

Selecting a Comprehensive Set of Reviews

Panayiotis Tsaparas
Search Labs
Microsoft Research
panats@microsoft.com

Alexandros Ntoulas*
Zynga
San Francisco, CA
antoulas@zynga.com

Evimaria Terzi†
Boston University
evimaria@cs.bu.edu

ABSTRACT

Online user reviews play a central role in the decision-making process of users for a variety of tasks, ranging from entertainment and shopping to medical services. As user-generated reviews proliferate, it becomes critical to have a mechanism for helping the users (information consumers) deal with the information overload, and presenting them with a small comprehensive set of reviews that satisfies their information need. This is particularly important for mobile phone users, who need to make decisions quickly, and have a device with limited screen real-estate for displaying the reviews. Previous approaches have addressed the problem by ranking reviews according to their (estimated) helpfulness. However, such approaches do not account for the fact that the top few high-quality reviews may be highly redundant, repeating the same information, or presenting the same positive (or negative) perspective. In this work, we focus on the problem of selecting a *comprehensive* set of few high-quality reviews that cover many different aspects of the reviewed item. We formulate the problem as a maximum coverage problem, and we present a generic formalism that can model the different variants of review-set selection. We describe algorithms for the different variants we consider, and, whenever possible, we provide approximation guarantees with respect to the optimal solution. We also perform an experimental evaluation on real data in order to understand the value of coverage for users.

Categories and Subject Descriptors

H.3.3 [Information Search and Retrieval]: [Selection Process];
H.2.8 [Database Management]: Database applications—*Data Mining*

General Terms

Algorithms, Experimentation, Theory

Keywords

Review Selection, Set Cover, Greedy Algorithms

*Work done while the author was at Microsoft Research.

†Work done while visiting Search Labs.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

KDD'11, August 21–24, 2011, San Diego, California, USA.

Copyright 2011 ACM 978-1-4503-0813-7/11/08 ...\$10.00.

1. INTRODUCTION

Online user reviews are an invaluable resource for making informed decisions for a variety of tasks such as purchasing products, booking flights and hotels, selecting restaurants, or picking movies to watch. Sites like `Yelp.com` and `Epinions.com` have created a viable business as review portals, while part of the popularity and success of `Amazon.com` and `TripAdvisor.com` is attributed to their extensive user reviews. The benefit of user reviews is that they are voluminous and comprehensive: multiple people, with different needs, different viewpoints, and different experiences review the same item, composing collectively a picture that is rich in detail and diverse in perspective.

At the same time, this information abundance can be overwhelming to the users. In `Amazon.com`, for popular products such as digital cameras, there are typically several hundreds of reviews, many of which are fraudulent, uninformative, or repetitive. Typical online users do not have the patience to go through all of them to sort out the ones with useful information content. To address this problem, most online portals allow the users to rate reviews according to their helpfulness, and there has been substantial amount of research in automatically estimating the quality of a review [19, 29, 9, 10, 32, 17, 15, 20].

Such approaches produce a score for each review, or an ordered list of reviews. However, they do not account for the redundancy in the content of the reviews, or the fact that some important aspects of the reviewed item may not be covered at all by the top results. For example, all top reviews may be highly informative about the long-range zoom of a new camera, but mention nothing about how easy it is to use, or to carry. Similarly, for a restaurant, the most helpful reviews may be detailed about the quality of the food, but mention nothing about the ambiance of the place, or how kid-friendly it is.

Furthermore, an ordered list of reviews does not necessarily represent all different viewpoints (e.g., positive vs. negative) of the item reviewed. There is experimental evidence [9] that users tend to consider helpful (and vote them as such) the reviews that agree with the average item rating. As a result the top reviews are more likely to represent a single viewpoint. Users usually need to explicitly filter on the review rating in order to get a diverse set of opinions.

Therefore, ordering reviews according to their user-defined, or algorithmically-estimated quality does not guarantee that there is a small set of reviews that covers the different aspects of an item with diverse viewpoints. The need for such compact and comprehensive information becomes critical in the case of mobile phone users. In mobile phones, screen real-estate and time resources are even more sparse, and users need helpful information to make quick decisions as they do not have the luxury to carefully go through multiple reviews. Instead, they want to quickly browse through a few reviews

and get a well rounded view of how good a product is, whether they are likely to enjoy a film, or whether the restaurant that is two blocks away has good vegetarian dishes. Therefore, the few reviews that they will read should have high information content and cover the aspects of an item with diverse opinions.

In this paper we consider the *review set selection* problem where given a set of reviews for a specific item, we want to select a comprehensive subset of small size. The notion of comprehensiveness is defined with respect to the attributes of the product and the viewpoints of the reviews. Given a review of a specific item, we assume that we have the following information: (a) the attributes of the item that are discussed in the review; (b) the quality of the review; (c) the viewpoint of the review (e.g., positive or negative). The selected subset should cover as many attributes of the item as possible, while containing reviews of high quality, which offer different viewpoints for the attributes of the product. We formalize this intuition as a *maximum coverage* problem, and we show how we can extend existing algorithms for maximizing coverage to address our requirements.

In this paper we make the following contributions:

- We formulate the review set selection problem as a coverage problem and we define a generic formalism that can be used to model the different variations of our problem. The GROUP-COVERAGE problems we define in Section 2 are novel, and have not been previously studied.
- We provide a theoretical analysis of the coverage problems we consider, and describe efficient algorithms for review selection. Whenever possible, we provide approximation guarantees for the proposed algorithms.
- We perform an experimental analysis of the algorithms on real data, and we study the value of coverage to users by conducting a user study on Amazon Mechanical Turk.

The rest of the paper is structured as follows. In Section 2, we define the coverage formalism, and model our selection problem. In Section 3, we provide a theoretical analysis of the different coverage variants we consider; we give algorithms for each one of these variants in Section 4. In Section 5, we present the experimental analysis. In Section 6, we present the related work, and we conclude in Section 7 with a discussion about possible extensions of our framework.

2. COVERAGE PROBLEM FORMULATION

We now show how to formulate the review set selection problem as a maximum coverage problem and we define different variants of the problem that we will consider. Consider a given item x , e.g., a product, that has a set of attributes $\mathcal{A} = \{a_1, a_2, \dots, a_m\}$. Let $\mathcal{R} = \{r_1, \dots, r_n\}$ be a set of reviews for item x . We assume that each review r is associated with a subset $\mathcal{A}_r \subseteq \mathcal{A}$ of attributes of x : these are the attributes that are discussed in review r for item x . We also say that review r covers attribute a if $a \in \mathcal{A}_r$. We use \mathcal{R}_a to denote the set of reviews in \mathcal{R} that cover attribute a .

Now, let $\mathcal{S} \subseteq \mathcal{R}$ denote a subset of reviews. We assume a *coverage scoring function* $f(\mathcal{S}, a)$ that assigns a score to attribute a given the subset \mathcal{S} . Intuitively, the scoring function measures the the benefit we obtain from covering attribute a with set \mathcal{S} . Let $\mathcal{A}_{\mathcal{S}}$ denote the union of attributes covered by the reviews in set \mathcal{S} , $\mathcal{A}_{\mathcal{S}} = \cup_{r \in \mathcal{S}} \mathcal{A}_r$. We define the function f such that $f(\mathcal{S}, a) = 0$ for all attributes $a \notin \mathcal{A}_{\mathcal{S}}$. Therefore, in order to define f , given a set \mathcal{S} , we only need to specify the value $f(\mathcal{S}, a)$ for the attributes $a \in \mathcal{A}_{\mathcal{S}}$.

We now define the following generic *coverage* problem, parametrized by the coverage scoring function f .

PROBLEM 1 (COVERAGE(f)). *Given a set of attributes \mathcal{A} and a set of reviews \mathcal{R} for an item x , an integer budget value k , and a coverage scoring function f , find a subset of reviews $\mathcal{S} \subseteq \mathcal{R}$ of size $|\mathcal{S}| = k$ that maximizes*

$$F(\mathcal{S}) = \sum_{a \in \mathcal{A}} f(\mathcal{S}, a)$$

We will refer to function $F(\mathcal{S})$ as the *cumulative coverage scoring function*, and it is defined with respect to the coverage scoring function f .

This broad definition allows us to define different coverage problems by varying the coverage scoring function f . In the simplest case, the scoring function f assigns the same score to all attributes covered by the set \mathcal{S} . We use f_u to denote this scoring function and we have that $f_u(\mathcal{S}, a) = 1$ for all $a \in \mathcal{A}_{\mathcal{S}}$. We use F_u to denote the corresponding cumulative scoring function, and we denote the COVERAGE(f_u) problem as UNIT-COVERAGE. The UNIT-COVERAGE problem asks for a set of k reviews that cover as many attributes as possible; it is also known in the literature as the MAX-COVERAGE problem [12].

Assume now that we are given a quality function $q : \mathcal{R} \rightarrow [0, 1]$ that maps a review r to a real number $q(r)$ that measures the *quality*, or *helpfulness*, of the review. This value may be present in our data (for example, via users voting for the quality of different reviews), or it may be inferred algorithmically. Given the quality values, we would like our selected subset to contain reviews that have high quality, while at the same time cover as many different attributes of a product as possible. The coverage scoring function should then give higher score to a pair (\mathcal{S}, a) if the attribute a is covered by a review of high quality. Let \mathcal{S}_a denote the set of reviews in \mathcal{S} that cover a , that is, $\mathcal{S}_a = \mathcal{S} \cap \mathcal{R}_a$. Given the quality function q , we define the coverage scoring function f_q , as follows:

$$f_q(\mathcal{S}, a) = \max_{r \in \mathcal{S}_a} q(r)$$

That is, the coverage score of attribute a is the highest review quality among the reviews that cover a . Note that the scoring function f_u (and the COVERAGE(f_u) problem) is a special case of the f_q (and COVERAGE(f_q)) when $q(r) = 1$ for all reviews. Similar to before, we use F_q to denote the cumulative scoring function with respect to f_q . We refer to the COVERAGE(f_q) problem as the QUALITY-COVERAGE problem.

Assume now that we can partition the reviews into g groups, $\mathcal{R} = \{\mathcal{R}^1, \dots, \mathcal{R}^g\}$. In our work, we assume that each group \mathcal{R}^i contains reviews of the same viewpoint. When $g = 2$ we have a positive and a negative viewpoint, while for $g > 2$ we have a finer granularity of viewpoints. Given this partition, we now require the selected subset to represent all different groups.

For a subset \mathcal{S} of reviews, let $\mathcal{S}^i = \mathcal{S} \cap \mathcal{R}^i$ denote the set of reviews in \mathcal{S} that belong to group \mathcal{R}^i . Let f denote a *base* scoring function that is applied to each group (e.g., this can be either f_u , or f_q). We define the group coverage scoring function as follows:

$$f_g(\mathcal{S}, a) = \min_{i=1 \dots g} f(\mathcal{S}^i, a) \quad (1)$$

By using the min operator in the definition of the f_g function in Equation 1, we guarantee that the score of an attribute that is not covered by all groups is zero. Therefore, in order to obtain benefit for a given attribute we need the attribute to be covered by all different groups. Again, we use F_g to denote the cumulative coverage scoring function, and we will refer to the COVERAGE(f_g) problem

as the GROUP-COVERAGE problem. To the best of our knowledge this is the first time that the GROUP-COVERAGE problem is defined. Maximum coverage with groups has been considered in [7] but in this case they require that only one review can be selected from each group.

We will define the group coverage function f_g using f_u or f_q as the base scoring functions. When necessary to discriminate between the two, we will use f_{gu} to denote the group coverage function defined with respect to f_u , and f_{gq} to denote the group coverage function defined with respect to f_q . We use F_{gu} and F_{gq} to denote the corresponding cumulative functions, and GROUP-UNIT-COVERAGE and GROUP-QUALITY-COVERAGE to denote the corresponding problems.

The COVERAGE(f_g) problem requires that the selected subset S covers each attribute a from all possible viewpoints. We can relax this requirement, by replacing the min operator in the f_g function, by a sum operator. We thus have the following scoring function:

$$f_s(S, a) = \sum_i f_q(S^i, a) \quad (2)$$

We use F_s to denote the corresponding cumulative scoring function, and we refer to the COVERAGE(f_s) as the SOFT-COVERAGE problem. Similar to before, we define scoring (cumulative) functions f_{su} (F_{su}) and f_{sq} (F_{sq}) depending on the choice of the basis function. We will refer to the corresponding problems as SOFT-UNIT-COVERAGE and SOFT-QUALITY-COVERAGE.

3. ANALYSIS OF COVERAGE

We will now analyze the complexity of the different variants of the COVERAGE problem.

For the simple unit coverage scoring function f_u , the UNIT-COVERAGE problem corresponds to the MAX-COVERAGE problem. In the MAX-COVERAGE problem, given a universe of elements $\mathcal{A} = \{a_1, \dots, a_m\}$, and a collection of sets $\mathcal{R} = \{r_1, \dots, r_n\}$, and an integer budget value k , we are looking for a set $S \subseteq \mathcal{R}$ of size k , such that the set of elements in \mathcal{A} that are covered by the sets in S is maximized. This is identical to the definition of the COVERAGE(f_u) problem where the elements are attributes, and the sets are reviews. The MAX-COVERAGE problem is known to be NP-hard [12]. Since functions f_q , f_g and f_s contain f_u as a special case, it follows that the QUALITY-COVERAGE, GROUP-COVERAGE and SOFT-COVERAGE problems are also NP-hard.

Since getting an optimal solution is hard, we turn to approximation algorithms. If $\text{OPT}(X)$ is the benefit of the optimal solution for a maximization problem on an input instance X , and $\text{ALG}(X)$ is the benefit of algorithm ALG on instance X , we say that ALG is an α -approximation algorithm, if $\frac{\text{OPT}(X)}{\text{ALG}(X)} \geq \alpha$ for all possible input instances X .

The MAX-COVERAGE (UNIT-COVERAGE) problem is known to have a simple $(1 - 1/e)$ -approximation algorithm [12]. The algorithm proceeds in k iterations, where in each iteration a new review is added to the result set. The algorithm chooses the next review to be added in a greedy fashion each time selecting the review that covers most of the attributes that have not already been covered. The proof for the approximation factor relies on the fact that the cumulative scoring function F_u that we want to maximize is *monotone* and *submodular*. Let $2^{\mathcal{R}}$ denote the powerset of the set \mathcal{R} . The function $F : 2^{\mathcal{R}} \rightarrow \mathbb{R}$ is monotone if for every $S, T \in 2^{\mathcal{R}}$, such that $S \subseteq T$, $F(S) \leq F(T)$. The function F is submodular, if it has the *diminishing returns* property: for every $S, T \in 2^{\mathcal{R}}$, such that $S \subseteq T$, and every $r \in \mathcal{R}$,

$$F(S \cup \{r\}) - F(S) \geq F(T \cup \{r\}) - F(T)$$

That is, the *incremental gain* of adding an element to a set decreases as the size of the set increases. When looking for a subset of reviews of size k that maximizes a submodular function F , the greedy algorithm that each time adds the review r that maximizes the incremental gain to the existing set is $(1 - 1/e)$ -approximate.

It is very easy to show that all cumulative functions are monotone. The function F_u is known to be submodular. We can prove the following lemma for function F_q .

LEMMA 1. *The function $F_q(S) = \sum_{a \in \mathcal{A}_S} f_q(S, a)$ is submodular.*

We omit the proof due to space constraints. The submodularity of F_q is also discussed in [4]. The submodularity of F_s follows directly from that of F_q . The review sets \mathcal{R}^i for the different groups are disjoint. Therefore, the gain in F_s of adding review $r \in \mathcal{R}^i$ to a multi-group set of reviews S is the same as the gain in F_q of adding review r to the set S^i . Since we know that F_q is submodular it follows that F_s is also submodular.

We can prove that the F_g function is *not* submodular.

LEMMA 2. *The function $F_g(S)$ is not submodular.*

PROOF. We consider the case where the f_g function is defined with respect to f_u , and we assume that there are just two groups $\{\mathcal{R}^1, \mathcal{R}^2\}$. For a given set of reviews S , we use \mathcal{A}_S to denote the set of attributes that are covered by at least one review in S , that is, $f_u(S, a) = 1$. Let $C_S \subseteq \mathcal{A}_S$ denote the set of attributes in \mathcal{A}_S for which $f_g(S, a) = 1$, that is, they are covered by a review from both groups, and let $U_S \subseteq \mathcal{A}_S$ denote the set of attributes in \mathcal{A}_S for which $f_g(S, a) = 0$, that is they are covered by reviews of only one of the two groups. We say that the attributes in C_S are *fully* covered, while the attributes in U_S are *partially* covered. Without loss of generality, we assume that all attributes in U_S are covered (only) by reviews in group \mathcal{R}^1 .

Consider now another set of reviews $\mathcal{T} \supseteq S$, such that $C_{\mathcal{T}} = C_S$, and $U_{\mathcal{T}} \supset U_S$, and assume that the attributes in $U_{\mathcal{T}} \setminus U_S$ are covered (only) by reviews in group \mathcal{R}^1 . Let $r \in \mathcal{R}^2$ be a review from group 2, such that $\mathcal{A}_r = U_{\mathcal{T}}$. We have $F_g(S \cup \{r\}) - F_g(S) = |U_S|$. On the other hand, $F_g(\mathcal{T} \cup \{r\}) - F_g(\mathcal{T}) = |U_{\mathcal{T}}| > |U_S|$. It follows that $F_g(S)$ is not submodular. \square

In fact, we can show that the GROUP-COVERAGE problem contains as a special case the DENSEST k -SUBGRAPH (DKS) problem for bipartite graphs. In the DKS problem, given a bipartite graph G we want to find a set of k vertices, such that the number of edges in the induced subgraph is maximized. The approximation-ratio of the DKS is unresolved. It is known that there is no Polynomial-Time Approximation Scheme (PTAS), and the best known approximation algorithm has approximation ratio $O(n^{1/4})$ [3]. It is not known if there is a $O(n^\epsilon)$ approximation algorithm for $\epsilon \geq 0$. A constant factor approximation for the COVERAGE(f_g) would imply a constant factor approximation for DKS.

To reduce GROUP-COVERAGE to the DKS problem, we consider again the case of two groups ($g = 2$), where all reviews have uniform quality. We create a bipartite graph $G = (V_1, V_2, E)$, where the left side V_1 contains one node for each review $r \in \mathcal{R}^1$ in group 1, while V_2 contains one node for each review $r \in \mathcal{R}^2$ in group 2. We create an edge (r_1, r_2) , if reviews r_1 and r_2 share a covered attribute, that is $\mathcal{A}_{r_1} \cap \mathcal{A}_{r_2} \neq \emptyset$. The edge $e = (r_1, r_2)$ is associated with the edge attribute-set $\mathcal{A}_e = \mathcal{A}_{r_1} \cap \mathcal{A}_{r_2}$. Then, the GROUP-COVERAGE becomes the problem of finding k vertices in G , such that the cardinality of the union of the edge attribute-sets in the induced subgraph is maximized. In the special case where each edge is associated with exactly one attribute, and all edge attribute-sets are disjoint, this is exactly the DKS problem on a bipartite graph.

4. ALGORITHMS FOR COVERAGE

We now present our algorithms for the different variants of the COVERAGE problem. For the coverage scoring functions f_u , f_q , and f_s , since the resulting cumulative functions F_u , F_q , and F_s are submodular, we use a greedy algorithm that is easy to implement, and it is theoretically proven to produce a solution that is a constant-factor approximation of the optimal.

Let $\Delta_S(r) = F(S \cup \{r\}) - F(S)$ denote the incremental gain in the cumulative score function F when adding review r to the set S . The algorithm proceeds in k iterations, incrementally building the review set, by adding one review at the time. Let S_i denote the set of reviews constructed in iteration i , where $S_0 = \emptyset$. At iteration i , given the set of reviews S_{i-1} , we compute for each review $r \in \mathcal{R} \setminus S_{i-1}$ the incremental gain $\Delta_{S_{i-1}}(r)$ and we select the one with the maximum value to generate S_i . The outline of the greedy algorithm is shown in Algorithm 1 and it is parametrized by the scoring function f (and the corresponding cumulative function F).

Algorithm 1 The GREEDY algorithm

Input: Set of reviews $\mathcal{R} = \{r_1, \dots, r_n\}$; Set of attributes $\mathcal{A} = \{a_1, \dots, a_m\}$; Integer budget value k ; Scoring function f .
Output: A set of reviews $S \subseteq \mathcal{R}$ of size k .
1: $S_0 = \emptyset$
2: **for all** $i = 1, \dots, k$ **do**
3: **for all** $r \in \mathcal{R} \setminus S_{i-1}$ **do**
4: Compute $\Delta_{S_{i-1}}(r)$
5: **end for**
6: $r_i = \arg \max_{r \in \mathcal{R} \setminus S_{i-1}} \Delta_{S_{i-1}}(r)$
7: $S_i = S_{i-1} \cup \{r_i\}$
8: **end for**
9: return S_k

Applying the GREEDY algorithm to the GROUP-COVERAGE problem is not straightforward. Due to the requirement that each attribute should be covered by at least g reviews (one from each group), adding a single review r to a review set S incurs no benefit, unless there are already $g - 1$ reviews in S from all other groups that cover the same attribute(s) as r . For an attribute a , we have that $f_g(S, a) > 0$ if and only if the set S contains a *tuple* $t = (r^1, \dots, r^g)$ of g reviews, such that review r^i belongs to group \mathcal{R}^i , and all reviews in t cover attribute a . In this case, we say that the tuple t *fully* covers attribute a . For the attributes in \mathcal{A}_t that are not fully covered, we say that they are *partially* covered by tuple t .

We propose a greedy algorithm for the GROUP-COVERAGE problem that selects tuples of reviews instead of individual reviews, thus guaranteeing that at each step it incurs non-zero gain, if this is possible. Let $\mathcal{T} = \mathcal{R}^1 \times \dots \times \mathcal{R}^g$ denote the set of all possible tuples. Given a set of reviews S , and a tuple $t \in \mathcal{T}$, let $\Delta_S(t)$ denote the incremental gain of adding the reviews in tuple t to the set S . Also let $C_S(t) = |t \setminus S|$ denote the number of reviews contained in t that are not in S ; $C_S(t)$ captures the *cost* of adding tuple t to the set S . The tuples with cost within the remaining review budget are *candidate* tuples. The algorithm proceeds greedily, at each step adding to the output review set S the candidate tuple t that maximizes the gain-to-cost ratio $\Delta_S(t)/C_S(t)$. We also experimented with maximizing just the gain $\Delta_S(t)$ of the newly introduced tuple, but this algorithm performed worse experimentally, so we omit any further discussion.

When adding a tuple in our set, we obtain an immediate gain, but we also create a potential for future gains, by introducing attributes that are only partially covered. We take into account the potential

of a tuple in order to break ties between tuples with equal gain. For example, if we have a two tuples t_1, t_2 which both have the same gain-to-cost ratio, but the set of attributes partially covered by reviews in t_1 is larger than that of t_2 , then we should prefer t_1 over t_2 since we expect future additions to give higher gains at a lower cost.

Formally, for a tuple t , let U_t denote the set of attributes that are partially covered by tuple t . We define the *potential* of t with respect to S as $P_S(t) = |U_t \setminus \mathcal{A}_S|$, that is, the number of attributes partially covered by t that are not already partially or fully covered by S . We use the potential to break the ties between tuples with the same gain-to-cost ratio. We observed experimentally that this modification can improve the performance of the algorithm significantly.

Algorithm 2 outlines the new greedy algorithm. In line 8 of the algorithm we find the *set* of tuples that have the maximum gain-to-cost ratio, and then in line 9, we select the one with the maximum potential. If there are still ties, we break them by selecting a tuple arbitrarily.

Algorithm 2 The t -GREEDY algorithm.

Input: Set of reviews $\mathcal{R} = \{r_1, \dots, r_n\}$ and groups $\{\mathcal{R}^1, \dots, \mathcal{R}^g\}$; Set of attributes $\mathcal{A} = \{a_1, \dots, a_m\}$; Integer budget value k ; Scoring function f_g .
Output: A set of reviews $S \subseteq \mathcal{R}$ of size k .
1: Compute $\mathcal{T} = \mathcal{R}^1 \times \dots \times \mathcal{R}^g$
2: $S = \emptyset$
3: **while** $|\mathcal{S}| < k$ **do**
4: $b = k - |\mathcal{S}|$
5: **for all** $t \in \mathcal{T}$ **do**
6: Compute $\Delta_S(t), C_S(t), P_S(t)$
7: **end for**
8: $T = \arg \max_{t: C_S(t) \leq b} \Delta_S(t)/C_S(t)$
9: $t = \arg \max_{t \in T} P_S(t)$
10: $S = S \cup t$
11: **end while**
12: return S

5. EXPERIMENTAL ANALYSIS

In this section we perform an experimental analysis of our algorithms. The goals of the analysis are two-fold: to quantitatively compare the different algorithms over different measures, and understand when each algorithm performs better; to qualitatively compare the algorithms by performing a user study.

5.1 Datasets

For our analysis we use real data which are publicly available from the Bing shopping portal. The Bing shopping portal aggregates reviews from multiple sites such as Amazon and CNET. We consider a collection of 4,362 products, over three categories: Digital Cameras, Cell Phones, and MP3 Players. This amounts to 129,783 reviews in total.

For each review, we have the content of the review, the rating of the review for the product, and the *helpfulness* votes for the review, from the users in the site. A helpfulness vote can be either positive or negative. We use the fraction of positive votes over the total number of votes as a measure of the *quality* of the review. We intentionally choose to make the quality of a review to be algorithm-independent, so as to understand the effect of our selection algorithms without having to account for the effects of quality estimation. In a real application scenario, where not all reviews

have sufficient number of helpfulness votes (or any at all), some algorithm for estimating the quality of the review [19, 29, 9, 10, 32, 17, 15, 20] can be utilized to obtain a quality value.

From the collection of products and reviews, we prune away products that have less than 20 reviews so as to make the selection process meaningful. We also prune reviews with less than 10 votes in total, since for these reviews we consider that we do not have sufficient evidence to determine the quality of the review.

For the attribute extraction we use the attributes extracted from the Bing search engine for the "Product Scorecard". There are 96 total possible attributes for category Digital Cameras, 84 attributes for category Cell Phones, and 68 attributes for category MP3 Players. The attributes include a broad range of characteristics of the products such as "ease of use", "battery life", "sound quality", "image quality", "screen" and more. To avoid spurious attribute references, we consider that a review covers an attribute if it is mentioned at least twice in the review, and we only keep attributes that are covered by at least two reviews of the product. On average each review covers 4 attributes, and there are 20.7 distinct attributes per product. We note that our selection algorithms are independent of the attribute extraction algorithm. Any off-the-shelf (such as [13, 14, 6]), or custom-built attribute extraction algorithm can be used.

We use the product ratings to define groups of reviews. We consider the case where we have two groups: positive reviews, and negative reviews. Ratings take discrete values between 1 and 5. Following the convention in the Amazon.com site, we consider a review to be positive if the rating is 4 or 5, and negative if it is 3 or less. To avoid the effect of outliers, we create a group only if it contains at least three reviews. Otherwise we assume a single group.

5.2 Algorithms

We consider six different algorithms one for each variation of the COVERAGE problem. The algorithms are all greedy, as described in Section 4. More specifically we have the following algorithms.

GREEDY-U, GREEDY-Q, GREEDY-SU, GREEDY-SQ: The greedy Algorithm 1 that optimizes the F_u , F_q , F_{su} and F_{sq} cumulative score functions respectively.

GREEDY-GU, GREEDY-GQ: The greedy Algorithm 2 that optimizes the F_{gu} and F_{gq} cumulative score functions respectively.

In addition to the greedy algorithms we also compare against the following baselines.

TOPQLTY: Sort the reviews according to their quality and select the top- k reviews. To aid the TOPQLTY algorithm in the comparisons with the greedy algorithms with respect to coverage, we break ties in quality using the number of attributes covered by the reviews.

TOPLEN: Sort the reviews according to length, and select the top- k reviews. This is meant to serve as a natural basic baseline. Longer reviews are also expected to be of higher quality and cover more attributes.

RANDOM: The RANDOM algorithm selects randomly k reviews. This is meant to serve as a sanity check in order to calibrate the results of the other algorithms. We perform 1000 runs for the RANDOM algorithm, and take the average performance of the algorithm.

For all the runs we set the value of k to be 5. We consider this to be a reasonable number of reviews that can give a thorough picture of a product. Furthermore, in the mobile applications we envision, where the screen and time resources are limited, we do not expect the user to be able to read more than five reviews.

5.3 Quantitative Evaluation

The goal of this section is to study how our algorithms perform with respect to coverage metrics, and compare them against the baseline algorithms, and each other. As coverage metrics, we consider all the different coverage scoring functions that we defined. For the following, we use UCOV to denote the *unit coverage* value of the F_u function; QCOV to denote the *quality coverage* value of the F_q function; GUCOV to denote the *group coverage* value of the F_{gu} function; GQCOV to denote the *group quality coverage* value of the F_{gq} function; SUCOV to denote the *soft coverage* value of the F_{su} function; and SQCOV to denote the *soft quality coverage* value of the F_{sq} function. For a given set we also compute the *average review quality* QLTY of the set. Note that for each metric there is a corresponding greedy algorithm that aims at maximizing this metric. We call this the *target* metric of the greedy algorithm.

In order to be able to aggregate or compare values across different products, we normalize the coverage values by the maximum possible coverage value that can be obtained if we include in our set *all* of the reviews (i.e., set $k = n$). For the QLTY metric the maximum is achieved by the TOPQLTY algorithm. Table 1 shows the average normalized measures for all algorithms, and the standard deviation. For the RANDOM algorithm, we compute for each product the average over the 1000 random samples, and then we compute the average of the averages, and the standard deviation over the averages.

The following high-level observations emerge from the analysis of Table 1. First, all algorithms perform on average better than the random baseline (with the exception of TOPQLTY on GUCOV), although there are cases where the values are within the standard deviation interval.

Second, each greedy algorithm achieves the best value on the target metric (the value in bold), although for some metrics (e.g., UCOV) there are other algorithms that are close. This indicates that the greedy heuristics, although not optimal, do a good job at optimizing the function at hand.

Third, the normalized coverage values are high. The greedy algorithms achieve at least 83% of the maximum possible for their target metric, and as high as 98% for the GREEDY-U on the unit coverage metric. This indicates that our premise that there is a small subset of reviews that covers the attributes of a product and the different viewpoints of the reviews is valid in practice, and thus it makes sense to look for such a set.

We investigate further the results in Table 1, and we perform some statistical tests to better quantify the difference between the different algorithms. For each algorithm, each metric, and each product we compute the *empirical p-value* of the output value against the RANDOM algorithm. Let COV_{ALG} denote the value of algorithm ALG for the coverage metric COV. Let COV_i denote the value of the i -th random trial of the RANDOM algorithm. We define the empirical p -value of the COV_{ALG} measurement as

$$p(\text{COV}_{\text{ALG}}) = \frac{1}{N_S} \sum_{i=1}^{N_S} \mathbb{I}(\text{COV}_i \geq \text{COV}_{\text{ALG}})$$

where N_S is the number of RANDOM samples ($N_S = 1000$) in our experiments, and \mathbb{I} is an indicator function that is 1 if the predicate is true and zero otherwise. The p -value is the fraction of random trials for which the RANDOM algorithm achieves a value on COV greater or equal than COV_{ALG} . We consider the value COV_{ALG} to be statistically significant if the p -value is less than 0.05, meaning that we can reject the *null hypothesis* that the value was generated by a random process with confidence greater than 95%.

	UCov	QCov	GUCov	GQCov	SUCov	SQCov	QLTY
GREEDY-U	0.98 (0.04)	0.90 (0.09)	0.27 (0.24)	0.26 (0.24)	0.73 (0.11)	0.70 (0.12)	0.83 (0.11)
GREEDY-Q	0.97 (0.05)	0.96 (0.06)	0.21 (0.26)	0.21 (0.27)	0.70 (0.12)	0.73 (0.12)	0.92 (0.07)
GREEDY-GU	0.72 (0.27)	0.66 (0.26)	0.84 (0.14)	0.77 (0.17)	0.62 (0.15)	0.55 (0.16)	0.80 (0.12)
GREEDY-GQ	0.71 (0.27)	0.70 (0.28)	0.82 (0.15)	0.83 (0.15)	0.61 (0.15)	0.58 (0.16)	0.86 (0.11)
GREEDY-SU	0.95 (0.07)	0.87 (0.11)	0.77 (0.17)	0.70 (0.18)	0.86 (0.10)	0.79 (0.13)	0.80 (0.11)
GREEDY-SQ	0.95 (0.07)	0.93 (0.08)	0.71 (0.20)	0.72 (0.19)	0.84 (0.10)	0.84 (0.11)	0.89 (0.08)
TOPQLTY	0.74 (0.19)	0.77 (0.17)	0.14 (0.23)	0.15 (0.25)	0.52 (0.18)	0.58 (0.18)	1.00 (0.00)
TOPLEN	0.88 (0.11)	0.84 (0.13)	0.31 (0.30)	0.31 (0.30)	0.68 (0.14)	0.68 (0.15)	0.85 (0.11)
RANDOM	0.61 (0.11)	0.54 (0.12)	0.16 (0.08)	0.14 (0.07)	0.43 (0.11)	0.40 (0.11)	0.77 (0.08)

Table 1: Mean and standard deviation of the performance measures for the different algorithms

We compute the p -values for all algorithms, all metrics, and all products. Then, for each algorithm and each metric, we compute the fraction of products for which we have a p -value greater than 0.05. This is the fraction of products for which there is a probability at least 5% that a random selection of k reviews can produce a metric value greater or equal to that of the algorithm. We call this the *null-hypothesis fraction* (or simply the fraction).

Table 2 shows the null-hypothesis fraction for all algorithm and metric pairs. We observe that all algorithms achieve fraction zero, or close to zero, for their target metric. Therefore, the performance of the algorithms cannot be viewed as a random artifact. The few products for which we have high p -value for the GREEDY-U and GREEDY-Q algorithms correspond to cases where the number of attributes to be covered is small, and they can be covered with one or two reviews.

We observe higher fraction values for the other algorithm-metric combinations. High null-hypothesis fraction values correspond also to low normalized coverage metric values in Table 1. These are the metrics for which the algorithm performs worse. We have the highest fraction (worst performance) values for the greedy coverage algorithms GREEDY-U and GREEDY-Q when evaluated against the group coverage metrics GUCOV and GQCov, and the group coverage algorithms GREEDY-GU and GREEDY-GQ when evaluated against the UCov and QCov metrics. In these cases, we have high p -values for more than 50% of the products, and for as many as 83.6% of the products for the case of the GREEDY-Q algorithm against the GUCOV metric. This is expected since the target metric of the algorithm, and the evaluation metric are qualitatively very different, and optimizing one does not offer any guarantees for the other. On the other hand, for metrics that are related, such as UCov and QCov the corresponding greedy algorithm performs well on both metrics. The soft coverage metrics (and the corresponding algorithms) sit somewhere in between these two extremes and as a result we obtain relatively good performance for these metrics for all greedy algorithms, while the corresponding greedy algorithms perform relatively well on all metrics.

To better understand the relationship between the different algorithms and metrics we perform pairwise comparisons between the different algorithms. What we want to understand is whether some of the algorithms we consider are redundant and their results could be obtained by some of the other variations. For example for the UCov metric, the performance of the GREEDY-Q is very similar to that of the GREEDY-U; is it the case that we can obtain similar coverage with the GREEDY-Q algorithm?

To measure the similarity between the different algorithms we compute the average pairwise intersection of the output reviews. The results are shown in Table 3. The resulting matrix is asymmetric since there are cases that one algorithm may output less than k

results. Entry (ALG_i, ALG_j) contains the average (over products) fraction of reviews in the returned results of algorithm ALG_i that appear in the results of algorithm ALG_j . The table indicates high overlap between pairs of algorithms that have similar underlying intuition such as the different quality versions of the greedy heuristics, or the soft and non-soft counterparts. The intersection can be as high as 81%. However, this similarity is not statistically significant. For a coverage metric COV, and the corresponding greedy algorithm ALG that optimizes this metric, let $\overrightarrow{COV_{ALG}}$ denote the vector of COV values of algorithm over all products. We performed a paired t -test between $\overrightarrow{COV_{ALG}}$ and $\overrightarrow{COV_{ALG'}}$ for every other algorithm ALG' to determine if the vectors come from a distribution with the same mean. The significance tests indicate that this is not the case with confidence 95%. Therefore, the similarity between the algorithms is not statistically significant.

Finally, it is interesting to understand the relationship between coverage and quality. From the TOPQLTY row of Tables 1 and 2 we see that high quality does not guarantee high coverage. The TOPQLTY algorithm achieves very low coverage values (and high null-hypothesis fraction values) for all metrics. For the GUCOV metric, the performance of TOPQLTY is actually worse than that of RANDOM. Surprisingly the TOPLEN algorithm performs better than TOPQLTY with respect to all coverage metrics. One possible explanation is that longer reviews are more likely to cover more attributes. On the flip side, the greedy coverage algorithms have high null-hypothesis fraction (low performance) values for the QLTy metric. The algorithms that perform best are GREEDY-Q, GREEDY-SQ and GREEDY-GQ that take quality into account in their optimization criterion. Out of these GREEDY-GQ performs the worst, probably due to the hard constraint it imposes on covering every attribute from both groups.

The disconnect between quality and coverage implies that in our set selection we need to balance between these two competing metrics. In the following section we study how the preferences of the users align between these two metrics.

5.4 Qualitative Analysis

In the previous section we investigated how the algorithms perform under different coverage measures, and how they compare to each other. We will now perform a user study to understand how useful the selected sets are to actual users. Our goal is to determine whether the set of reviews produced by a given algorithm for a given product would enable the users to make an informed buy/not-buy decision on the product.

To this end, we ran an experiment with a set of workers from Amazon’s Mechanical Turk. For our task we selected a set of 25 products from our dataset for which we can create a positive and negative group. This was necessary in order to have a fair com-

	UCov	QCov	GUCov	GQCov	SUCov	SQCov	QLTY
GREEDY-U	0.98%	3.43%	74.59%	70.49%	7.38%	9.02%	88.24%
GREEDY-Q	6.37%	0.49%	83.61%	77.87%	23.77%	11.48%	40.20%
GREEDY-GU	57.35%	56.86%	0.00%	0.82%	51.64%	61.48%	91.67%
GREEDY-GQ	61.27%	54.90%	0.00%	0.00%	54.10%	50.82%	60.78%
GREEDY-SU	13.24%	17.16%	1.64%	3.28%	0.00%	2.46%	89.71%
GREEDY-SQ	17.65%	3.43%	14.75%	9.84%	0.82%	0.00%	53.43%
TOPQLTY	83.33%	51.96%	90.16%	86.89%	80.33%	59.02%	1.47%
TOPLEN	48.53%	34.80%	68.85%	61.48%	42.62%	35.25%	67.65%

Table 2: Null hypothesis fraction for the different algorithms and different performance measures

	GREEDY-U	GREEDY-Q	GREEDY-GU	GREEDY-GQ	GREEDY-SU	GREEDY-SQ	TOPQLTY	TOPLEN
GREEDY-U	100%	67.78 %	65.29 %	54.13 %	82.99 %	66.04 %	31.20 %	50.90 %
GREEDY-Q	57.48 %	100%	44.90 %	64.04 %	53.28 %	80.96 %	51.22 %	49.21 %
GREEDY-GU	61.58 %	50.29 %	100%	76.67 %	80.92 %	66.95 %	28.01 %	46.31 %
GREEDY-GQ	44.18 %	63.12 %	69.97 %	100%	62.29 %	81.09 %	37.39 %	46.22 %
GREEDY-SU	75.77 %	57.18 %	79.00 %	66.95 %	100%	73.18 %	28.18 %	48.24 %
GREEDY-SQ	53.66 %	78.38 %	58.86 %	79.41 %	66.94 %	100%	43.94 %	49.78 %
TOPQLTY	23.82 %	47.16 %	23.53 %	35.10 %	24.51 %	42.25 %	100%	34.61 %
TOPLEN	39.80 %	46.08 %	39.41 %	44.31 %	42.25 %	48.63 %	34.61 %	100%

Table 3: Average result intersection between different algorithms

parison among all the algorithms. We created a HIT (Human Intelligence Task) for every algorithm and product combination. In each HIT we present the worker with the name of the product, a link to a web page describing the product, and the review set output by the algorithm. For each review we display the text of the review and the rating of the product. We then asked the workers to evaluate the set of reviews produced by each of the algorithms in terms of how informed they felt in making a buy/not-buy decision after reading the set of reviews. More specifically the users were asked to select one of the options of “perfectly informed”, “well informed”, “somewhat informed” or “not informed at all”. We assigned each one of these options a score from 0 (“not informed at all”) up to 3 (“perfectly informed”) and we requested 18 workers for each algorithm-product combination. For each algorithm, the review set consists of 5 reviews for each product ($k = 5$). To eliminate spam and minimize noise, out of these 18 workers we kept the 5 which spent the most time during their task (and thus we were relatively certain that they had done a thorough job in reading the reviews).

The first column of Table 4 shows the average user satisfaction score for each algorithm, computed over all workers and over all products. The higher the score (with a maximum of 3.00) the more informed the users felt after reading the reviews. Our first observation is that the performance of all algorithms is comparable. Surprisingly, the TOPQLTY does not perform as well as one would expect. In our experiment, selecting the top-5 reviews with the highest quality does not produce informative reviews. This could possibly be attributed to a perceived redundancy in the results of the TOPQLTY algorithm. The algorithm that performs best is the TOPLEN algorithm, indicating a user preference towards more lengthy reviews. With the exception of the GREEDY-GU and GREEDY-GQ algorithms, our greedy heuristics perform well; they outperform the TOPQLTY algorithm, and achieve performance close to that of the TOPLEN. This result indicates that coverage is important to the users, and it should be accounted for when selecting a small set of reviews to present. The “soft” versions of the algorithms perform slightly better (although the difference is rather

small) indicating that representing different viewpoints in the results is a desirable property. The GREEDY-GU and GREEDY-GQ algorithms performed the worst among our greedy algorithms. This is mostly due to the hard requirement of covering attributes from both viewpoints, which results in a more restrictive selection. As a result these algorithms perform worse compared to their “soft” counterparts, i.e., GREEDY-SU and GREEDY-SQ respectively.

To further study the tradeoff between coverage and quality we also performed an additional Mechanical Turk experiment, where we asked the users to compare our greedy algorithms against the TOPQLTY algorithm. In the task we showed the users two sets of reviews, and asked them to select the one that they felt gave them more information about the product. We assigned the task to 18 workers, out of which we selected again the 5 best as before, and we measured the average difference of workers that prefer the greedy algorithm over the TOPQLTY algorithm minus those that prefer TOPQLTY over the greedy. The results we obtain are shown in the second column of Table 4, and are consistent with the average user satisfaction. The greedy algorithms are preferred over TOPQLTY and TOPLEN wins the comparisons with the highest margin. In this experiment, the value of introducing the review quality in the greedy optimization stands out more clearly. These experiment confirm our finding that coverage and viewpoint diversity are important to the users, together with review quality.

6. RELATED WORK

The set selection problem we consider in this paper is closely related to the quality estimation, and ranking of reviews. Ranking can be viewed as another way to obtain a small set of reviews by selecting the top- k best reviews and we compare our algorithms against a ranking approach. There has been substantial amount of research in this area [19, 29, 9, 10, 32, 17, 15, 20]. In all of these works, the output is a score for each review, or an ordering of the reviews. Our work is different in that we are trying to select a *set* of reviews that collectively perform well, rather than score each individual review.

	User Satisfaction	Average Difference over TOPQLTY
GREEDY-U	2.10	1.02
GREEDY-Q	2.10	1.74
GREEDY-GU	1.98	0.20
GREEDY-GQ	1.94	0.46
GREEDY-SU	2.13	0.78
GREEDY-SQ	2.11	1.44
TOPLEN	2.26	2.84
TOPQLTY	1.98	0.00

Table 4: Average results in our user study.

There is also substantial amount of work on opinion summarization [14, 13, 31, 22, 21, 23]. The overall goal is to extract aspects (or features) of an product, and a short piece of text that summarizes the opinions on the different aspects. This is different from our approach where we want to find a subset of reviews that best captures the different aspects of a product. Our work is to a large extent complimentary to this work.

The most related work to ours is a recent work by Lappas and Gunopoulos [16] where they consider the problem of finding a small set of reviews that cover *all* product attributes, while preserving the distribution of positive and negative reviews. This is similar to our problem definition, However, our formulation is different from that in [16], since we look for a set of fixed size, and our goal is to cover attributes from both the positive and negative viewpoint, rather than preserve the viewpoint distribution. This difference leads to a different optimization problem, and as a result, to different algorithmic challenges.

Our problem is also related to *query result diversification* where the goal is to produce a ranking of the query results that cover as many *query intents* as possible. There is substantial amount of work in this area [1, 8, 11, 30, 2, 25, 28, 26, 27]. In this case, the goal is to return search results that are as orthogonal as possible and still relevant to the input query. This goal is usually formulated as an optimization problem that tries to maximize two competing measures: relevance and diversity. In our setting maximizing diversity is not an end in itself; we are content with non-orthogonal reviews as long as they collectively have high coverage of attributes. In the case of the GROUP-COVERAGE problem, we want reviews from different viewpoints to work together to cover the attributes of the item, which is substantially different from the traditional diversification setting.

At a high level the problem we consider is similar to work on *document summarization via sentence extraction* where the goal is to pick a set of sentences that summarize a given document. Despite this high-level similarity, there are fundamental differences in the problem definition. Traditional document summarization (e.g., MMR [5] and Mead [24]) focus on designing scoring functions for candidate sentences: every sentence is scored independently of other sentences and the output contains a collection of highly scored sentences. These scoring functions do not take into account the interaction between sentences. In our case, we select sets of reviews rather than scoring each review separately.

The combinatorial version of the document-summarization problem has been only considered lately by Liu et al. [18]. In this case, the goal is to pick a set of sentences that collectively cover as many document topics and, at the same time have the minimum possible topical overlap. This formulation is different from the one we consider here: we do want high-coverage reviews, but we do not

want them to be necessarily orthogonal to each other, and we also want to represent all possible viewpoints. As a result, the objective function and the corresponding optimization problem we consider are different from the one considered by Liu et al. [18]. Overall, although some special cases of the COVERAGE problem have been considered in the past for review-corpora management, our work is the first to provide and analyze the formalism in its full generality, including the GROUP-COVERAGE problem.

7. DISCUSSION AND CONCLUSIONS

In this paper we studied the problem of selecting a small subset of reviews from a large collection of reviews for a product, such that we cover the different attributes of the product with high quality content that represents different viewpoints. We provide a generic framework for formulating coverage problems that capture these requirements. We proposed algorithms for these problems, and studied their theoretical properties. We performed experiments, and showed that our algorithms achieve performance that is statistically significant. Our user study indicates that attribute coverage and viewpoint diversity are properties that appeal to users.

There are several interesting extensions of our work, and connections with different domains. First, in this paper we work with product reviews, but our generic framework can be applied to other domains as well. One such domain is news and updates. In this setting, we have a collection of articles about a specific event, or storyline. Each article covers specific aspects of the event, has some quality, and a specific viewpoint (e.g., conservative vs. progressive). We want to select a small subset of articles of high quality, that cover as many aspects of the event as possible and present all different viewpoints. This problem can be naturally modeled using the coverage formalism we introduced in this paper. This formulation can also be applied to the case that instead of articles we have tweets, or facebook updates about a specific event, person, or URL.

Another interesting extension of our work is the case where the attributes of the product that we want to cover are not statically predefined, but rather defined dynamically by the user. In this case, we are looking for the smallest set of reviews that covers all the specified attributes. We can easily modify our framework to handle this case, and the same greedy algorithms are still applicable to the minimization problem. For some of these algorithm we can again obtain approximation guarantees. Note that the user-defined attributes could be in the form of queries, rather than be selected from a fixed pool of attributes. It is not clear what results should be returned by a search engine when handling a *collection* of queries, rather than a single query. Our work provides a possible approach for addressing this question.

Finally, our formulation allows for additional complexity that we did not explore in this paper due to size limitations. First, it is easy to incorporate the importance of attributes in our optimization criteria, if such information is available. Second, in our formulation we assumed that each review belongs to a specific group, for example positive or negative. It is though possible that the same review covers some attributes positively, and some attributes negatively. Our framework can be easily extended to handle this case, and our greedy group algorithms naturally generalize to this case.

Acknowledgements

We would like to thank Kunal Talwar for helpful discussions on the complexity of GROUP-COVERAGE. We would also like to thank Pei Yue, Raghuram Rajagopal, Rama Shenai, and Stuart Marshal with their help in getting the review data.

8. REFERENCES

- [1] R. Agrawal, S. Gollapudi, A. Halverson, and S. Jeong. Diversifying search results. In *WSDM*, pages 5–14, 2009.
- [2] Y. Azar, I. Gamzu, and X. Yin. Multiple intents re-ranking. In *Proceedings of the 41st annual ACM symposium on Theory of computing*, STOC '09, 2009.
- [3] A. Bhaskara, M. Charikar, E. Chlamtac, U. Feige, and A. Vijayaraghavan. Detecting high log-densities: an $(1/4)$ approximation for densest -subgraph. In *STOC*, pages 201–210, 2010.
- [4] G. Calinescu, C. Chekuri, M. Pál, and J. Vondrák. Maximizing a submodular set function subject to a matroid constraint (extended abstract). In *IPCO*, pages 182–196, 2007.
- [5] J. G. Carbonell and J. Goldstein. The use of MMR, diversity-based reranking for reordering documents and producing summaries. In *SIGIR*, pages 335–336, 1998.
- [6] G. Carenini, R. T. Ng, and E. Zwart. Extracting knowledge from evaluative text. In *Proceedings of the 3rd international conference on Knowledge capture, K-CAP '05*, pages 11–18, New York, NY, USA, 2005. ACM.
- [7] C. Chekuri and A. Kumar. Maximum coverage problem with group budget constraints and applications. In *APPROX-RANDOM*, pages 72–83, 2004.
- [8] H. Chen and D. R. Karger. Less is more: probabilistic models for retrieving fewer relevant documents. In *SIGIR*, pages 429–436, 2006.
- [9] C. Danescu-Niculescu-Mizil, G. Kossinets, J. Kleinberg, and L. Lee. How opinions are received by online communities: a case study on amazon.com helpfulness votes. In *WWW '09*, pages 141–150, New York, NY, USA, 2009. ACM.
- [10] A. Ghose and P. G. Ipeirotis. Designing novel review ranking systems: predicting the usefulness and impact of reviews. In *ICEC '07*, pages 303–310, New York, NY, USA, 2007. ACM.
- [11] S. Gollapudi and A. Sharma. An axiomatic approach for result diversification. In *WWW*, pages 381–390, 2009.
- [12] D. Hochbaum, editor. *Approximation algorithms for NP-hard problems*. PWS Publishing Company, 1997.
- [13] M. Hu and B. Liu. Mining and summarizing customer reviews. In *KDD*, pages 168–177, 2004.
- [14] M. Hu and B. Liu. Mining opinion features in customer reviews. In *AAAI*, pages 755–760, 2004.
- [15] S.-M. Kim, P. Pantel, T. Chklovski, and M. Pennacchiotti. Automatically assessing review helpfulness. In *EMNLP*, pages 423–430, Sydney, Australia, July 2006.
- [16] T. Lappas and D. Gunopulos. Efficient confident search in large review corpora. In *ECML/PKDD (2)*, pages 195–210, 2010.
- [17] J. Liu, Y. Cao, C.-Y. Lin, Y. Huang, and M. Zhou. Low-quality product review detection in opinion summarization. In *EMNLP-CoNLL*, pages 334–342, 2007. Poster paper.
- [18] K. Liu, E. Terzi, and T. Grandison. Highlighting diverse concepts in documents. In *SDM*, pages 545–556, 2009.
- [19] Y. Liu, X. Huang, A. An, and X. Yu. Modeling and predicting the helpfulness of online reviews. In *ICDM*, pages 443–452, 2008.
- [20] Y. Lu, P. Tsaparas, A. Ntoulas, and L. Polanyi. Exploiting social context for review quality prediction. In *WWW*, 2010.
- [21] Y. Lu and C. Zhai. Opinion integration through semi-supervised topic modeling. In *WWW*, pages 121–130, 2008.
- [22] Y. Lu, C. Zhai, and N. Sundaresan. Rated aspect summarization of short comments. In *WWW*, pages 131–140, 2009.
- [23] A.-M. Popescu, B. Nguyen, and O. Etzioni. Opine: Extracting product features and opinions from reviews. In *HLT/EMNLP*, 2005.
- [24] D. Radev, T. Allison, S. Blair-Goldensohn, J. Blitzer, A. Çelebi, S. Dimitrov, E. Drabek, A. Hakim, W. Lam, D. Liu, J. Otterbacher, H. Qi, H. Saggion, S. Teufel, M. Topper, A. Winkel, and Z. Zhang. MEAD - a platform for multidocument multilingual text summarization. In *LREC 2004*, Lisbon, Portugal, May 2004.
- [25] F. Radlinski, P. N. Bennett, B. Carterette, and T. Joachims. Redundancy, diversity and interdependent document relevance. *SIGIR Forum*, 43(2):46–52, 2009.
- [26] F. Radlinski, R. Kleinberg, and T. Joachims. Learning diverse rankings with multi-armed bandits. In *ICML*, pages 784–791, 2008.
- [27] D. Rafiei, K. Bharat, and A. Shukla. Diversifying web search results. In *WWW*, 2010.
- [28] A. Slivkins, F. Radlinski, and S. Gollapudi. Learning optimally diverse rankings over large document collections. In *ICML*, 2010.
- [29] O. Tsur and A. Rappoport. Revrank: a fully unsupervised algorithm for selecting the most helpful book reviews. In *ICWSM*, 2009.
- [30] E. Vee, U. Srivastava, J. Shanmugasundaram, P. Bhat, and S. Amer-Yahia. Efficient computation of diverse query results. In *ICDE*, pages 228–236, 2008.
- [31] H. Wang, Y. Lu, and C. Zhai. Latent aspect rating analysis on review text data: a rating regression approach. In *KDD*, pages 783–792, 2010.
- [32] Z. Zhang and B. Varadarajan. Utility scoring of product reviews. In *CIKM '06*, pages 51–57, New York, NY, USA, 2006. ACM.