}{2-\epsilon}\right)}$ with probability at least $1 - 1/n$.*

The next lemma bounds the competitive ratio of the algorithm.

LEMMA 5. *If $\rho_{\text{OPT}} \geq \frac{3(2-\epsilon)\ln n}{\epsilon^2}$, then the competitive ratio of the algorithm is at most $\left(\frac{1-\epsilon}{2-\epsilon}\right) - \frac{1}{\alpha}$ with probability at least $1 - 1/n$.*

PROOF. Suppose not, and let $i_{\min}$ be the feature with the minimum fractional coverage at the end of the algorithm. Then,

$$\bar{\Phi} \geq \bar{\Phi}_{i_{\min}} = n^{-\alpha\left(\frac{1-\epsilon}{2-\epsilon} - \frac{1}{\alpha}\right)} > n^{1-\alpha\left(\frac{1-\epsilon}{2-\epsilon}\right)},$$

which violates Lemma 4. □

Observe that since $\epsilon < 1$,

$$\frac{1-\epsilon}{2-\epsilon} > \frac{1}{2} - \epsilon.$$

We now obtain Theorem 6 by setting $\epsilon = \delta/2$ and $\alpha = 2/\delta$.

5. EXPERIMENTS

We perform experiments on both real-world and synthetically generated data sets and compare our diversification algorithm with several natural and intuitive alternative algorithms for the problem. Our experimental evaluation compares the performance of these algorithms measured in terms of feature coverage by varying multiple parameters. We first describe the data sets used in our experiments; next we describe the alternative algorithms we compare our algorithm against; and finally we describe the experiments that we perform and interpret the results we obtain in these experiments.

5.1 Description of Data Sets

Real-world news feeds. This dataset is obtained from the stream of items received by a commercial news feed generator, from which it selects a news feed for individual users. Each item considered has several features associated with it, such as the source of the item, the broad category it belongs to, the types of contents in it etc. We extracted 18 features such that each item has a binary value associated with each feature (that is, either has or does not have the feature). This feature set includes a variety of dense (i.e. present in a large fraction of items) as well as sparse features such as whether the item has an embedded video, is in English language, is about Politics or Sports, and so on. The selection of these 18 features was aimed at capturing multiple kinds of dependencies: e.g., hierarchical dependency (i.e. a feature occurs always with another feature), exclusivity (an item can have only one of a set of features), dense features, or very sparse features. These features were chosen in a careful manner so as to test the performance of the diversification algorithm on dependencies, on sparse features (i.e. even when the optimum is small), on very commonly occurring features, etc. We ran our experiments on 100 such different data sets, each containing about a million items with feature vectors along these 18 dimensions.

Synthetically generated data. In addition to the real data set, we also test our algorithm on various carefully chosen synthetically generated data sets. In each of these data sets, we again generated items with 18 features and tested the algorithms on data sets of one million items each. The synthetic data sets we tested on are the following.

- **Independent.** In this data set, each entry (that is each feature for every item) is independently set to 1 or 0 (that is item either has or does not have the feature) with probability half each. Notice that this generates a data set where, in expectation, each item has nine features. Further, in expectation, each feature is contained in half of all the items.

- **Parity.** In this data set, initially we fix a bit vector of length 18 (where each bit is set to 1 or 0 independently and with equal probability) and then for each item, we pick a 1 or 0 independently and with equal probability, and XOR it with the bit vector. Observe that this results in a very strong dependence between different features, and the whole data set contains only two kinds of items.

- **Dependent Mixed.** This data set is generated in a manner very similar to parity, but in addition, after each item has been generated, each of the feature bits is flipped independently and with a small probability (set to 0.1). This results in a milder dependence between the item features.

- **Dependent.** The dependent data set is similar to the dependent mixed data set. The only difference is that the initial bit vector is set to all 1's. Therefore, there is uniform correlation between all item features.

5.2 Algorithms

We now describe all the algorithms that we compare in our experiments.

Diversifier. This is the diversification algorithm in Section 4.

Diversifier (Uniform Coverage). This algorithm is the same as the diversifier presented previously with a small difference that the reward function ϕ is not decreasing; in particular, the reward function is fixed at $\phi(k) = 1$. Comparing with this algorithm highlights the importance of the decreasing reward function in the diversification algorithm.

Fixed Threshold. The fixed threshold algorithm is a naïve baseline algorithm where a specific threshold is fixed at the beginning (between 1 and 18, the number of features). Subsequently, when items arrive online, every item that has at least as many features as the threshold is picked, until the budget is exhausted. We compare our diversification algorithm against this fixed threshold algorithm for different thresholds. We performed experiments with all possible thresholds between 1 and 18 but present only a representative set of results, for thresholds $3, 6, 9, 12, 15$. The performance of the fixed threshold algorithm with other thresholds is similar to the ones we show.

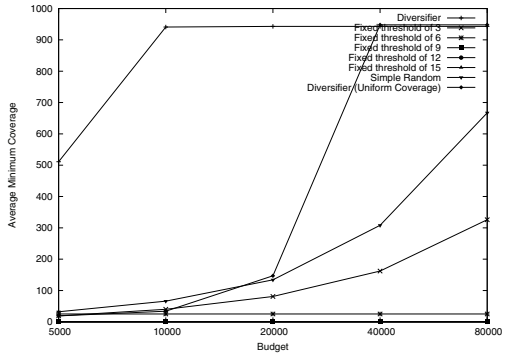
Simple Random. In this algorithm, items are selected randomly based on the number of features they contain, in such a way that the expected number selected items equals the budget B. The probability that an item containing k features is picked is determined by optimally solving a linear program that aims to maximize the coverage on every feature. This algorithm performs two passes over the input stream of items: in the first pass, it computes frequency counts of features and uses them to obtain the probability of selecting an item with k features. In the second pass, it uses these probabilities to actually select the items.

5.3 Description of the Experiments

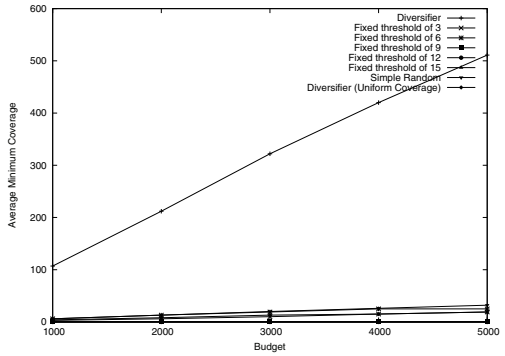
Sorted Coverage. In this experiment, we evaluate the performance of each of the algorithms on the coverage achieved on the features. Recall that our objective is to maximize the minimum coverage over all features. In addition to the minimum coverage, we also look at the performance of the algorithms on the 2nd to 6th least covered features as well. The specific goal of the diversifier is to maximize minimum coverage, but at the same time, it is desirable that the coverage be substantial on other features as well. This experiment highlights that even though the specific goal of the diversification algorithm is to maximize minimum coverage, it achieves substantial coverage on other features as well (particularly for features where a larger coverage is attainable without compromising on the minimum coverage achieved).

Varying Budgets. We perform a series of experiments by varying the available budget to the algorithms, to see if the performance of the diversification algorithm scales with larger budgets. We also perform experiments for low budgets to test whether the algorithm is able to achieve reasonable coverage on each of the features (even on the sparse ones). These experiments show that the diversification algorithm performs admirably for a variety of budgets.

In all our experiments, the total number of features is 18 and the number of items in each data set is around $1M$. Also, when we plot sorted coverage, the default value of the budget is set to $20K$ (which is roughly 2% of the data set), and the targets for all the features are set to the budget itself. For the plots where we

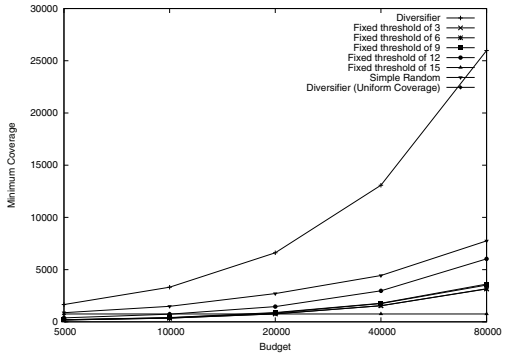


(a) Higher range of budgets between 0.5% and 8%

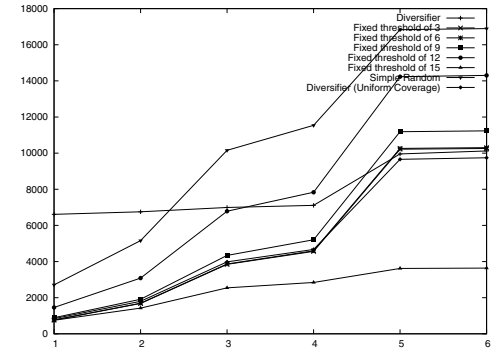


(b) Lower range of budgets between 0.1% and 0.5%

Figure 2: The average minimum coverage achieved by various algorithms over 100 real world data sets of $1M$ items each.



(a) Minimum coverage with varying budgets



(b) Coverage on the least covered features

Figure 4: Experimental results for the Independent data set

vary budgets on the x-axis, the coverage plotted on the y-axis is the minimum over all the 18 features (which is the objective function of the diversification algorithm).

Note that the budgets are too large for individual news feeds. However, as we noted earlier, our algorithm is useful for generating a single itemset of intermediate size that contains relevant items for all users, from which individual personalized news feeds can be generated using more resource-intensive techniques. Our experiments can be viewed as generating these intermediate itemsets.

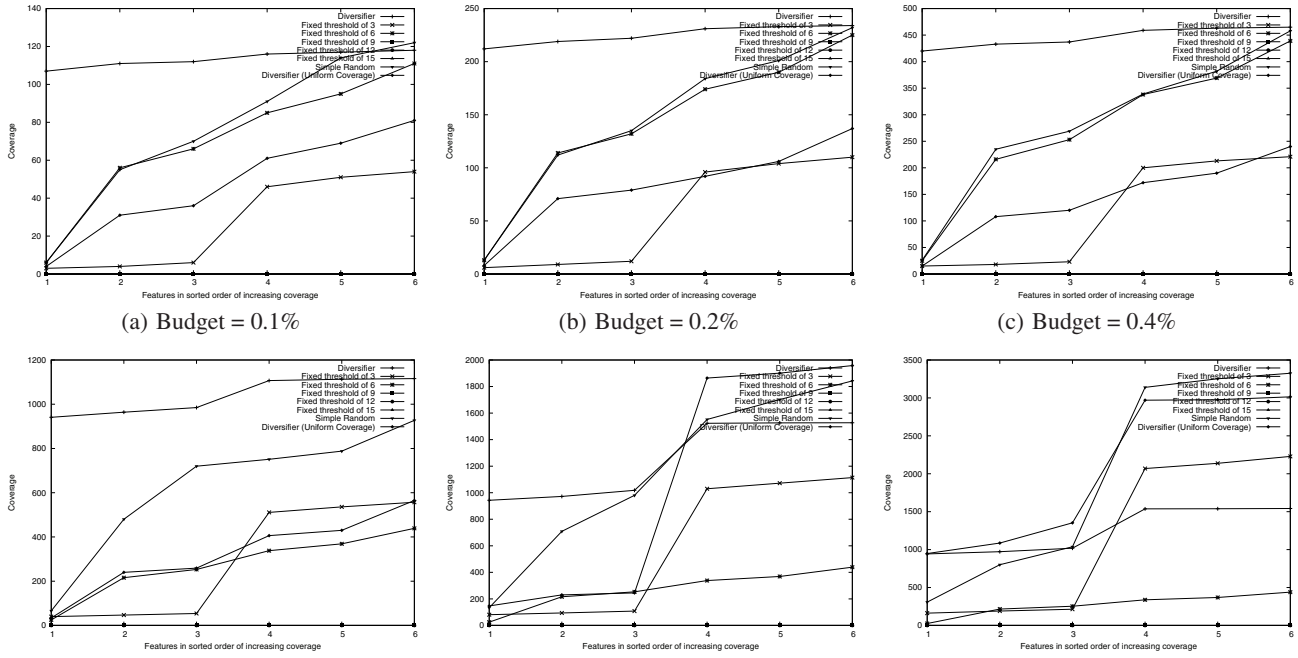
As stated in Theorem 6, our diversification algorithm achieves a $\frac{1}{2} - \delta$-approximation ratio as long as the expected optimum coverage is at least $\frac{24 \ln n}{\delta^2}$. In all our experiments, the minimum coverage on any feature (for both real and synthetic data sets) is at least 900, out of around $1M$ items. Further, since we are dealing with $n = 18$ features, $\ln n \approx 2.89$. Note that $900 \geq \frac{24 \ln n}{\delta^2}$ for values of $\delta \geq 0.28$. As we note in the experiments shortly, the performance of the diversification algorithm is significantly better than the theoretical guarantee.

5.4 Description of plots

Our first set of plots (Figure 2) compare the minimum coverage achieved by various algorithms, averaged over the 100 real data sets of $1M$ items each, with budgets varying from 0.1% to 8% of the input. Observe that the Diversifier significantly outperforms all the other algorithms for any budget between 0.1% and 4%. The minimum coverage achieved by the Diversifier levels off at a value between 900 and 1000 beyond a budget of 1% since there are features in our dataset that occur in fewer than 1000 items (and therefore the Diversifier has already achieved an almost optimal solution). Once the budget reaches 4%, the Diversifier (Uniform Coverage) also achieves optimal performance. All the other algorithms fair significantly worse, especially for smaller budgets. Notice that the other algorithms however require a budget of at least $40K$ to achieve any reasonable performance. In fact the minimum coverage achieved by the Diversifier increases linearly with the budget thereby confirming scalability of the algorithm.

For our next set of plots (Figure 3) we show the coverage achieved by the various algorithms on the least covered features. These experiments are performed on the real data set for six fixed budgets varying from 0.1% to 4%. Throughout these plots, we see a consistent trend of the Diversifier performing extremely well for the lesser covered features while the other algorithms are able to perform well only on the features that get higher coverage. In other words, these algorithms fail to perform well at the specific task of diversification which requires spreading out the coverage uniformly. Observe that in the plot for the comparatively high budget of 4%, the other algorithms also perform well for features that receive lesser coverage. This is because even the optimal solution can only attain a coverage of about as much obtained by these algorithms. However, for small budgets, the diversifier significantly outperforms all other algorithms.

Now, we describe the experimental results obtained for synthetically generated data sets. The plots for the experiments described above performed on the independent data set are given in Figure 4. In Figure 4 part (a), we see that the performance of the Diversifier rapidly improves with increasing budget while the other algorithms do not scale as well. This highlights that our algorithm is able to quickly adapt to varying coverages across features and assign importance to features that suffer from low coverage. On the



(a) Budget = 0.1% (b) Budget = 0.2% (c) Budget = 0.4%

(d) Budget = 1% (e) Budget = 2% (f) Budget = 4%

Figure 3: The coverage achieved on the least covered features on the real world data set by various algorithms.

other hand, the other algorithms continue to select items oblivious to previously selected items and therefore suffer from lower values of minimum coverage. In Figure 4 part (b), we again observe that the Diversifier performs well on the features with low coverage, i.e. it is able to balance out the coverage along different features and obtain a large minimum coverage, while other algorithms fail to do so.

Very similar trends are seen for the same experiments performed on the parity data set Figure 5 part (a). The diversifier does significantly better throughout, and the contrast is particularly noticeable as the budgets are increased. Observe that in the plot for sorted coverage in the parity data set (Figure 5 part (b)), the Diversifier does significantly better than all other algorithms. In fact, the Diversifier is very close to the optimum in this case as well, and therefore performs much better than guaranteed by Theorem 6. All these plots are horizontal because the data set contains only two kinds of items.

Finally, we show plots for the minimum coverage achieved for the different algorithms on the dependent and dependent-mixed data sets for varying budgets (Figure 6). In these plots, we notice that some of the other algorithms also perform well; in fact the randomized algorithm even outperforms the Diversifier for the dependent data set. Further, we observe that some of the fixed threshold algorithms perform well for large budgets. This is not surprising given that the features are very rigidly dependent on each other - therefore an algorithm that chooses the right threshold performs well on all features. Of course, note that we are comparing against an algorithm that somehow knows this threshold value, which is not feasible in practice. The takeaway here therefore is that the diversifier loses a bit in learning what threshold to use in an online fashion, but is then able to adapt and obtain a good coverage.

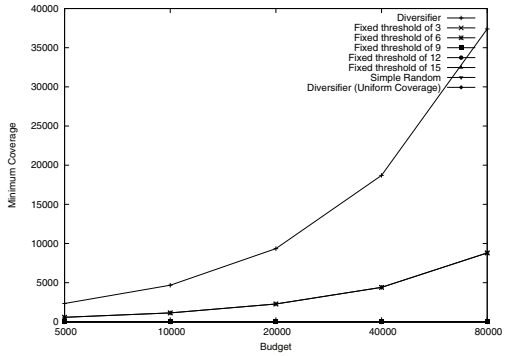
5.5 Summary

To summarize the experiments, we have seen that the diversifier performs extremely well on a wide range of parameters, and in particular, does significantly better than all the other algorithms we implemented on the real data set. The real data set comprised of several kinds of features with varying densities among items, hierarchical dependence, exclusive dependence etc., and yet the diversifier performs really well on the coverage for all budget ranges. The algorithm is fairly general, extremely simple to implement, low cost and efficient, provably approximate, and performs near-optimally even at large scales. Therefore these ideas and techniques may be useful in a wide range of other settings and applications.

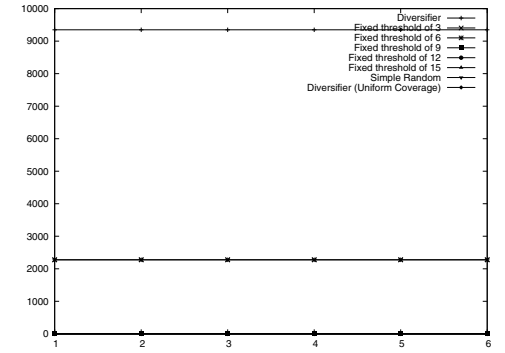
6. RELATED WORK

Diversification has been studied in a variety of different contexts so we only mention some of the references here. Among them search result diversification has arguably received the most attention (e.g. [11, 8, 16, 15, 14, 13, 12]). In particular, Gollapudi and Sharma [8] adopt an axiomatic approach and specify a set of rules that must be adhered to in any reasonable definition of diversification in the context of search results. A different approach is adopted by Slivkins *et al* [16] who employ learning theoretic techniques for diversification of rankings. For some recent work in diversification, the reader is referred to [15, 14].

Search diversification is different from our context in a couple of fundamental ways. First, the primary motivation for search result diversification is keyword disambiguation, and the intent is usually to provide the user with a set of result webpages such that includes at least one s/he is looking for. However, in our context, the goal is to ensure that the user is presented with a set of items that satisfies her cumulatively. Second, in the search architecture, one normally has access to all the (meta data related to) webpages stored on disk, and for efficiency reasons, one may need to perform online/streaming decisions on the documents, or quickly prune them to a candidate set; however, making irrevocable online decisions is not a necessity. This consideration makes our context a significantly harder one.

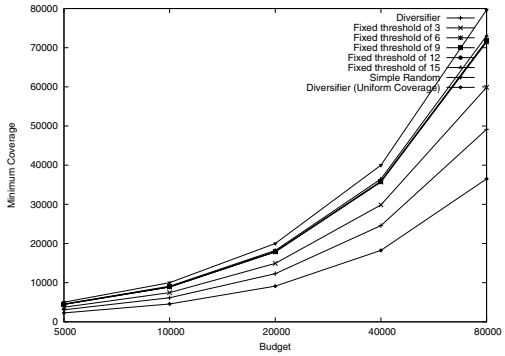


(a) Minimum coverage with varying budgets

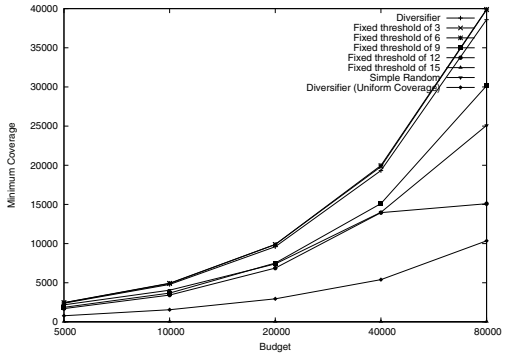


(b) Coverage on the least covered features

Figure 5: Experimental results for the Parity data set



(a) Dependent data set



(b) Dependent Mixed data set

Figure 6: The coverage achieved on the least covered features for various budgets by all algorithms on the Dependent and Dependent Mixed data sets.

The diversification problem has also been considered extensively in the context of recommender systems (see e.g. [19, 18, 9]). Several other works on diversification have been undertaken. For example, Agrawal *et al* [1] propose a maximum likelihood based diversification objective for finding relevant documents given the categorical information of queries and documents. Other papers on topical diversification and information retrieval include [21, 20]. Vee *et al* [17] consider diversification for online shopping, while Drosou and Pitoura [5, 6] have done work on diversity over continuous data such as in the context of publish/subscribe systems. Diversification for re-ranking documents and producing summaries has been considered by Carbonell and Goldstein [3]. The problem of selecting a diverse, representative set of posts in the blogosphere was considered by El-Arini *et al* [7]. Their approach is similar to ours in that they model the task of selecting a representative set as a covering problem. However, one key difference with our problem is that the entire set of blog posts is available to the algorithm before it makes any selection, i.e. their problem is offline. In addition, they consider a linear objective function, which is submodular unlike our objective function of minimum coverage. We also refer the reader to the references of these papers for further work in any specific context.

Significant algorithmic research has previously focused on *resource allocation* problems, where the goal is to allocate a set of resources in a manner that maximizes rewards. Our algorithm also falls in this broad framework, where we have B resources in the form of the items that we can opt to select, and the reward for a set of selected items is given by the minimum coverage on any feature. However, while we have a single-dimensional budget and multi-dimensional profit, recent work on this problem (see [4] and references therein) has largely focused on the scenario where the profit function is single-dimensional but the budgets are multi-dimensional. Another related set of resource allocation problems that have a multi-dimensional profit function are the *Santa Claus problems* (see [2] and subsequent work), where the goal is to allocate a set of items among a set of agents so as to maximize the reward of the least satisfied agent. However, this problem typically differs from our problem in two ways: first, each item can be allocated to only one agent and therefore earns rewards in only one dimension; and second, this problem is typically considered in the offline setting where all the items and their valuations by the agents are known to the algorithm.

7. DISCUSSIONS

In this paper, we considered the problem of selecting a diverse and representative subset from a large corpus of items. We modeled each item as being decorated by a set of features, and the goal was to ensure that the selected subset of items achieved large coverage on all features. We discussed various problem formulations representing this goal, and studied both the online and offline versions of the problem. Our key technical contribution was an easily implementable and scalable online algorithm for this problem. We analyzed this algorithm using theoretical techniques and showed that it achieves an approximation ratio of (roughly) 50%. We also performed wide-scale experiments on a variety of real-world and synthetically generated data sets and concluded that the algorithm performs even better than its theoretical guarantees in practice, and

also confirmed that it outperforms several natural and intuitive algorithms for this problem by a wide margin.

In fact, our algorithm adapts well to several real-world complications. For example, consider a data set that has *outliers*, i.e. some features that are very sparse. Our experimental results on the sorted coverage values show that in such cases, not only is the algorithm able to obtain near-optimal minimum coverage, but also does well on all features that have low coverage. This is crucial in situations where we do not want to maximize minimum coverage only, but also want to ensure that the algorithm performs well with respect to more relaxed notions of diversity, e.g. the coverage on the bottom 10% of features. This was in fact the case with our real data set, and here the diversifier algorithm performed particularly well. The algorithm is also extremely efficient and at each online stage, it only needs time proportional to the number of features to compute if a threshold is satisfied by the new arriving item; since it is very efficient, we do not plot graphs with running times, but the algorithm clearly scales to extremely large data sets as shown in our experiments.

In real-world applications, it is often the case that all items are not of identical *quality*, and while we wish to select a set of items that are diverse in terms of their feature coverage, we would also like to ensure that that these selected items have high overall quality. In such situations, we may interpret the quality of items as an additional feature on which we also want to meet a given quality target. Alternatively, we may opt to set a quality threshold on items and not select any item that does not meet this threshold irrespective of the coverage it achieves on the set of features. Other applications may have more complicated quality requirements, and an interesting direction of future research is to investigate the impact of item quality on the diversification problem.

In some other situations, items cover their constituent features to different degrees that may be represented by *coverage weights* (typically in the range $[0, 1]$). It can be shown that the same approximation guarantees hold from a theoretical perspective for this more general situation by using a slightly modified algorithm (for simplicity we omit these details). Even from an experimental standpoint, since the algorithm does well on features that are sparsely populated, it is expected to handle data sets with weighted coverages without a significant degradation in performance. However, further experimental work in this direction is desirable.

Yet another possibility is that features not only have a target coverage that we would like to achieve, but also an upper limit on coverage. Such two-sided errors can be handled by pretending to duplicate each feature by adding a compliment with the corresponding complemented coverage target.

Finally, in certain situations, the coverage function on a feature may not be additive, i.e. the overall coverage obtained by a set of items on a feature may be different from the sum of their individual coverage weights for the feature. An open problem is to design algorithms that are capable of handling such general coverage functions.

Acknowledgement. We would like to thank Vahab S. Mirrokni for useful discussions in the early stages of this work. We are also grateful to the anonymous reviewers, whose comments helped to improve the presentation of this paper.

8. REFERENCES

[1] Rakesh Agrawal, Sreenivas Gollapudi, Alan Halverson, and Samuel Ieong. Diversifying search results. In *WSDM*, pages 5–14, 2009.

[2] Nikhil Bansal and Maxim Sviridenko. The Santa Claus problem. In *STOC*, pages 31–40, 2006.

[3] Jaime G. Carbonell and Jade Goldstein. The use of mmr, diversity-based reranking for reordering documents and producing summaries. In *SIGIR*, pages 335–336, 1998.

[4] Nikhil R. Devanur, Kamal Jain, Balasubramanian Sivan, and Christopher A. Wilkens. Near optimal online algorithms and fast approximation algorithms for resource allocation problems. In *ACM Conference on Electronic Commerce*, pages 29–38, 2011.

[5] Marina Drosou and Evaggelia Pitoura. Diversity over continuous data. *IEEE Data Eng. Bull.*, 32(4):49–56, 2009.

[6] Marina Drosou and Evaggelia Pitoura. Search result diversification. *SIGMOD Record*, 39(1):41–47, 2010.

[7] Khalid El-Arini, Gaurav Veda, Dafna Shahaf, and Carlos Guestrin. Turning down the noise in the blogosphere. In *KDD*, pages 289–298, 2009.

[8] Sreenivas Gollapudi and Aneesh Sharma. An axiomatic approach for result diversification. In *WWW*, pages 381–390, 2009.

[9] Sean M. McNee, John Riedl, and Joseph A. Konstan. Being accurate is not enough: how accuracy metrics have hurt recommender systems. In *CHI Extended Abstracts*, pages 1097–1101, 2006.

[10] R. Motwani and P. Raghavan. *Randomized Algorithms*. Cambridge University Press, 1997.

[11] Filip Radlinski, Paul N. Bennett, Ben Carterette, and Thorsten Joachims. Redundancy, diversity and interdependent document relevance. *SIGIR Forum*, 43(2):46–52, 2009.

[12] Rodrygo L. T. Santos, Craig Macdonald, and Iadh Ounis. Exploiting query reformulations for web search result diversification. In *WWW*, pages 881–890, 2010.

[13] Rodrygo L. T. Santos, Craig Macdonald, and Iadh Ounis. Selectively diversifying web search results. In *CIKM*, pages 1179–1188, 2010.

[14] Rodrygo L. T. Santos, Craig Macdonald, and Iadh Ounis. How diverse are web search results? In *SIGIR*, pages 1187–1188, 2011.

[15] Rodrygo L. T. Santos, Craig Macdonald, and Iadh Ounis. Intent-aware search result diversification. In *SIGIR*, pages 595–604, 2011.

[16] Aleksandrs Slivkins, Filip Radlinski, and Sreenivas Gollapudi. Learning optimally diverse rankings over large document collections. In *ICML*, pages 983–990, 2010.

[17] Erik Vee, Utkarsh Srivastava, Jayavel Shanmugasundaram, Prashant Bhat, and Sihem Amer-Yahia. Efficient computation of diverse query results. In *ICDE*, 2008.

[18] Cong Yu, Laks V. S. Lakshmanan, and Sihem Amer-Yahia. It takes variety to make a world: diversification in recommender systems. In *EDBT*, pages 368–378, 2009.

[19] Cong Yu, Laks V. S. Lakshmanan, and Sihem Amer-Yahia. Recommendation diversification using explanations. In *ICDE*, pages 1299–1302, 2009.

[20] ChengXiang Zhai and John D. Lafferty. A risk minimization framework for information retrieval. *Inf. Process. Manage.*, 42(1):31–55, 2006.

[21] Cai-Nicolas Ziegler, Sean M. McNee, Joseph A. Konstan, and Georg Lausen. Improving recommendation lists through topic diversification. In *WWW*, pages 22–32, 2005.