Effective Selection of a Compact and High-Quality Review Set with Information Preservation

JIAWEI CHEN, Shanghai University of Finance and Economics
HONGYAN LIU, Tsinghua University
YINGHUI (CATHERINE) YANG, University of California, Davis
JUN HE, Renmin University of China

15

Consumers increasingly make informed buying decisions based on reading online reviews for products and services. Due to the large volume of available online reviews, consumers hardly have the time and patience to read them all. This article aims to select a compact set of high-quality reviews that can cover a specific set of product features and related consumer sentiments. Selecting such a subset of reviews can significantly save the time spent on reading reviews while preserving the information needed. A unique review selection problem is defined and modeled as a bi-objective combinatorial optimization problem, which is then transformed into a minimum-cost set cover problem that is NP-complete. Several approximation algorithms are then designed, which can sustain performance guarantees in polynomial time. Our effective selection algorithms can also be upgraded to handle dynamic situations. Comprehensive experiments conducted on twelve real datasets demonstrate that the proposed algorithms significantly outperform benchmark methods by generating a more compact review set with much lower computational cost. The number of reviews selected is much smaller compared to the quantity of all available reviews, and the selection efficiency is deeply increased by accelerating strategies, making it very practical to adopt the methods in real-world online applications.

CCS Concepts: • **Information systems → Retrieval tasks and goals**; • **Applied computing → Multi-criterion optimization and decision-making**;

Additional Key Words and Phrases: Review selection, information preservation, approximation algorithms, dynamic updating

ACM Reference format:
Jiawei Chen, Hongyan Liu, Yinghui (Catherine) Yang, and Jun He. 2019. Effective Selection of a Compact and High-Quality Review Set with Information Preservation. *ACM Trans. Manage. Inf. Syst.* 10, 4, Article 15 (December 2019), 22 pages.
https://doi.org/10.1145/3369395

This work was supported in part by the National Natural Science Foundation of China (grants 71771131, 71490724, and U1711262) and the MOE Project of Key Research Institute of Humanities and Social Sciences at Universities (grant 17JJD630006).

Authors' addresses: J. Chen, School of Information Management and Engineering, Shanghai University of Finance and Economics, Shanghai, 200433, China; email: chenjiawei@mail.shufe.edu.cn; H. Liu (corresponding author), School of Economics and Management, Tsinghua University, Beijing, 100084, China; email: hyliu@tsinghua.edu.cn; Y. (Catherine) Yang, Graduate School of Management, University of California, Davis, One Shields Avenue, Davis, CA 95616; email: yiyang@ucdavis.edu; J. He, School of Information, Renmin University of China, Beijing, 100872, China; email: hejun@ruc.edu.cn.

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. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
© 2019 Association for Computing Machinery.
2158-656X/2019/12-ART15 $15.00
https://doi.org/10.1145/3369395

1 INTRODUCTION

Online reviews are exerting an increasingly important influence on the consumer decision-making process. A survey reported by eMarketer [2016] states that only 9.9% of their respondents do not check online reviews before making a purchase. However, consumers seldom have the time or patience to read a large number of available reviews. If consumers can first read a small number of selected reviews to get a comprehensive understanding of the product of interest, it is beneficial for both consumers and sellers. Even though the volume of reviews can be vast, the number of features discussed is often limited for most products and services. Furthermore, if the Web site has access to information about the consumers, such as products they browsed and product reviews they have written, the site can potentially learn the features consumers are most interested in. When presenting reviews, they can first present a selected set of reviews that covers the concerned features. Consumers can potentially amass almost all of the content they care most about by reading a very small subset of reviews. In this article, we define a review set selection problem with the following characteristics:

(1) *Coverage*: The review set selected needs to cover all features extracted from reviews or a subset of them tailored to individuals, as well as their sentiment polarities (i.e., whether a consumer thinks positively or negatively about a feature).
(2) *Compact*: The number of reviews in the selected review set should be as small as possible.
(3) *High quality*: The quality of the selected review set, which can usually be measured by its functionality, reliability, and usability [Knight and Burn 2005], should be as high as possible.
(4) *Dynamic*: As reviews accumulate over time, the selected review set should be updated efficiently to reflect the most up-to-date information.

There exist several streams of research on review selection. They each have different sets of objectives when selecting reviews and thus can be adopted in distinct application settings with different managerial priorities. For example, most existing research focuses on assessing review quality that can be used to rank and select reviews [Ghose and Ipeirotis 2011; Hong et al. 2017; Liu et al. 2008]. This approach works well when the ranking of individual reviews is of the most interest and can be applied to the setting where a user will read very few reviews. However, it is not suited for generating a small set of representative reviews to cover a given set of features. The first issue is information incompleteness. The selected reviews with the highest quality tend to be about popular product features. A significant number of less popular features that might be of importance to some users may be omitted from the selection, leading to sizable information loss. The second issue is information redundancy. On the one hand, selected reviews have the highest quality among all reviews; however, on the other hand, they may contain a great deal of repetitive information. More recently, several review selection algorithms have been proposed to select a predefined number (k) of reviews to preserve statistical properties [Lappas et al. 2012; Paul et al. 2017] or optimize a given metric [Ma et al. 2017; Tsaparas et al. 2011; Yu et al. 2013; Zhang et al. 2016]. These approaches should be suitable if the goal is just to select a very small number of reviews. If the purpose is to preserve information related to specific features, these are not the appropriate solutions. Given a small k, it is very difficult to guarantee that all features of interest and their sentiment polarities are preserved. When k is big enough, their objective does not allow reviews to be selected in a way to preserve information about specific features.

Compared to the existing literature, our research with a unique set of objectives provides solutions to various application settings where the managerial priority is to select a compact set of good-quality reviews to cover all of the given features and their sentiments. From the original

reviews, we extract all available features, which is a superset of the specific set of features for individual users who may consider different sets of features to be important. These features of interest for different individuals can be learned before applying our review selection method when information is available (e.g., through questionnaires or observed historical behaviors). The specific feature set to cover can be provided to our method as input. If no individual interest can be derived, the default is to cover the complete set of features extracted. We provide a flexible approach that can take in a predefined feature set and discover a subset of good reviews to cover all specific features along with their sentiment polarities.

For a seller or maker of a product, it is also essential to learn all of the different opinions buyers have about the product. Even though we used online reviews as our data for this research, our method can be easily extended to settings where information completeness is highly valued. For example, among all customer service calls received, managers will be interested in reading a small set of call transcripts that can cover all of the issues or a specific set of issues raised by customers. By satisfying the coverage and compactness requirements, our approach can circumvent both information incompleteness and redundancy. Since the number of reviews selected is not limited by a given number, our research should not be adopted in applications where the number of selected reviews has to be under a certain number. However, as we show in our experiments, the number of reviews we selected to cover all qualified features and their sentiments is quite small.

Another concept we use in this article is "opinion," which we will illustrate with the following review about a hotel, "The breakfast is fantastic, and the decoration is stylish. However, the location of this hotel is not that good." In this example, location=negative is an opinion, in which "location" is the feature and "negative" is the sentiment polarity. Based on most methods of performing sentiment analysis at the review level, this entire review has a positive sentiment polarity. If sentiment analysis is merely performed at the review level, information at the feature level will be overlooked. In the review set selection problem we define, complete coverage needs to be satisfied—that is, every single opinion discussed in the entire review set, which is related to the given features, is covered by the selected review set. For the preceding example review, our solution will extract three opinions: breakfast=positive, decoration=positive, and location=negative, each of which will be contained in at least one review in the review set selected if no individualized feature set is provided.

We model our review set selection problem as a bi-objective combinatorial optimization problem. The first objective is to select a review set as compact as possible, and the second objective is that the quality of the selected reviews should be as high as possible. In addition, this optimization problem has a constraint that the result review set should cover all opinions for the given features. This problem was first introduced in our preliminary study [Xu et al. 2014], and we will thoroughly discuss the differences from that work in Section 2.

To solve the bi-objective combinatorial optimization problem, we adopt the weighted sum scalarization method to transform the problem into a single objective problem that can be regarded as a minimum-cost set cover problem defined in Vazirani [2013]. Since this set cover problem is NP-complete, we develop approximation algorithms with performance guarantees that can efficiently produce excellent results for our review selection. A basic greedy algorithm is first introduced. On top of it, we subsequently present an enhanced algorithm based on the concept of opinion rareness to make the result set more compact. An accelerated strategy is further adopted for efficiency enhancement by reducing a significant number of dominated reviews. In addition to effective selection, we develop a method to dynamically update the representative review set as new reviews are added and old reviews are deleted or modified. To evaluate our methods, we conduct experiments on multiple real-world review datasets, and experimental results show that the effectiveness and efficiency of our methods far outweigh baseline algorithms. Our methods not

only perform well in selecting a compact and high-quality review set that covers a specific set of opinions but also efficiently update review selection in dynamic situations.

Our research makes several important contributions. First, we address a unique problem of selecting a compact, high-quality, and dynamically updated review set that also preserves the information content. This problem as a whole has not been addressed by prior research other than our preliminary study [Xu et al. 2014]. Important aspects of a practical review selection application are captured in the comprehensive problem defined in this article. A good solution to this problem will bring real benefits to both online sellers and consumers. Second, we provide powerful solutions to the problem. As shown in the experiments, the size of the review set selected is very small (mostly between ten and forty) compared to the size of the entire review set (from a few thousand to nearly a million), which makes it practical for companies to use our methods in real-world applications. On top of the high-quality compact review set selected, our methods are extremely fast compared to alternatives, making it more feasible for companies to implement them.

2 RELATED WORK

The earlier research in review selection focused mostly on assessing review quality and revealing factors affecting review quality [Ghose and Ipeirotis 2011; Hong et al. 2017; Liu et al. 2008]. Once the quality is assessed, reviews can naturally be ranked and selected. However, review selection based on this strategy has suffered the most from information redundancy and incompleteness. The top high-quality reviews may be highly redundant because they could represent similar viewpoints. In addition, the top-ranked reviews could omit information contained in the entire review set.

More recently, several review selection algorithms have been proposed to select a predefined number (k) of reviews to optimize a given metric, such as coverage [Tsaparas et al. 2011], closeness (to the original statistical distribution of features) [Lappas et al. 2012; Paul et al. 2017], consistency [Zhang et al. 2016], and diversity [Ma et al. 2017; Yu et al. 2013]. Although the works under this category are optimizing different objectives while selecting k reviews, they do share some common drawbacks due to restricting the number of selected reviews. If the purpose is to preserve information related to specific features, these are not appropriate solutions. Given a small k, it is very difficult to guarantee that all interested features and their sentiment polarities are preserved. When k is big enough, their objective does not allow reviews to be selected in a way to preserve information about specific features. Under the constraint of k, Tsaparas et al. [2011] select a set of high-quality reviews to cover as many features as possible from reviews partitioned by groups with different sentiment polarities. As one of the studies in top-k review selection, this research has the closest goal to ours and thus is used as a baseline for comparison in the experiments demonstrating that their solution is much more time consuming and their feature coverage is only about 80% of ours. In addition, they only consider sentiment polarities at the review level and build this into their algorithm so that it can hardly consider sentiment polarities at the feature level.

Several studies have modeled the review selection problem as a set cover problem, in which the selected reviews are required to cover different aspects of the reviews [Nguyen et al. 2015, 2017; Xu et al. 2014]. Nguyen et al. [2015] study a slightly different problem than ours. They consider a collection of reviews and micro-reviews about an item and select a small number of reviews that best cover the content of the micro-reviews. Their approach first matches review sentences to micro-reviews and then selects a small set of reviews that contain as many micro-reviews as possible with few sentences. Further, Nguyen et al. [2017] shift the focus from reviews for a single entity to reviews for multiple entities and address the problem of summarizing the short reviews of multiple entities in a collection. In our preliminary study [Xu et al. 2014], we first raised the research question of finding a small subset of high-quality reviews to cover all of the opinions.

However, the optimization process of the bi-objective function is not properly handled in our previous work. The solution provided in Xu et al. [2014] is not complete because it did not consider all of the Pareto optimal solutions for the bi-objective combinatorial optimization problem. In addition to having a more rigorous problem definition and optimization process, this article also provides improved algorithms, adopts different datasets, conducts additional comprehensive experiments, and offers more theoretical analysis. In addition, the earlier version did not include the dynamic algorithm.

Some studies have introduced clustering techniques [Aggarwal and Zhai 2012] into representative information extraction. Guo et al. [2017] propose a novel clustering approach to group blogs with similar contents into clusters and then select one representative blog from each cluster. This kind of study focuses more on designing effective clustering methods. In addition, there is a substantial amount of work on text summarization [Gambhir and Gupta 2017] and opinion summarization [Condori and Pardo 2017]. The goal is to extract aspects from reviews and generate short pieces of texts (words, phrases, or sentences) that summarize opinions on different aspects. A collection of extracted short sentences that can preserve the most accurate thoughts of the person who wrote the review is provided to readers. This is a different problem than ours where we aim to find a subset of reviews that covers a given set of opinions.

3 REVIEW SELECTION PROBLEM

In this section, we define the problem of selecting a compact and high-quality set of reviews that can cover a given set of opinions. Without loss of generality, we present the review selection for covering all opinions contained in the complete set of reviews. This selection problem can be easily adapted to covering an individualized set of opinions.

3.1 Preliminaries

Let $R = \{r_i\}$, $i = 1, \ldots, n$, denote the set of reviews for a given product. All qualified features discussed in R are $F = \{f_l\}$, $l = 1, \ldots, m$. For each feature, we consider two sentiment polarities, positive and negative. Thus, there are at most $2m$ opinions collectively. Let $O = \{o_j\}$, $j = 1, \ldots, k$, where $k = 2m$, be the set of opinions for F. It can be easily extended to more sentiments. Any existing methods can be used to extract opinions from reviews, and we simply take the obtained opinions as inputs to our method. In the review set R, each review r_i has a quality score denoted by q_i. According to Knight and Burn [2005], the quality of a review can be assessed along multiple dimensions, such as functionality, reliability, and usability. Details about the methods adopted for feature extraction, sentiment analysis, and quality estimation are provided in Section 6.

Given the review set R and the opinion set O, we use matrix $\mathbf{A} = (a_{ij})_{n \times k}$ to record the opinions contained in each review. If review r_i covers opinion o_j, $a_{ij} = 1$, otherwise $a_{ij} = 0$. We also use x_i to indicate whether review r_i is included in the selected review set. If review r_i is selected, then $x_i = 1$, otherwise $x_i = 0$. Therefore, if review r_i is selected into the result set and meanwhile this review covers opinion o_j, then $a_{ij}x_i = 1$, otherwise $a_{ij}x_i = 0$.

3.2 Problem Definition

Due to the large number of online reviews, users seldom have time and energy to go through all of the reviews. It is beneficial for users to minimize the time needed to get a comprehensive understanding of the product of interest. Therefore, the first objective of our review selection problem is to find a review set as compact as possible (i.e., $\min \sum_{i=1}^{n} x_i$). Because users only get to read a subset of all of the reviews, the quality of the selected reviews should be as high as possible (i.e., $\max \sum_{i=1}^{n} q_i x_i$), which is our second objective. Note that the second objective is to maximize the total quality of all selected reviews instead of the average quality. Using the total quality as the

second objective will enable us to address the optimization problem properly instead of directly maximizing average quality, which has the first objective in the denominator. In addition, for each opinion o_j, we require that at least one selected review in the result set covers the opinion—for instance, $\sum_{i=1}^{n} a_{ij}x_i \geq 1$, for every j from 1 to k.

Given these two objectives, we can define the review selection problem as a bi-objective combinatorial optimization problem. Based on the objectives and constraints described earlier, the representative set selection problem (RSSP) can be formally defined. Note that the selected review set is used to represent the entire set of available reviews, so we call it the representative review set.

Definition 1 (Representative Set Selection Problem). The representative set selection problem is defined as the following bi-objective optimization problem:

$$(P): \quad \min \sum_{i=1}^{n} x_i;\ \max \sum_{i=1}^{n} q_i x_i$$
$$\text{subject to } \sum_{i=1}^{n} a_{ij}x_i \geq 1, j = 1, \ldots, k; x_i \text{ is binary}, i = 1, \ldots, n. \tag{1}$$

A feasible solution is one that satisfies all constraints in an optimization problem. In Problem P, a feasible solution $\mathbf{X} = (x_1, \ldots, x_n)$ represents the selection of reviews that can cover a given set of opinions. Because we require complete coverage, only feasible solutions to Problem P will be considered. In a multi-objective optimization problem, there seldom exists a feasible solution that optimizes all objectives simultaneously. This is the case for our problem as well. Among all of the feasible solutions, the most compact one may not be the one with the highest aggregated quality score and vice versa. When solving a multi-objective optimization problem, attention is often paid to Pareto optimal solutions. In our bi-objective problem, the Pareto optimal solutions are those that cannot be improved in one objective without degrading the other objective. In other words, for any Pareto optimal solution, there does not exist another feasible solution outperforming it in both objectives. Thus, our focus is to find all Pareto optimal solutions for Problem P defined in Equation (1). In addition, since the constraint to cover all opinions must be satisfied, we are interested only in the Pareto optimal solutions that are also feasible.

3.3 Problem Transformation

Many existing methods for solving a bi-objective optimization problem essentially combine the two objectives into one single objective. The most popular method is the weighted sum scalarization method [Ehrgott 2006, chap. 8.3], which we adopt in this work to transform two objectives into a single one using a positive scalarization weight, λ. By varying λ, all Pareto optimal solutions can be identified. We provide more discussions on the selection of λ in the experimental section. Our bi-objective optimization problem is transformed into the following single objective problem:

$$(P_\lambda): \quad \min\ \lambda \sum_{i=1}^{n} x_i + \sum_{i=1}^{n} (-q_i)x_i$$
$$\text{subject to } \sum_{i=1}^{n} a_{ij}x_i \geq 1, j = 1, \ldots, k; x_i \text{ is binary}, i = 1, \ldots, n. \tag{2}$$

Problem P_λ can be regarded as a minimum-cost set cover problem defined in Vazirani [2013]. Given a set U, a collection V of the subsets of U such that $\bigcup_{v \in V} v = U$, and a collection C of the corresponding costs for the elements in V, the task of a minimum-cost set cover problem is to find a subset of V, V_{sub}, to cover all elements in U and minimize the aggregated cost. In our problem

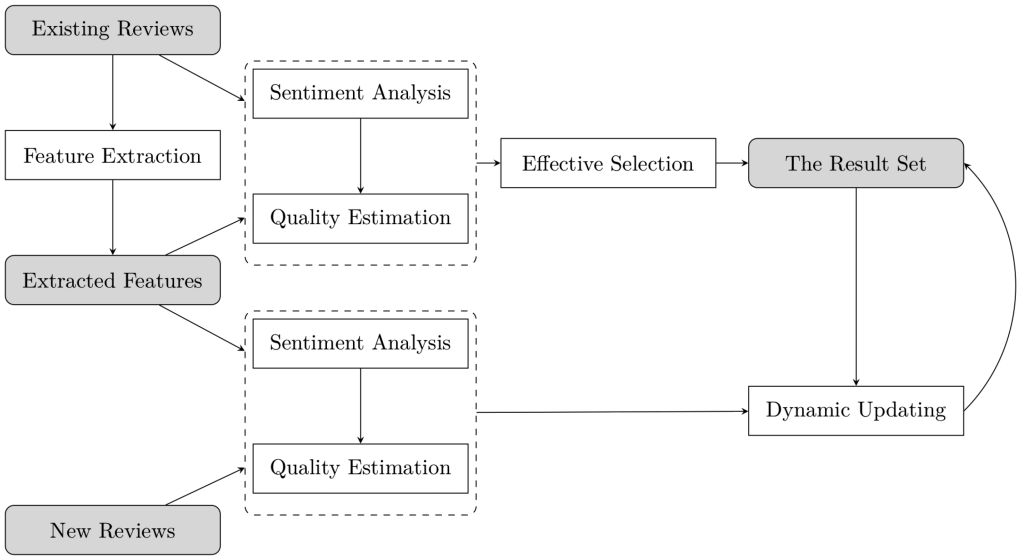

Fig. 1. An overview of the proposed framework for review set selection and updating.

setting, the set of opinions O corresponds to U, and the review set R corresponds to V. Our task is to minimize the aggregated cost value of selected reviews, $\sum_{i=1}^{n}(\lambda - q_i)x_i$, where $\lambda \geq 0$. Satisfying the constraint in Problem P_λ translates to having V_{sub} covering all elements in U.

A Pareto optimal solution of problem P is called a *supported Pareto optimal solution* if it can be recovered by solving Problem P_λ with some λ [Ehrgott 2006, p. 203]. If the decision variables x_i are continuous, then all Pareto optimal solutions of Problem P can be obtained by varying λ in Problem P_λ, because the two objectives are convex functions and the constraint is linear. However, because the decision variables in Problem P are binary, the convexity of the Pareto frontier is destroyed and there exist non-supported Pareto optimal solutions [Ehrgott and Gandibleux 2000]. This means that all Pareto optimal solutions for Problem P_λ are only a subset of all Pareto optimal solutions for Problem P. However, since both Problem P and Problem P_λ are NP-hard, our focus is on finding good solutions using approximation algorithms. Therefore, focusing on solving Problem P_λ using approximation algorithms should lead us to a reasonably good solution.

Further, we denote the cost of each review as $c(r_i) = \lambda - q_i$, and the RSSP can be formally transformed into the minimum-cost review set cover Problem P_c defined in Equation (3):

$$(P_c): \quad \min \sum_{i=1}^{n} c(r_i)x_i$$
$$\text{subject to} \sum_{i=1}^{n} a_{ij}x_i \geq 1, j = 1, \ldots, k; x_i \text{ is binary}, i = 1, \ldots, n. \tag{3}$$

3.4 Framework Overview

To solve the defined RSSP, we propose a comprehensive framework for review set selection and updating. An overview of the proposed framework is presented in Figure 1. Here, extracted features, learned sentiments, and quality values all serve as inputs to our proposed algorithms, which we will discuss in more detail in the experiments. In the subsequent two sections, we present our proposed algorithms for effective selection and dynamic updating.

4 EFFECTIVE SELECTION

The minimum-cost review set cover problem we focus on is a classic NP-complete problem [Karp 1972] that often is addressed by approximation algorithms. Thus, we develop approximation algorithms and introduce enhanced strategies that can efficiently achieve an effective selection.

4.1 The Basic Greedy Algorithm

Among approximation algorithms, greedy algorithms are classic methods for solving the minimum-cost review set cover problem [Johnson 1973]. We first present a basic greedy algorithm (Alg1_Greedy) for solving Problem P_c, and then in subsequent sections, we build other algorithms on top of this basic greedy algorithm to improve result quality and efficiency.

For each review $r_i \in R$, $o(r_i)$ denotes the set of opinions covered by r_i. For a review subset $S \subseteq R$, all opinions covered by the reviews in S are denoted by $o(S) = \bigcup_{r_i \in S} o(r_i)$. Intuitively, after adding a review r_i into S, the set of newly covered opinions is $\Delta o(r_i, S) = o(S \cup \{r_i\}) \setminus o(S)$. We use $|\cdot|$ to represent the cardinality of an opinion set. Thus, after adding a review r_i into S, the number of distinct opinions covered in S will increase by $\Delta|o(r_i, S)| = |o(S \cup \{r_i\})| - |o(S)| = |\Delta o(r_i, S)|$. For a newly added review r_i, the corresponding cost is $c(r_i)$. We call $c(r_i)/\Delta|o(r_i, S)|$ the *opinion cost*, which measures the average cost per newly covered opinion. In Alg1_Greedy, we select the review with the minimum opinion cost at every iteration.

ALGORITHM 1: The Basic Greedy Algorithm (Alg1_Greedy)

Input: The review set $R = \{r_1, r_2, \ldots, r_n\}$, and its opinion set $O = \{o_1, o_2, \ldots, o_k\}$.
Output: A subset of the review set $S \subseteq R$.

```
S = ∅;
while |o(S)| < |O| do
    r = arg min_{r_i ∈ (R\S)} c(r_i) / Δ|o(r_i, S)|;
    S = S ∪ {r};
end
return S;
```

According to the performance guarantee of the greedy algorithm for the minimum-cost set cover problem in Du et al. [2011, chap. 2.4], Alg1_Greedy can achieve a polynomial-time $(1 + \ln \sigma)$ approximation, where σ is the maximum number of opinions each review in R can cover, $\sigma = \max_{r \in R} |o(r)|$. It means that the total cost of the reviews selected (i.e., the objective function) by Alg1_Greedy is no more than $(1 + \ln \sigma)$ times the total cost of the optimal result set. The detailed proof is provided in Online Appendix A.

4.2 The Enhanced Algorithm with Opinion Rareness

In Alg1_Greedy, we treat all distinct opinions equal when calculating the opinion cost. Through experiments, we discover that opinions appearing in a large number of reviews are more likely to be covered in early iterations. Toward the end of the review selection process, we have to select additional reviews just to cover opinions that appear only in a limited number of reviews. At the same time, these additional reviews with uncommon opinions also contain common opinions that are covered in reviews selected earlier in the process. This creates redundancy and increases the size of the result review set. If we select reviews containing uncommon opinions earlier in the process, this selection heuristic may lower the number of reviews in the result set.

Table 1. Illustrative Example for Uncommon Opinions

	o_1	o_2	o_3	o_4	o_5	o_6	o_7	o_8	$\|o(r_i)\|$
r_1	√	√		√	√		√		5
r_2	√		√		√		√		4
r_3		√		√		√		√	4
r_4	√					√			2
r_5		√						√	2
N_j	3	3	**1**	2	2	2	2	2	

For example, we suppose that $R = \{r_1, r_2, r_3, r_4, r_5\}$. Table 1 shows the opinions contained in each review, and N_j denotes the number of reviews containing opinion o_j.

In this example, we assume that each review has the same cost value. According to Alg1_Greedy, r_1 will be the first selected review because it has the greatest number of opinions. Next, r_3 will be selected adding two opinions. Last, r_2 is added because it is the only review covering o_3. Note that r_2 also covers three other more common opinions, creating unnecessary redundancy. The final result set based on Alg1_Greedy is $\{r_1, r_3, r_2\}$. However, if we consider the uncommonness of the opinions (called *opinion rareness*) and select r_2 first because of opinion o_3, the final result set becomes $\{r_2, r_3\}$, which is smaller than $\{r_1, r_3, r_2\}$. This example illustrates that using opinion rareness could potentially help reduce the size of the result set. It does not mean that rare opinions would dominate in the result review set. This strategy primarily changes the order of how reviews are selected, because reviews with rare opinions will need to be included in any way to cover all opinions. The strategy of selecting reviews containing rare opinions earlier does not systematically create the problem of under-representing popular opinions, because reviews containing a rare opinion could also very likely contain many other more popular opinions.

Motivated by a more compact review subset, we introduce a measure to quantify the rareness of an opinion. For each opinion o_j, we define its rareness w_j, and $W = \{w_1, \ldots, w_k\}$. Here, we define $w_j = 1/N_j > 0$, where N_j is the number of reviews that contain opinion o_j in the entire review set. For each review r_i, the rareness $w(r_i)$ is the aggregated rareness of all distinct opinions covered in this review (i.e., $\sum_{o_j \in o(r_i)} w_j$). Similarly, the rareness of a review subset $w(S)$ is the aggregated rareness of all distinct opinions covered in S (i.e., $w(S) = \sum_{o_j \in o(S)} w_j$). Moreover, after adding a review to S, its aggregated rareness value increases by $\Delta w(r_i, S) = w(S \cup \{r_i\}) - w(S) = \sum_{o_j \in \Delta o(r_i, S)} w_j$. In Alg2_Rare, we update the opinion cost with $c(r_i)/\Delta w(r_i, S)$.

ALGORITHM 2: The Enhanced Algorithm with Opinion Rareness (Alg2_Rare)

Input: The review set $R = \{r_1, r_2, \ldots, r_n\}$, and its opinion set $O = \{o_1, o_2, \ldots, o_k\}$.
Output: A subset of the review set $S \subseteq R$.

```
S = ∅;
while |o(S)| < |O| do
    r = arg min_{r_i ∈ (R\S)} c(r_i)/Δw(r_i, S);
    S = S ∪ {r};
end
return S;
```

Similar to Alg1_Greedy, Alg2_Rare can achieve a polynomial-time $(1 + \ln \sigma)$ approximation, but σ is changed to $\max_{r_i \in R} w(r_i)/\min_{w_j \in W} w_j$. Due to space constraints, we theoretically analyze the performance guarantee property of Alg2_Rare in Online Appendix B.

Table 2. Example to Illustrate the Bitmap-Based Method for Skyline Detection

(a) The review set

	opinions	q
r_1	o_1, o_2	0.9
r_2	o_1, o_3	0.5
r_3	o_2	0.9
r_4	o_3	0.7
r_5	o_3	0.7

⇒

(b) The bitmap

	o_1	o_2	o_3
r_1	1	1	0
r_2	1	0	1
r_3	0	1	0
r_4	0	0	1
r_5	0	0	1

⇒

(c) After sorting

	o_1	o_2	o_3	$\lvert o(\cdot)\rvert$	q
r_1	1	1	0	2	0.9
r_3	0	1	0	1	0.9
r_5	0	0	1	1	0.7
r_4	0	0	1	1	0.7
r_2	1	0	1	2	0.5

⇒

(d) The skyline

	o_1	o_2	o_3
r_1	1	1	0
r_5	0	0	1
r_2	1	0	1

4.3 The Accelerated Algorithm with Skyline

In Alg2_Rare, the entire set of reviews aside from the ones already selected will be traversed at every iteration, which is time consuming, especially in large-scale datasets. In fact, not every review needs to be considered at each iteration. Only the ones that are not dominated by any unselected review could be the winning review. Generally, a review is said to be a dominated review if there exists another review better than this one in at least one dimension and as good as this one in all other dimensions. We can use this concept to reduce the number of candidate reviews we traverse in each iteration. In this way, we can speed up the selection process and keep the result set unchanged. Next, we formally define a dominated review in our problem.

Definition 2 (Dominated Review). Given a review set R', a review $r_i \in R'$ is a dominated review if there exists a review $r_j \in (R' \setminus \{r_i\})$ with $r_i \prec r_j$, which means that $o(r_i) \subseteq o(r_j)$ and $q_i \leq q_j$. When $o(r_i) = o(r_j)$ and $q_i = q_j$, $r_i \prec r_j$ if r_j is posted later than r_i.

Note that R' in Definition 2 represents the reviews we consider at every iteration that have not been selected. The set of all reviews not dominated by any other one in R' constitutes a candidate set called *Skyline* [Borzsony et al. 2001], which can be formally defined as follows.

Definition 3 (Skyline). Given a review set R', its skyline R'_s is a set of reviews that are not dominated by any other review in R'—that is, $R'_s = R' \setminus \{r_i \mid r_i \in R', \exists r_j \in (R' \setminus \{r_i\})$ and $r_i \prec r_j\}$.

A fair amount of literature [Lee and Hwang 2010; Papadias et al. 2005; Tan et al. 2001] has been dedicated to skyline computation since its introduction [Borzsony et al. 2001]. Based on one of the most popular skyline computation algorithms, *Bitmap* [Tan et al. 2001], which was adopted in our preliminary study [Xu et al. 2014], we tailor a faster algorithm (Alg3_Skyline) for our specific problem. In Table 2, we use an example to explain how our Skyline algorithm works.

Given the review set $R' = \{r_1, r_2, r_3, r_4, r_5\}$, we assign the ordinal number of the review labels chronologically to keep track of the timestamps of the reviews (i.e., r_i is posted earlier than r_{i+1}). The corresponding opinions and quality scores for all reviews are displayed in Table 2(a). For skyline computation for R', we only need to consider the set of opinions O' that are not yet covered by the reviews selected in earlier iterations. In Table 2, there are three such opinions: o_1, o_2, and o_3. As defined in Section 3, matrix $A_{5\times3}$ in Table 2(b) represents the opinions contained in each review. We can obtain the bitmap data structure directly from the matrix.

Table 2(c) shows the sorted reviews after we put them in order first by quality score (from high to low), then the number of opinions covered (from high to low), and last time posted (from later to earlier). It is easy to conclude based on Definition 2 that a review is never dominated by any review placed lower on the sorted list. Thus, for each review on the list, we only need to check the preceding reviews rather than all other reviews as to whether it is dominated. With the sorted list, we generate the skyline as follows. We go through reviews on the list in a top-down order. The very top review is the first one added to the skyline, and for every other review on the list, we

check to see if it is dominated by the existing skyline. If not, it will be added to the skyline, and we move to the next review on the list. If it is dominated, we discard it and move on to check the next one.

To check whether a given review r_i is dominated by the current skyline consisting of reviews selected from the ones preceding r_i, we need to check whether there exists a review in the skyline that covers all opinions contained in r_i. Fortunately, we can leverage the bitwise AND operation to make this search process very fast by defining a function: $y = OV_{j_1} \& \cdots \& OV_{j_{|o(r_i)|}}$. Note, that $o(r_i)$ is all of the opinions r_i contains. We use an opinion vector OV_j to track which reviews in the current skyline cover opinion o_j. An opinion vector is a column vector and is a subset of a column in Table 2(b) because the opinion vector covers only reviews in the current skyline. As the size of the skyline changes during computation, the length of the opinion vector will change accordingly.

For example, assume the current skyline is $\{r_1, r_5\}$ when checking review r_2. Since r_2 has two opinions o_1 and o_3, we have two opinion vectors $\begin{bmatrix}1\\0\end{bmatrix}$ and $\begin{bmatrix}0\\1\end{bmatrix}$. The first vector $\begin{bmatrix}1\\0\end{bmatrix}$ indicates that r_1 covers o_1, and r_5 does not cover o_1. After we apply AND operations to these two vectors, we get $\begin{bmatrix}0\\0\end{bmatrix}$, which means that there exists no review that can cover all opinions in r_2 in the current skyline. If such a review exists, there will be at least one element in the result vector of y with value one. We can use the value of y to conclude whether review r_i is dominated by any one in the current skyline.

Based on the preceding example, we can easily see how to use y to help identify whether a given review r_i is dominated by any review already in the current skyline. If y is zero for review r_i, it means that no review in the current skyline contains all opinions in r_i. This review is not dominated and can be added to the skyline. If y is not zero, we can discard review r_i and move on to the next review on the sorted list. Because the list is already sorted by quality and time, we do not need to check these two metrics when identifying whether a review is dominated.

Next, we use this example to illustrate the whole skyline computation process. As review r_1 is on top of the list in Table 2(c), it is the first one added to the skyline. Next, we consider r_3 and calculate its y. Since $y = OV_2 = [1] \neq [0]$, r_3 is dominated by r_1 and thus discarded. Subsequently, we examine r_5 with $o(r_5) = \{o_3\}$, and $y = OV_3 = [0]$, so r_5 is added into the skyline. Then for r_4 with $o(r_4) = \{o_3\}$, $y = OV_3 = \begin{bmatrix}0\\1\end{bmatrix} \neq \begin{bmatrix}0\\0\end{bmatrix}$, which implies that review r_4 is a dominated review and can be discarded. Finally, we consider r_2 with $o(r_2) = \{o_1, o_3\}$. Because $y = OV_1 \ \& \ OV_3 = \begin{bmatrix}1\\0\end{bmatrix}$

ALGORITHM 3: The Accelerated Algorithm with Skyline (Alg3_Skyline)

Input: The review set $R = \{r_1, r_2, \ldots, r_n\}$, and its opinion set $O = \{o_1, o_2, \ldots, o_k\}$.
Output: A subset of the review set $S \subseteq R$.

$S = \emptyset$, $R' = R$;
The set of uncovered opinions $O' = O$;
while $|o(S)| < |O|$ **do**
  R'_s = Skyline(R', O');
  $r = \arg\min_{r_i \in R'_s} \frac{c(r_i)}{\Delta w(r_i, S)}$;
  $S = S \cup \{r\}$;
  Update O' based on review r;
  $R' = R'_s \setminus \{r\}$;
end
return S;

& $\begin{bmatrix}0\\1\end{bmatrix} = \begin{bmatrix}0\\0\end{bmatrix}$, r_2 is added to the skyline. The final skyline is displayed in Table 2(d). Due to space constraints, pseudocode of skyline computation is displayed in Online Appendix C.

Since bitwise operations are fast, we can efficiently prune the dominated reviews and accelerate the skyline detection process. In every iteration of Alg2_Rare, we can first detect the skyline and then only consider the reviews in the skyline. This strategy can significantly speed up the process. After each iteration when a review is added into the result set, the newly covered opinions will be removed from the uncovered opinion set to form the new O' for the next iteration.

With the efficient strategy of removing dominated reviews implemented, we present the accelerated Alg3_Skyline to further enhance the algorithm with the opinion rareness (Alg2_Rare). At the start of every iteration, we compute the skyline in Step 4 and then traverse reviews in the skyline to select the review with the minimum opinion cost in Step 5.

5 DYNAMIC UPDATING

Over time, the entire review set keeps changing with new reviews added and old reviews modified or deleted. In such a dynamic environment, how to update the representative review set efficiently is an important issue. Instead of conducting the selection process from scratch, we propose a partial updating algorithm, which can save a considerable amount of time.

There are three possible types of review changes: addition, deletion, and modification. We first describe the process of updating the representative review set for a review addition and then show that review deletion can be handled in the same way. A review modification can be treated as a review deletion and then addition.

To dynamically update the results as the review set changes, we need to record some intermediate results during the execution of Alg3_Skyline. First of all, we record the sequence of the reviews selected. The result set is represented as $S = \{r_{s_1}, \ldots, r_{s_{|S|}}\}$ with reviews sorted by the order they were selected (i.e., r_{s_i} is added before $r_{s_{i+1}}$). In addition, we record the set of uncovered opinions after every iteration during which a review is selected in Alg3_Skyline. The memory space used to save the intermediate results is small, as it is proportional to the number of selected reviews, which is much smaller compared to that of the original review set. Alg4_Dynamic describes the updating procedure, and the information we recorded during the execution of Alg3_Skyline serves as the input of Alg4_Dynamic.

If there is a newly added review, we can first find the iteration where the new review is superior to the review selected in that iteration in Alg3_Skyline. Generally speaking, in the earlier iterations in Alg3_Skyline, the newly added review is not the best candidate, thus having this new review will not change how the reviews are selected in those iterations. At a certain iteration, the newly added review becomes the best review and thus will be selected. Then for the iterations after this one, we need to recalculate the best review until the set of uncovered opinions becomes the same as the old recorded set of uncovered opinions in the same iteration. Once the sets of uncovered opinions are identical, the review selection becomes the same as before in the rest of the iterations.

Essentially, the new selection path will start to divert from the old selection path at a certain iteration, then the diverted path could also converge back to the old selection path at a later iteration after which we will select the same reviews as before. The dynamic updating procedure is to identify the start point and end point of such deviation, and the reviews selected before the start point and after the end point are exactly the same as those in the old result set S.

As for a deleted review, if it is not in the old result set, the review selection process will not be affected at all. If it is included, we can easily identify the iteration where this deleted review was selected and reselect the rest until the end point after which we keep the old selection process.

Table 3. Example to Illustrate the Dynamic Algorithm

(a) The result set

	$o(\cdot)$	q
r_1	$\{o_1, o_3\}$	0.8
r_2	$\{o_2, o_4\}$	0.6
r_3	$\{o_5, o_7\}$	0.2
r_4	$\{o_6, o_8\}$	0.5

(b) The start point

	i	$\frac{c(r_{s_i})}{\Delta w(r_{s_i}, S_{i-1})}$		$\frac{c(r_{new})}{\Delta w(r_{new}, S_{i-1})}$
	1	$\frac{1-0.8}{2} = 0.1$	$<$	$\frac{1-0.5}{4} = 0.125$
$\overrightarrow{start}$	2	$\frac{1-0.6}{2} = 0.2$	$>$	$\frac{1-0.5}{3} = 0.167$
	...	...		...

(c) The end point

	i	O'_i	O'_{new}
	1	...	...
$\overrightarrow{start}$	2	$\{o_5, o_6, o_7, o_8\}$	$\{o_6, o_7, o_8\}$
$\overrightarrow{end}$	3	$\{o_6, o_8\}$	$\{o_6, o_8\}$
	4	$\{\}$	$\{\}$

5.1 Start Point

At every iteration of Alg3_Skyline, the review with the minimum opinion cost is selected. We record the minima at each iteration. Starting from the first iteration, the new review's opinion cost is compared to the stored minima at each iteration. If the new review's opinion cost is not lower than the stored minima, we hold the review selected in the iteration and move to the next. If the new review has a lower opinion cost, it becomes the selected review at this iteration. This iteration is regarded as the start point of the dynamic updating. After this point, we follow the same selection procedure in Alg3_Skyline for later iterations until the end point.

A simple example in Table 3 further explains how to identify the start point. For illustration purposes, we assume that all eight opinions contained in the original review set R have the same rareness (i.e., $w_1 = \cdots = w_8 = 1$) and that these rareness values remain fixed when a new review is posted. The current representative review set is $S = \{r_1, r_2, r_3, r_4\}$ as shown in Table 3(a). Then, a new review r_{new} with $o(r_{new}) = \{o_1, o_2, o_4, o_5\}$ and $q_{new} = 0.5$ is added. We compare the opinion cost of the new review to each review in S in Table 3(b). Here, we assume the weight λ used in the cost $c(r_i)$ is one. The first selected review r_1 has a smaller opinion cost than r_{new}, and thus r_1 cannot be replaced. Compared to the second review r_2, r_{new} has a smaller opinion cost, so r_2 is substituted with r_{new}. The start point is right before the second iteration. Table 3(c) is used to further illustrate how to identify the end point, which will be discussed next.

5.2 End Point

Based on Section 5.1, we can find the start point of the update and redo the selection after that. However, we do not need to redo the selection for all iterations after the start point. If there exists an iteration where the uncovered opinion set is exactly the same as the one recorded, we can preserve the previous selection results after this point, which is called the *end point*.

The example in Table 3(c) further illustrates the process of finding the end point. The $O_i^{'}$ column is the uncovered opinion set recorded before and the $O_{new}^{'}$ column is the uncovered opinion set in the updating procedure. After the second iteration, r_{new} is selected, and o_5 is covered by r_{new}. Then, the set of uncovered opinions is not identical to O'_2. After the third iteration, the uncovered opinion set $O_{new}^{'}$ becomes $\{o_6, o_8\}$, which is identical to O'_3. Thus, this iteration can be regarded as the end point, and we can preserve the previous selection results $\{r_4\}$ after this point.

5.3 The Dynamic Algorithm

Benefiting from the detection of the start and end points, we design a partial updating algorithm displayed in Alg4_Dynamic. Here, we record intermediate results of the previous selection process and use them as inputs. The added storage space is very small compared to the original review set. We detect the start point in Steps 3 through 7 and then redo the review selection until the end point in Steps 12 through 23. In addition, this algorithm can be easily extended to handle the review deletion and review modification. For the review deletion, the start point is just the iteration

where the deleted review is selected, and the detection of the end point is exactly the same. For the review modification, we can first do a review deletion and then a review addition.

ALGORITHM 4: The Dynamic Algorithm (Alg4_Dynamic)

Input: The review set R, its opinion set O, and the new review r_{new}. For i from 1 to $|S|$, the set of the first i reviews selected $S_i = \{r_{s_1}, r_{s_2}, \ldots, r_{s_i}\}$ with reviews sorted by the order they were selected (i.e., r_{s_j} is added before $r_{s_{j+1}}$), and the corresponding set of the uncovered opinions O'_i after the i^{th} iteration in Alg3_Skyline. Note that $S_{|S|}$ is also the current result set S.

Output: The new result set S_{new}.

```
1  S_new = ∅, R' = R ∪ {r_new}, Idx = 0;
2  The set of uncovered opinions O'_new = O;
3  for i ← 1 to |S| do
4      if c(r_new)/Δw(r_new, S_{i−1}) ≤ c(r_i)/Δw(r_i, S_{i−1}) then
5          Let Idx = i and break;
6      end
7  end
8  if Idx ≠ 0 then
9      S_new = S_{Idx−1} ∪ {r_new};
10     R' = R' \ S_new;
11     Update O'_new based on S_new;
12     while |o(S_new)| < |O| do
13         if O'_new ≠ O'_Idx then
14             R'_s = Skyline(R', O'_new);
15             r = arg min_{r_i ∈ R'_s} c(r_i)/Δw(r_i, S_new);
16             S_new = S_new ∪ {r};
17             Update O'_new based on review r;
18             R' = R'_s \ {r};
19             Idx++;
20         else
21             Break and go to Step 24;
22         end
23     end
24     S_new = S_new ∪ (S \ S_Idx);
25 else
26     S_new = S;
27 end
28 return S_new;
```

6 EXPERIMENTS

In this section, several experiments are conducted on real datasets to validate our ideas and evaluate our algorithms. We begin with a brief description of the datasets and some relevant preprocessing procedures, then introduce baseline methods, followed by the discussion of our experimental results.

6.1 Data Description

We constructed twelve real-world datasets based on reviews collected from Yelp and Amazon. Specifically, we have three hotel datasets, three datasets on tablets, three mobile phone datasets on different brands (Apple, BLU, and Samsung), and three datasets on different business categories (Bars, Nightlife, and Restaurants). These datasets covering reviews on different types of entities increase the diversity of the data used to evaluate our algorithms. In addition, the size of all review sets ranges from very small to very big so that we can observe how the performance of the algorithms changes as the size of data increases. For each review, its author profile, review content, and usefulness score is obtained. Descriptive statistics of each dataset are displayed in Table 4.

Table 4. Summary of Datasets

Dataset	Hotel			Tablet			Mobile			Business		
	1	2	3	1	2	3	Apple	BLU	Samsung	Bars	Nightlife	Restaurants
Reviews (#)	3,430	3,622	3,634	5,857	12,565	24,955	17,249	18,982	35,966	218,717	256,219	849,883
Features (#)	60	60	60	86	86	86	74	75	75	50	50	50
Opinions (#)	120	120	119	162	172	170	143	149	150	100	100	100

6.2 Preprocessing

Before implementing the proposed review selection algorithms, we adopt some necessary procedures to preprocess the data, namely feature extraction, sentiment analysis, and quality estimation:

- *Feature extraction*: In this step, we use the open source Nature Language Processing API TextBlob [Loria et al. 2018] to select frequent noun expressions as candidates for features. Then, we remove meaningless expressions manually and obtain the feature set for each dataset respectively. This flexible method can be applied to various domains and is easily adapted to our diverse datasets.
- *Sentiment analysis*: The sentiment of a review may be positive, but some features discussed in the review may still have negative sentiments. In our article, we extract distinct sentiment polarities for each feature in every review. Various studies have approached sentiment analysis at the feature level through lexicon-based methods and machine learning techniques [Schouten and Frasincar 2015]. In this article, we use TextBlob APIs to perform dependency parsing for each sentence and sentiment analysis for each clause extracted from the sentence.
- *Quality estimation*: To estimate the quality of a review, we adopt a quality measurement capturing three dimensions of a review: functionality, reliability, and usability [Knight and Burn 2005], which can be assessed from its content, author, and vote information. For each review r_i, we extract information from the content to reflect its functionality, including the review's length, t_i, the number of features, n_i, and the number of opinion words, p_i. Author information is about the author's reliability level, v_i/w_i, where w_i is the total number of reviews authored by the user and v_i is the number of useful votes the user has received. The usability of a review is described by the number of useful votes a review has received, u_i. Based on the preceding information, each factor is normalized respectively (i.e., $Scl(x_i) = \frac{x_i - \min(\mathbf{x})}{\max(\mathbf{x}) - \min(\mathbf{x})}$). Afterward, we use the average of all normalized factors to measure the overall quality of a review, $q_i = \frac{Scl(t_i)+Scl(n_i)+Scl(p_i)+Scl(\frac{v_i}{w_i})+Scl(u_i)}{5}$.

6.3 Candidate Algorithms

In this article, we proposed four algorithms: Alg1_Greedy (Basic Greedy Algorithm), Alg2_Rare (Enhanced Algorithm with Opinion Rareness), Alg3_Skyline (Accelerated Algorithm with Skyline), and Alg4_Dynamic (Dynamic Algorithm). We conduct experiments to demonstrate the advantages of our enhanced algorithms over our basic ones to show the degree of improvement in terms of efficiency and reduction in result size. Alg2_Rare should strictly dominate Alg1_Greedy in terms of the result size while comparable on efficiency. Alg3_Skyline should strictly dominate Alg2_Rare in terms of efficiency while generating the same result review set. Alg4_Dynamic is the extension of Alg3_Skyline for the dynamic situation and should have higher efficiency than Alg3_Skyline when the result review set needs to be updated. Thus, Alg3_Skyline and

Table 5. Comparison of the Result Size

Dataset	Hotel			Tablet			Mobile			Business		
	1	2	3	1	2	3	Apple	BLU	Samsung	Bars	Nightlife	Restaurants
Reviews (#)	3,430	3,622	3,634	5,857	12,565	24,955	17,249	18,982	35,966	218,717	256,219	849,883
Base1_IP	20	21	21	48	36	34	49	23	24	12	12	11
Base4_RUO	23	24	25	50	43	37	51	27	27	16	16	14
Alg1_Greedy	23	27	27	54	43	39	54	27	26	15	16	13
Alg2_Rare & Alg3_Skyline	**21**	**22**	**23**	**49**	**39**	**36**	**50**	**24**	**26**	**14**	**15**	**13**

Alg4_Dynamic are our final algorithms that can be deployed in application settings. In addition, we compare our algorithms to several baseline algorithms:

- *Base1_IP (integer programming)*: A method that provides an exact solution to Problem P_c. We use an efficient linear integer programming solver named *lp_solve* (available from http://lpsolve.sourceforge.net) based on the revised simplex method and the branch-and-bound method.
- *Base2_RQ (ranking by quality)*: A method that ranks all reviews by the quality and then selects the top-k reviews with the highest quality.
- *Base3_GQC (group quality coverage)*: The method used in Tsaparas et al. [2011]. It can select a given number (k) of high-quality reviews to cover as many different features as possible from reviews partitioned by groups with different sentiment polarities. Since this method only utilizes the review-level sentiment, which is integrated into the grouping of reviews, we label the features in a review with the same sentiment as the review's sentiment to proceed with the comparison.
- *Base4_RUO (rarest uncovered opinion)*: The method proposed in our preliminary study [Xu et al. 2014]. It also aims to select high-quality reviews that can cover all opinions. However, this method is much simpler than our current proposed algorithm. At each iteration, the strategy in the earlier work is to select the review with the rarest uncovered opinion instead of using the opinion cost. In addition, Base4_RUO only adopted the original skyline computation *Bitmap* [Tan et al. 2001], whereas this article tailors a faster algorithm.

6.4 Experimental Results

Since the quality score is normalized between 0 and 1, we first set $\lambda = 1$ for our experiments. All algorithms, except for Base2_RQ and Base3_GQC, are required to cover all distinct opinions without the preset parameter k. Our algorithms can tailor the set of opinions to be covered for different individuals if such information is available.

6.4.1 Size and Coverage Comparison. We first compare our algorithms to baselines with respect to size and coverage. Table 5 presents the size of the result set for the algorithms to which this comparison applies. The results show that our proposed algorithms Alg2_Rare and Alg3_Skyline provide the most compact result set than the other two approximation methods (on average 8.3% more compact than Alg1_Greedy and 6.8% than Base4_RUO). Note that Base1_IP provides the exact solution, so its results have the smallest size. Compared to the size of the original set (the second row), the number of reviews in the representative set (the last row) is extremely small. This can lead to a significant time saving on learning all of the opinions that users concern, making our method more practical to handle different information needs in real-world applications.

Table 6. Coverage of Base2_RQ and Base3_GQC

Dataset	Hotel			Tablet			Mobile			Business		
	1	2	3	1	2	3	Apple	BLU	Samsung	Bars	Nightlife	Restaurants
k	21	22	23	49	39	36	50	24	26	14	15	13
Base2_RQ	75.0%	82.5%	78.2%	79.0%	79.7%	81.8%	67.8%	85.2%	81.3%	64.0%	65.0%	70.0%
Base3_GQC	85.0%	84.2%	83.2%	82.1%	78.5%	86.5%	81.1%	87.2%	83.3%	74.0%	76.0%	83.0%

Table 7. Comparison of the Average Runtime

Dataset	Hotel			Tablet			Mobile			Business		
	1	2	3	1	2	3	Apple	BLU	Samsung	Bars	Nightlife	Restaurants
Reviews (#)	3,430	3,622	3,634	5,857	12,565	24,955	17,249	18,982	35,966	218,717	256,219	849,883
Base1_IP	551 ms	119 ms	475 ms	81 ms	405 ms	288 ms	115 ms	300 ms	750 ms	1 m 31 s	14 m 33 s	1 h 22 m 35 s
Base2_RQ	2.57 ms	2.90 ms	3.04 ms	1.33 ms	2.23 ms	4.53 ms	3.54 ms	2.05 ms	4.13 ms	0.029 s	0.049 s	0.171 s
Base3_GQC	42.7 s	32.3 s	37.5 s	6 m 45 s	21 m 55 s	1 h 20 m	49 m 34 s	26 m 57 s	1 h 53 m	$\gg$1 h	$\gg$1 h	$\gg$1 h
Base4_RUO	6.24 ms	5.99 ms	6.35 ms	3.64 ms	12.02 ms	27.50 ms	5.08 ms	12.76 ms	14.18 ms	1.083 s	1.264 s	14.49 s
Alg1_Greedy	6.32 ms	7.15 ms	8.34 ms	6.05 ms	23.33 ms	48.56 ms	20.52 ms	17.65 ms	28.28 ms	0.222 s	0.265 s	1.044 s
Alg2_Rare	7.24 ms	7.51 ms	8.63 ms	5.72 ms	21.85 ms	44.42 ms	20.30 ms	19.16 ms	29.83 ms	0.235 s	0.276 s	1.111 s
Alg3_Skyline	4.06 ms	4.86 ms	5.65 ms	2.58 ms	6.87 ms	12.67 ms	2.65 ms	5.25 ms	8.16 ms	0.130 s	0.147 s	0.879 s

We further compare our methods with Base2_RQ and Base3_GQC, both of which select k reviews. The input parameter k can be determined by the size of the result set generated by Alg2_Rare, which means that the second column in Table 6 is the same as the last column in Table 5. Table 6 presents the percentage of opinions covered by the result review set selected by Base2_RQ and Base3_GQC. As we can see, Base2_RQ and Base3_GQC can only cover about 60% to 80% of the opinions by selecting the same number of reviews as Alg2_Rare and Alg3_Skyline, whereas Alg2_Rare and Alg3_Skyline can cover all of the distinct opinions in such a compact review set.

We further compare the quality of the reviews selected by different algorithms, and the comparison presented in Online Appendix D illustrates that our proposed algorithms can select compact results with complete coverage without sacrificing review quality. Moreover, we present a case study in Online Appendix E to show some selected reviews and illustrate opinions covered by the reviews.

6.4.2 Efficiency Comparison. Compared to existing methods, our algorithms are not only effective but also efficient. Base1_IP is time consuming because getting an exact solution to the set cover problem is NP-hard. In Base3_GQC, reviews are partitioned into two groups based on their sentiments. At every iteration, Base3_GQC traverses all possible pairs generated by the Cartesian product of the two groups, so the time complexity is $O(kn^2)$. However, our algorithms traverse candidate reviews one by one at every iteration, and thus the time complexity is $O(kn)$, which is the same as that of Base2_RQ. Table 7 presents each alternative's average runtime of 10 runs. All algorithms are implemented on a desktop with an Intel Core-I5 CPU and 8 GB of RAM. Our proposed Alg3_Skyline algorithm saves more than 90% of the time compared to Base1_IP and Base3_GQC, and especially for larger datasets with hundreds of thousands of reviews (Business: Bars, Nightlife, and Restaurants), Alg3_Skyline can save more than 99% of the runtime. Since Base2_RQ is developed based on a very simple heuristic that is selecting the top-k reviews with the highest quality,

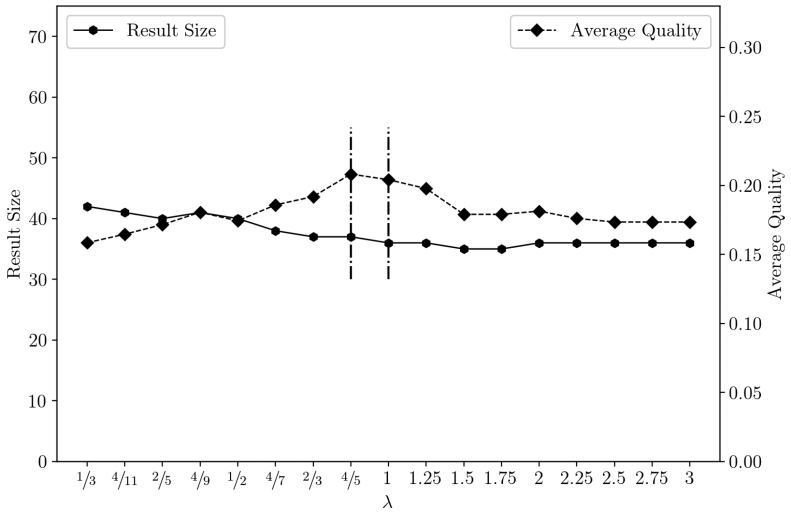

Fig. 2. Choice of λ for the dataset Tablet-3.

it is expected to have the lowest runtime among all alternatives. Our proposed Alg3_Skyline algorithm is comparable to Base2_RQ in terms of efficiency and performs much better than Base2_RQ in terms of coverage. With the help of the enhanced strategy, our proposed accelerated algorithm (Alg3_Skyline) also saves considerable time for each dataset compared to Base4_RUO (50.85% on average), Alg1_Greedy (52.68% on average), and Alg2_Rare (54.43% on average).

6.4.3 Varying λ and Managerial Decisions. To further examine the robustness of our algorithms, we conduct sensitivity analysis on the scalarization weight λ by comparing the value of the overall objective function achieved by different methods under different λ. Due to space constraints, we present the full analysis in Online Appendix F. The analysis indicates the robustness of our experimental performances.

When varying the scalarization weight λ, managers can obtain many different candidate solutions. Managers can further analyze the relationship between the result size and review quality to facilitate their decision making in choosing the appropriate λ. Taking dataset Tablet-3 as an example, the result size and the average quality score as λ changes are plotted in Figure 2.

As we can see, the result size is relatively small as λ changes within the range of $[2/3, 1]$, and within the same range, average review quality reaches the highest when λ takes the value of $4/5$ and 1. A reasonable λ choice for managers could be $4/5$ and 1, which properly balance both result size and review quality. If the managers are concerned about the result size, they can choose a slightly bigger λ. As is illustrated using this example, we can see how managers can use the results to balance result size and review quality to make the appropriate λ choice based on their preferences. Moreover, managers can also use other means to choose λ. For example, they can introduce external metrics to guide them to select the proper λ, such as closeness [Lappas et al. 2012], consistency [Zhang et al. 2016], and information diversity [Ma et al. 2017]. They can choose the λ that maximizes the closeness/consistency/diversity of the result review set.

6.4.4 Efficiency Improvement of the Skyline Strategy. As illustrated in Section 6.4.2, Alg3_Skyline is the most scalable algorithm. In each iteration, Alg3_Skyline prunes the dominated reviews and retains the rest as the skyline set. As displayed in Table 8, on average, about

Table 8. Proportion of Reviews Pruned by Alg3_Skyline

Dataset	Hotel			Tablet			Mobile			Business		
	1	2	3	1	2	3	Apple	BLU	Samsung	Bars	Nightlife	Restaurants
Reviews (#)	3,430	3,622	3,634	5,857	12,565	24,955	17,249	18,982	35,966	218,717	256,219	849,883
Reviews after Skyline (#)	1,592	1,467	1,599	664	1,596	2,312	701	1,681	1,928	25,112	26,206	76,045
Reviews pruned by Skyline	**53.6%**	**59.5%**	**56.0%**	**88.7%**	**87.3%**	**90.7%**	**95.9%**	**91.1%**	**94.6%**	**88.5%**	**89.8%**	**91.1%**

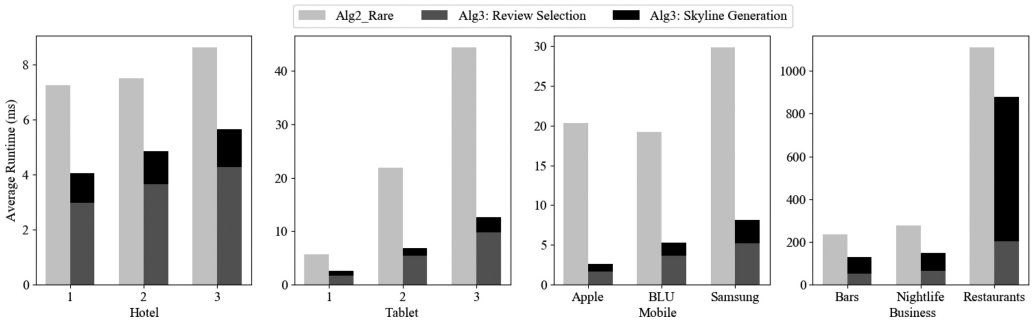

Fig. 3. Time saving with Skyline generation.

80% of all reviews are pruned and the size of the candidate set is dramatically reduced across all datasets.

In addition, we divide the time of Alg3_Skyline into two parts: the time spent on generating the skyline and the time spent on selecting reviews after the skyline is generated. In Figure 3, the bar on the left corresponds to the time for Alg2_Rare, and the sum of the two stacked bars on the right is the time for Alg3_Skyline. The time is averaged over 100 runs. With the little time added to generate the skyline (labeled as Alg3: Skyline Generation), we can significantly reduce the time spent on review selections (labeled as Alg3: Review Selection). The overall runtime is lower than that of Alg2_Rare, especially when the initial review set is large.

6.4.5 Efficiency Improvement of the Dynamic Algorithm. In this section, the efficiency improvement of the dynamic algorithm Alg4_Dynamic over Alg3_Skyline is assessed. For each review dataset, we split it into two parts: the first part (Part I) for selecting reviews as the input of Alg4_Dynamic, and the second part (Part II) for implementing Alg4_Dynamic to dynamically update the selected reviews as new reviews are added. First, Alg3_Skyline is applied on Part I (the old review set) to select the initial subset of representative reviews called the *old result set*, then for each review in Part II, we treat it as the newly added review and use Alg4_Dynamic to update the selected reviews. To compare the efficiency of Alg4_Dynamic with that of Alg3_Skyline, Alg3_Skyline is applied to the set of reviews containing the old review set and the newly added review, whereas Alg4_Dynamic is applied to the old result set and the newly added review. Then, both the old review set and the old result set are updated, and we repeat this process for the reviews in Part II one by one. For each iteration (one additional review in Part II), we record the time spent for Alg3_Skyline and Alg4_Dynamic respectively. We vary the percentage of Part I from 10% to 50% (i.e., 10%, 25%, 50%) and present the total runtime (the sum of the runtime of each iteration) in Table 9. The time saving is significant when Alg4_Dynamic is adopted. Further, we can anticipate

Table 9. Comparison of the Total Runtime between Alg3_Skyline and Alg4_Dynamic

Part I (%)	Algorithm	Hotel			Tablet			Mobile			Business		
		1	2	3	1	2	3	Apple	BLU	Samsung	Bars	Nightlife	Restaurants
10	Alg3_Skyline	2,772.6 ms	2,712.8 ms	2,993.0 ms	1,097.5 ms	3,674.7 ms	9,353.0 ms	1,984.6 ms	12,080.4 ms	8,664.4 ms	1,252.4 s	14,12.7 s	10,293.7 s
	Alg4_Dynamic	**21.2** ms	**33.4** ms	**16.1** ms	**45.7** ms	**43.5** ms	**54.1** ms	**55.1** ms	**88.1** ms	**52.3** ms	**0.665** s	**0.590** s	**2.338** s
25	Alg3_Skyline	2,041.9 ms	2,101.3 ms	2,370.8 ms	682.8 ms	3,174.1 ms	7,146.0 ms	1,284.6 ms	5,813.6 ms	5,683.5 ms	797.5 s	978.6 s	7,871.4 s
	Alg4_Dynamic	**11.7** ms	**25.2** ms	**10.8** ms	**31.5** ms	**35.5** ms	**42.4** ms	**40.9** ms	**42.5** ms	**28.2** ms	**0.572** s	**0.511** s	**2.297** s
50	Alg3_Skyline	1,414.6 ms	1,401.6 ms	1,604.4 ms	501.2 ms	1,750.8 ms	1,679.8 ms	572.3 ms	2,438.2 ms	3,190.2 ms	560.9 s	637.4 s	4933.7 s
	Alg4_Dynamic	**6.58** ms	**16.2** ms	**3.1** ms	**22.0** ms	**19.4** ms	**24.1** ms	**21.8** ms	**19.2** ms	**18.0** ms	**0.375** s	**0.318** s	**1.996** s

that the time saving from processing review deletion and modification should have a compatible scale with that from review addition processed here.

7 CONCLUSION AND FUTURE WORK

In this article, we propose methods to select a compact and high-quality set of representative reviews from a large number of online reviews. The selected reviews can cover a specific set of features and their sentiment polarities. The representative review selection problem is modeled as a bi-objective combinatorial optimization problem. The first objective is to find a review set as compact as possible, and the second objective is that the quality of the selected reviews is as high as possible. To solve the bi-objective combinatorial optimization problem defined, the weighted sum scalarization method is adopted to transform the problem with two objectives into a single objective problem that can be regarded as a minimum-cost set cover problem. Since the minimum-cost set cover problem is NP-complete, several algorithms are designed to approximate the optimal solution. We first introduce a basic greedy algorithm, on top of which we construct an enhanced algorithm based on the concept of opinion rareness to generate a more compact review set. We further develop an accelerated strategy for efficiency enhancement by pruning dominated reviews. In addition, we develop an efficient algorithm to dynamically update the representative review set as reviews are added, removed, or modified. Theoretical analysis of these algorithms is provided to prove that they can hold performance guarantees. Comprehensive experiments are conducted on real-world review sets. We compare the performances of our algorithms and existing methods based on the size of the review set, the coverage, and the runtime. Experimental results demonstrate the superiority of our algorithms along all dimensions.

The problem defined in this article is very comprehensive, capturing important aspects of a review selection process in real-world applications. Due to the excellent performance of our solution, it can be very practical for Web sites to adopt. The site can potentially learn the features the consumers are most interested in based on historical behavior observed. When presenting reviews to different users, they can first show a very small subset of selected reviews to cover almost all of the content they care most about. This will make users informed about the product very quickly, thus leading to a higher chance for the users to conduct business on a particular Web site instead of other competing Web sites that do not offer this option. The seller of the product can also have a compact review set to obtain a comprehensive overview of consumer feedback.

Our research has several limitations that can be explored in the future. First, the quality estimation process considers vote information of a review on top of the author and content information. For newly added reviews, it will take time to observe how users vote them. In the future, we can consider using predictive models to estimate vote information for new reviews. Another limitation of our research is that the opinion rareness of each review is assumed to stay unchanged when using dynamic algorithms to update the result review set. In practice, the opinion rareness

can be updated periodically as more reviews are generated. Furthermore, in our current study, we take feature extraction, sentiment analysis, and quality estimation as preprocessing steps that take place before review selection. In future research, we can investigate whether it is possible to incorporate them into the review selection process to achieve even better results.

REFERENCES

Charu C. Aggarwal and ChengXiang Zhai. 2012. A survey of text clustering algorithms. In *Mining Text Data*. Springer, 77–128.

Stephan Borzsony, Donald Kossmann, and Konrad Stocker. 2001. The skyline operator. In *Proc.eedings of the 17th IInternational Conference on Data Engineering*. IEEE, Los Alamitos, CA, 421–430.

Roque Enrique López Condori and Thiago Alexandre Salgueiro Pardo. 2017. Opinion summarization methods: Comparing and extending extractive and abstractive approaches. *Expert Systems with Applications* 78 (2017), 124–134.

Dingzhu Du, Ker I. Ko, and Xiaodong Hu. 2011. *Design and Analysis of Approximation Algorithms*. Vol. 62. Springer Science & Business Media.

Matthias Ehrgott. 2006. *Multicriteria Optimization*. Springer Science & Business Media.

Matthias Ehrgott and Xavier Gandibleux. 2000. A survey and annotated bibliography of multiobjective combinatorial optimization. *OR-Spektrum* 22, 4 (2000), 425–460.

eMarketer. 2016. Consumers like reading online reviews, not writing them. https://www.emarketer.com/Article/Consumers-Like-Reading-Online-Reviews-Not-Writing-Them/1014242.

Mahak Gambhir and Vishal Gupta. 2017. Recent automatic text summarization techniques: A survey. *Artificial Intelligence Review* 47, 1 (2017), 1–66.

Anindya Ghose and Panagiotis G. Ipeirotis. 2011. Estimating the helpfulness and economic impact of product reviews: Mining text and reviewer characteristics. *IEEE Transactions on Knowledge and Data Engineering* 23, 10 (2011), 1498–1512.

Xunhua Guo, Qiang Wei, Guoqing Chen, Jin Zhang, and Dandan Qiao. 2017. Extracting representative information on intra-organizational blogging platforms. *MIS Quarterly* 41, 4 (2017), 1105–1127.

Hong Hong, Di Xu, G. Alan Wang, and Weiguo Fan. 2017. Understanding the determinants of online review helpfulness: A meta-analytic investigation. *Decision Support Systems* 102 (2017), 1–11.

David S. Johnson. 1973. Approximation algorithms for combinatorial problems. In *Proceedings of the 5th ACM Symposium on Theory of Computing*. ACM, New York, NY, 38–49.

Richard M. Karp. 1972. Reducibility among combinatorial problems. In *Complexity of Computer Computations*. Springer, 85–103.

Shirlee-Ann Knight and Janice Burn. 2005. Developing a framework for assessing information quality on the World Wide Web. *Informing Science* 8 (2005), 1–14.

Theodoros Lappas, Mark Crovella, and Evimaria Terzi. 2012. Selecting a characteristic set of reviews. In *Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, New York, NY, 832–840.

Jongwuk Lee and Seungwon Hwang. 2010. QSkycube: Efficient Skycube computation using point-based space partitioning. *Proceedings of the VLDB Endowment* 4, 3 (2010), 185–196.

Yang Liu, Xiangji Huang, Aijun An, and Xiaohui Yu. 2008. Modeling and predicting the helpfulness of online reviews. In *Proceedings of the 8th International Conference on Data Mining (ICDM'08)*. IEEE, Los Alamitos, CA, 443–452.

Steven Loria, P. Keen, M. Honnibal, R. Yankovsky, D. Karesh, E. Dempsey, W. Childs, et al. 2018. TextBlob: Simplified text processing. Retrieved November 13, 2019 from https://textblob.readthedocs.io/en/dev/.

Baojun Ma, Qiang Wei, Guoqing Chen, Jin Zhang, and Xunhua Guo. 2017. Content and structure coverage: Extracting a diverse information subset. *INFORMS Journal on Computing* 29, 4 (2017), 660–675.

Thanh-Son Nguyen, Hady W. Lauw, and Panayiotis Tsaparas. 2015. Review selection using micro-reviews. *IEEE Transactions on Knowledge and Data Engineering* 27, 4 (2015), 1098–1111.

Thanh-Son Nguyen, Hady W. Lauw, and Panayiotis Tsaparas. 2017. Micro-review synthesis for multi-entity summarization. *Data Mining and Knowledge Discovery* 31, 5 (2017), 1189–1217.

Dimitris Papadias, Yufei Tao, Greg Fu, and Bernhard Seeger. 2005. Progressive skyline computation in database systems. *ACM Transactions on Database Systems* 30, 1 (2005), 41–82.

Debanjan Paul, Sudeshna Sarkar, Muthusamy Chelliah, Chetan Kalyan, and Prajit Prashant Sinai Nadkarni. 2017. Recommendation of high quality representative reviews in e-commerce. In *Proceedings of the10th ACM Conference on Recommender Systems*. ACM, New York, NY, 311–315.

Kim Schouten and Flavius Frasincar. 2015. Survey on aspect-level sentiment analysis. *IEEE Transactions on Knowledge and Data Engineering* 28, 3 (2015), 813–830.

Kian Lee Tan, Pin Kwang Eng, and Beng Chin Ooi. 2001. Efficient progressive skyline computation. In *Proceedings of the 27th International Conference on Very Large Data Bases (VLDB'01)*, Vol. 1. 301–310.

Panayiotis Tsaparas, Alexandros Ntoulas, and Evimaria Terzi. 2011. Selecting a comprehensive set of reviews. In *Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, New York, NY, 168–176.

Vijay V. Vazirani. 2013. *Approximation Algorithms*. Springer Science & Business Media.

Nana Xu, Hongyan Liu, Jiawei Chen, Jun He, and Xiaoyong Du. 2014. Selecting a representative set of diverse quality reviews automatically. In *Proceedings of the SIAM International Conference on Data Mining (SDM'14)*. 488–496.

Wenzhe Yu, Rong Zhang, Xiaofeng He, and Chaofeng Sha. 2013. Selecting a diversified set of reviews. In *Proceedings of the Asia-Pacific Web Conference*. 721–733.

Zunqiang Zhang, Guoqing Chen, Jin Zhang, Xunhua Guo, and Qiang Wei. 2016. Providing consistent opinions from online reviews: A heuristic stepwise optimization approach. *INFORMS Journal on Computing* 28, 2 (2016), 236–250.

Received December 2018; revised August 2019; accepted October 2019