Efficient Confident Search in Large Review Corpora

Theodoros Lappas[1] and Dimitrios Gunopulos[2]

[1] UC Riverside
[2] University of Athens

Abstract. Given an extensive corpus of reviews on an item, a potential customer goes through the expressed opinions and collects information, in order to form an educated opinion and, ultimately, make a purchase decision. This task is often hindered by false reviews, that fail to capture the true quality of the item's attributes. These reviews may be based on insufficient information or may even be fraudulent, submitted to manipulate the item's reputation. In this paper, we formalize the *Confident Search* paradigm for review corpora. We then present a complete search framework which, given a set of item attributes, is able to efficiently search through a large corpus and select a compact set of high-quality reviews that accurately captures the overall *consensus* of the reviewers on the specified attributes. We also introduce **CREST** (Confident REview Search Tool), a user-friendly implementation of our framework and a valuable tool for any person dealing with large review corpora. The efficacy of our framework is demonstrated through a rigorous experimental evaluation.

1 Introduction

Item reviews are a vital part of the modern e-commerce model, due to their large impact on the opinions and, ultimately, the purchase decisions of Web users. The nature of the reviewed items is extremely diverse, spanning everything from commercial products to restaurants and holiday destinations. As review-hosting websites become more popular, the number of available reviews per item increases dramatically. Even though this can be viewed as a healthy symptom of online information sharing, it can also be problematic for the interested user: as of February of 2010, Amazon.com hosted over 11,480 reviews on the popular "Kindle" reading device. Clearly, it is impractical for a user to read through such an overwhelming review corpus, in order to make a purchase decision. In addition, this massive volume of reviews on a single item inevitably leads to redundancy: many reviews are often repetitious, exhaustively expressing the same (or similar) opinions and contributing little additional knowledge. Further, reviews may also be misleading, reporting false information that does not accurately represent the attributes of an item. Possible causes of such reviews include:

- **Insufficient information:** The reviewer proceeds to an evaluation without having enough information on the item. Instead, opinions are based on partial or irrelevant information.
- **Fraud:** The reviewer maliciously submits false information on an item, in order to harm or boost its reputation.

J.L. Balcázar et al. (Eds.): ECML PKDD 2010, Part II, LNAI 6322, pp. 195–210, 2010.

The main motivation of our work is that a user should not have to manually go through massive volumes of redundant and ambiguous data in order to obtain the required information. The search engines that are currently employed by major review-hosting sites do not consider the particular nature of opinionated text. Instead, reviews are evaluated as typical text segments, while focused queries that ask for reviews with opinions on *specific* attributes are not supported. In addition, reviews are ranked based on very basic methods (e.g. by date) and information redundancy is not considered.

Ideally, false or redundant reviews could be filtered before they become available to users. However, simply labeling a review as "true" or "false" is over-simplifying, since a review may only be partially false. Instead, we propose a framework that evaluates the validity of the opinions expressed in a review and assigns an appropriate *confidence score*. High confidence scores are assigned to reviews expressing opinions that respect the *consensus* formed by the entire review corpus. For example, if 90% of the reviews compliment the battery-life of a new laptop, there is a strong positive consensus on the specific attribute. Therefore, any review that criticizes the battery-life will suffer a reduction in its confidence score, *proportional to the strength of the positive consensus*. At this point, it is important to distinguish between the two types of rare opinions: 1) those that are expressed on attributes that are rarely reviewed and 2) those that contradict the opinion of the majority of the reviewers on a specific attribute. Our approach only penalizes the latter, since the rare opinions in the first group can still be valid (e.g. expert opinions, commenting on attributes that are often overlooked by most users). Further, we employ a simple and efficient method to deal with ambiguous attributes, for which the numbers of positive and negative opinions differ marginally.

Confidence evaluation is merely the first phase of our framework; high-confidence reviews may still be redundant, if they express identical opinions on the same attributes. To address this, we propose an efficient redundancy filter, based on the skyline operator [2]. As shown in the experiments section, the filter achieves a significant reduction of the size of the corpus.

The final component of our framework deals with the evaluation of focused queries: given a set of attributes that the user is interested in, we want to identify *a minimal* set of high-confidence reviews that covers all the specified attributes. To address this, we formalize the *Review Selection* problem for large review corpora and propose a customized search engine for its solution. A complete diagram of our framework can be seen in Figure (1). Figure (2) shows a screenshot of **CREST** (Confident REview Search Tool), a user-friendly tool that implements the full functionality of our framework. In the shown example, **CREST** is applied on a corpus of reviews on a popular Las Vegas hotel. As soon as a review corpus is loaded, **CREST** evaluates the confidence of the available reviews and filters out redundant artifacts. The user can then select a set of features from a list extracted automatically from the corpus. The chosen set is submitted as a query to the search engine, which returns a compact and informative set of reviews. It is important to stress that our engine has no bias against attributes that appear sparsely in the corpus: as long as the user includes an attribute in the query, an appropriate review will be identified and included in the solution.

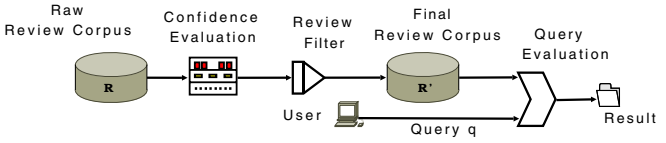


Fig. 1. Given a review corpus $\mathcal{R}$, we first evaluate the *confidence* of each review $r \in \mathcal{R}$. Then, the corpus is filtered, in order to eliminate redundant reviews. Finally, given a query of attributes, the search engine goes through the processed corpus to evaluate the query and select an appropriate set of reviews.

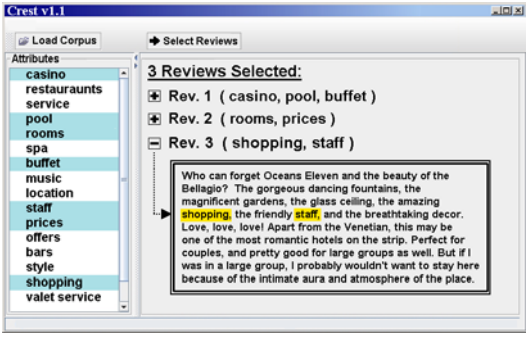


Fig. 2. A user loads a corpus of reviews and then chooses a query of attributes from the automatically-extracted list on the left. The "Select Reviews" button prompts the system to return an appropriate minimal set of reviews.

Contribution: Our primary contribution is an efficient search engine that is customized for large review corpora. The proposed framework can respond to any attribute-based query by returning an appropriate minimal subset of high-quality reviews.

Roadmap: We begin in Section 2 with a discussion on related work. In section 3 we introduce the Confident Search paradigm for large review corpora. In Section 4 we describe how we measure the quality of a review through evaluating the confidence in the opinions it expresses. In Section 5 we discuss how we can effectively reduce the size of the corpus by filtering-out redundant reviews. In Section 6 we propose a review-selection mechanism for the evaluation of attribute-based queries. Then, in Section 7, we conduct a thorough experimental evaluation of the methods proposed in our paper. Finally, we conclude in Section 8 with a brief discussion of the paper.

2 Background

Our work is the first to formalize and address the Confident Search paradigm for review corpora. Even though there has been progress in relevant areas individually, ours is the first work to synthesize elements from all of them toward a customized search engine for review corpora. Next, we review the relevant work from various fields.

Review Assessment: Some work has been devoted on the evaluation of review *helpfulness* [21,13], formalizing the problem as one of regression. Jindal and Liu [10] also adopt an approach based on regression, focusing on the detection of spam (e.g. duplicate reviews). Finally, Liu and Cao [12] formulate the problem as binary classification, assigning a quality rating of "high" or "low" to reviews. Our concept of review assessment differs dramatically from the above-mentioned approaches: first, our framework has no requirement of tagged training data (e.g. spam/not spam, helpful/not helpful). Second, our work is the first to address redundant reviews in a principled and effective manner (Section 5). In any case, we consider prior work on review assessment complementary to ours, since it can be used to filter spam before the application of our framework.

Sentiment Analysis: Our work is relevant to the popular field of sentiment analysis, which deals with the extraction of knowledge from opinionated text. The domain of customer reviews is a characteristic example of such text, that has attracted much attention in the past [1,6,8,14,15,19]. A particularly interesting area of this field is that of attribute and opinion mining, which we discuss next in more detail.

Attribute and Opinion Mining: Given a review corpus on an item, opinion mining [9,17,7,18], looks for the attributes of the item that are discussed in each review, as well as the polarities (i.e. positive/negative) of the opinions expressed on each attribute. For our experiments, we implemented the technique proposed by Hu and Liu [9]: given a review corpus $\mathcal{R}$ on an item, the technique extracts the set of the item's attributes $\mathcal{A}$, and also identifies opinions of the form $(a \rightarrow p), p \in \{-1 + 1\}, \alpha \in \mathcal{A}$ in each review. We refer the reader to the original paper for further details. Even though this method worked superbly in practice, it is important to note that our framework is compatible with *any* method for attribute and opinion extraction.

Opinion Summarization: In the field of opinion summarization [12,22,11], the given review corpus is processed to produce a cumulative summary of the expressed opinions. The produced summaries are statistical in nature, offering information on the distribution of positive and negative opinions on the attributes of the reviewed item. We consider this work complementary to our own: we present an efficient search engine, able to select a minimal set of actual reviews in response to a specific query of attributes. This provides the user with actual comments written by humans, instead of a less user-friendly and intuitive statistical sheet.

3 Efficient Confident Search

Next, we formalize the *Confident Search* paradigm for large review corpora. We begin with an example, shown in Figure (3). The figure shows the attribute-set and the available review corpus $\mathcal{R}$ for a laptop computer. Out of the 9 available attributes, a user selects only those that interest him. In this case: {"Hard Drive", "Price", "Processor", "Memory"}. Given this query, our search engine goes through the corpus and selects a set of reviews $\mathcal{R}^* = \{r_1, r_7, r_9, r_{10}\}$ that accurately evaluates the specified attributes. Taking this example into consideration, we can now define the three requirements that motivate our concept of *Confident Search*:

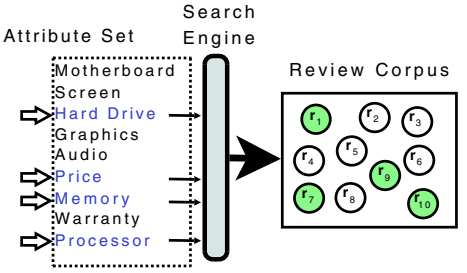


Fig. 3. A use case of our search engine: The user submits a query of 4 attributes, selected from the attribute-set of a computer. Then, the engine goes through a corpus of reviews and locates those that best cover the query (highlighted circles).

1. **Quality:** Given a query of attributes, a user should be presented with a set of high-quality reviews that accurately evaluates the attributes in the query.
2. **Efficiency:** The search engine should minimize the time required to evaluate a query, by appropriately pre-processing the corpus and eliminating redundancy.
3. **Compactness:** The set of retrieved reviews should be informative but also compact, so that a user can read through it in a reasonable amount of time.

Next, we will go over each of the three requirements, and discuss how they are addressed in our framework.

4 Quality through Confidence

We address the requirement for quality by introducing the concept of *confidence* in the opinions expressed within a review. Intuitively, a high-confidence review is one that provides accurate information on the item's attributes. Formally:

[Review Confidence Problem]: Given a review corpus $\mathcal{R}$ on an item, we want to define a function $conf(r, \mathcal{R})$ that maps each review $r \in \mathcal{R}$ to a score, representing the overall confidence in the *opinions* expressed within r.

Let $\mathcal{A}$ be the set of attributes of the reviewed item. Then, an *opinion* refers to one of the attributes in $\mathcal{A}$, and can be either positive or negative. Formally, we define an opinion as a mapping $(\alpha \rightarrow p)$ of an attribute $\alpha \in \mathcal{A}$ to a polarity $p \in \{-1, +1\}$. In our experiments, we extract the set of attributes $\mathcal{A}$ and the respective opinions using the method proposed in [9]. Further, let $\mathcal{O}^-_{r,a}$ and $\mathcal{O}^+_{r,a}$ represent the sets of negative and positive opinions expressed on an attribute α in review r, respectively. Then, we define $pol(\alpha, r)$ to return the *polarity* of α in r. Formally:

$$pol(\alpha, r) = \left\{ \begin{array}{l} +1, \text{ if } |\mathcal{O}^+_{r,\alpha}| > |\mathcal{O}^-_{r,\alpha}| \\ \\ -1, \text{ if } |\mathcal{O}^+_{r,\alpha}| < |\mathcal{O}^-_{r,\alpha}| \end{array} \right\} \quad (1)$$

Note that, for $|\mathcal{O}^{+}_{r,\alpha}| = |\mathcal{O}^{-}_{r,\alpha}|$, we simply ignore α, since the expressed opinion is clearly ambiguous. Now, given a review corpus $\mathcal{R}$ and an attribute α, let $n(\alpha \rightarrow p, \mathcal{R})$ be equal to the number of reviews in $\mathcal{R}$, for which $pol(\alpha, r) = p$. Formally:

$$n(\alpha \rightarrow p, \mathcal{R}) = |\{r : pol(\alpha, r) = p,\ r \in \mathcal{R}\}| \tag{2}$$

For example, if the item is a TV, then $n(\text{“screen”} \rightarrow +1, \mathcal{R})$ would return the number of reviews in $\mathcal{R}$ that express a positive opinion on its screen. Given Eq. (2), we can define the concept of the *consensus* of the review-corpus $\mathcal{R}$ on an attribute α as follows:

Definition 1 [Consensus]: Given a set of reviews $\mathcal{R}$ and an attribute α, we define the *consensus of* $\mathcal{R}$ *on* α as:

$$C_{\mathcal{R}}(a) = \underset{p \in \{-1,+1\}}{\operatorname{argmax}}\ n(\alpha \rightarrow p, \mathcal{R}) \tag{3}$$

Conceptually, the consensus expresses the polarity $\in \{-1, +1\}$ that was assigned to the attribute by the majority of the reviews. Formally, given a review corpus $\mathcal{R}$ and an opinion $\alpha \rightarrow p$, we define the *strength* $d(\alpha \rightarrow p, \mathcal{R})$ of the opinion as follows:

$$d(\alpha \rightarrow p, \mathcal{R}) = n(\alpha \rightarrow p) - n(\alpha \rightarrow -p) \tag{4}$$

Since the consensus expresses the majority, we know that $d(\alpha \rightarrow C_{\mathcal{R}}(\alpha), \mathcal{R}) \geq 0$. Further, the higher the value of $d(\alpha \rightarrow C_{\mathcal{R}}(\alpha))$, the higher is our confidence in the consensus. Given Eq. (4), we can now define the overall confidence in the opinions expressed within a given review. Formally:

Definition 2 [Review Confidence]: Given a review corpus $\mathcal{R}$ on an item and the set of the item's attributes $\mathcal{A}$, let $\mathcal{A}_r \subseteq \mathcal{A}$ be the subset of attributes that are actually evaluated within a review $r \in \mathcal{R}$. Then, we define the overall *confidence* of r as follows:

$$conf(r, \mathcal{R}) = \frac{\sum_{\alpha \in \mathcal{A}_r} d(\alpha \rightarrow pol(\alpha, r), \mathcal{R})}{\sum_{\alpha \in \mathcal{A}_r} d(\alpha \rightarrow C_{\mathcal{R}}(\alpha), \mathcal{R})} \tag{5}$$

The confidence in a review takes values in $[-1, 1]$, and is maximized when all the opinions expressed in the review agree with the consensus (i.e. $pol(\alpha, r) = C_{\mathcal{R}}(\alpha), \forall \alpha \in \mathcal{A}_r$). By dividing by the sum of the confidence values in the consensus on each $\alpha \in \mathcal{A}_r$, we ensure that the effect of an opinion $(\alpha \rightarrow p)$ on the confidence of r is proportional to the strength of the consensus on attribute α.

High-confidence reviews are more trustworthy and preferable sources of information, while those with low confidence values contradict the majority of the corpus. The confidence scores are calculated offline and are then stored and readily available for the search engine to use on demand.

5 Efficiency through Filtering

In this Section, we formalize the concept of *redundancy* within a set of reviews and propose a filter for its elimination. As we show with experiments on real datasets, the filter can drastically reduce the size of the corpus. The method is based on the following observation:

Observation 1. Given two reviews r_1 and r_2 in a corpus $\mathcal{R}$, let $\mathcal{A}_{r1} \subseteq \mathcal{A}_{r2}$ and $pol(\alpha, r_1) = pol(\alpha, r_2), \forall \alpha \in \mathcal{A}_{r1}$. Further, let $conf(r_1, \mathcal{R}) \leq conf(r_2, \mathcal{R})$. Then r_1 is redundant, since r_2 expresses the same opinions on the same attributes, while having a higher confidence score.

According to Observation 1, some of the reviews in the corpus can be safely pruned, since they are *dominated* by another review. This formulation matches the definition of the well-known *Skyline* operator [2][16][4], formally defined as follows:

Definition 3 [Skyline]: Given a set of multi-dimensional points $\mathcal{K}$, $Skyline(\mathcal{K})$ is a subset of $\mathcal{K}$ such that, for every point $k \in Skyline(\mathcal{K})$, there exists no point $k' \in \mathcal{K}$ that *dominates* k. We say that k' *dominates* k, if k' is no worse than k in all dimensions.

The computation of the skyline is a highly-studied problem, that comes up in different domains [16]. In the context of our problem, the set of dimensions is represented by the set of possible opinions $\mathcal{O}_\mathcal{R}$ that can be expressed within a review corpus $\mathcal{R}$. In the general skyline scenario, a point can assume any value in any of its multiple dimensions. In our case, however, the value of a review $r \in \mathcal{R}$ with respect to an opinion $op \in \mathcal{O}_\mathcal{R}$ can only assume one of two distinct values: if the opinion is actually expressed in r, then the value on the respective dimension is equal to $conf(r, \mathcal{R})$. Otherwise, we assign a value of -1, which is the minimum possible confidence score for a review. This ensures that a review r_1 can never be dominated by another review r_2, as long as it expresses at least one opinion that is not expressed in r_2 (since the value of r_2 for the respective dimension will be the lowest possible, i.e. -1).

Most skyline algorithms employ multi-dimensional indexes and techniques for high-dimensional search. However, in a constrained space such as ours, such methods lose their advantage. Instead, we propose a simple and efficient approach that is customized for our problem. The proposed method, which we refer to as `ReviewSkyline`, is shown in Algorithm (2).

Analysis of Algorithm (2): The input consists of a review corpus $\mathcal{R}$, along with the confidence score of each review $r \in \mathcal{R}$ and the set of possible opinions $O_\mathcal{R}$. The output is the skyline of $\mathcal{R}$.

Lines [1-2]: The algorithm first sorts the reviews in descending order by confidence. This requires $O(|\mathcal{R}| \log |\mathcal{R}|)$ time. It then builds an inverted index, mapping each opinion to the list of reviews that express it, sorted by confidence. Since we already have a sorted list of all the review from the previous step, this can be done in $O(|\mathcal{R}| \times M)$ time, where M is the size of the review with the most opinions in $\mathcal{R}$.

Lines [3-15]: The algorithm iterates over the reviews in $\mathcal{R}$ in sorted order, eliminating reviews that are dominated by the current Skyline. In order to efficiently check for this, we keep he reviews in the Skyline sorted by confidence. Therefore, since a review can only be dominated by one of higher or equal confidence, a binary search probe is used to check if a review r is dominated.

In line (6), we define a collection of lists $\mathcal{L} = \{L[op] | \forall op \in \mathcal{O}_r\}$, where $L[op]$ is the sorted list of reviews that express the opinion *op* (from the inverted index created in line (2)). The lists in $\mathcal{L}$ are searched in a round-robin fashion: the first $|\mathcal{L}|$ reviews to be

Algorithm 2. `ReviewSkyline`

Input: review corpus $\mathcal{R}$, $conf(r, \mathcal{R}) \forall r \in \mathcal{R}$, set of possible opinions $\mathcal{O}_{\mathcal{R}}$
Output: $Skyline$ of $\mathcal{R}$

1: Sort all reviews in $\mathcal{R}$ in descending order by $conf(r, \mathcal{R})$
2: Create an Inverted Index, mapping each opinion $op \in \mathcal{O}_{\mathcal{R}}$ to a list $L[op]$ of the reviews that express it, sorted by confidence.
3: **for** every review $r \in \mathcal{R}$ **do**
4: **if** (r is dominated by some set in $Skyline$) **then**
5: GOTO 3: // skip r
6: $\mathcal{L} = \{L[op] \mid \forall o \in \mathcal{O}_r\}$
7: **while** (NOT all Lists in $\mathcal{L}$ are exhausted) **do**
8: **for** every opinion $op \in \mathcal{O}_r$ **do**
9: $r' = getNext(L[op])$
10: **if** $(conf(r, \mathcal{R}) < conf(r', \mathcal{R}))$ **then**
11: Consider $L[op]$ to be exhausted
12: GOTO 8:
13: **if** (r' dominates r) **then**
14: GOTO 3: // skip r
15: $Skyline \leftarrow Skyline \cup \{r\}$
16: **return** $Skyline$

checked are those that are ranked first in each of the lists. We then check the reviews ranked 2nd and continue until all the lists have been exhausted.

The $getNext(L[op])$ routine returns the next review r' to be checked from the given list. If r' has a lower confidence than r, then we can safely stop checking $L[op]$, since any sets ranked lower will have an even lower score. Therefore, $L[op]$ is considered exhausted and we go back to check the list of the next opinion. If r' dominates r, we eliminate r and go back to examine the next review. If all the lists in $\mathcal{L}$ are exhausted without finding any review that dominates r, then we add it to the skyline.

Performance: In the worst case, all the reviews represent skyline points. Then, the complexity of the algorithm is quadratic in the number of reviews. In practice, however, the skyline includes only a small subset of the corpus. We demonstrate this on real datasets in the experiments section. We also show that `ReviewSkyline` is several times faster and more scalable than the state-of-the art for the general skyline computation problem. In addition, by using an inverted index instead of the multi-dimensional index typically employed by skyline algorithms, `ReviewSkyline` saves both memory and computational time.

6 Compactness through Selection

The requirement for compactness implies that simply evaluating the quality of the available reviews is not enough: top-ranked reviews may still express identical opinions on the same attributes and, thus, a user may have to read through a large number of reviews in order to obtain all the required information. Instead, given a query of attributes, a review should be included in the result, *only if it evaluates at least one attribute that is*

not evaluated in any of the other included reviews. Note that our problem differs significantly from conventional document retrieval tasks: instead of independently evaluating documents with respect to a given query, we want a *set* of reviews that collectively cover a subset of item-features. In addition, we want the returned set to contain opinions that respect the consensus reached by the reviewers on the specified features. Taking this into consideration, we define the *Review Selection Problem* as follows:

Problem 1 [Review Selection Problem]: Given the review corpus $\mathcal{R}$ on an item and a subset of the item's attributes $\mathcal{A}^* \subseteq \mathcal{A}$, find a subset $\mathcal{R}^*$ of $\mathcal{R}$, such that:

1. All the attributes in $\mathcal{A}^*$ are covered in $\mathcal{R}^*$
2. $pol(\alpha, r) = C_{\mathcal{R}}(\alpha), \forall \alpha \in \mathcal{A}^*, r \in \mathcal{R}^*$.
3. Let $\mathcal{X} \subseteq 2^{\mathcal{R}}$ be the collection of review-subsets that satisfy the first 2 conditions. Then:
   $\mathcal{R}^* = \underset{\mathcal{R}' \in \mathcal{X}}{\operatorname{argmax}} \sum_{r \in \mathcal{R}'} conf(r, \mathcal{R}')$

The 1st condition is straightforward. The 2nd condition ensures that the selected reviews contain no opinions that contradict the consensus on the specified attributes, in order to avoid selecting reviews with contradictory opinions. Finally, the 3rd condition asks for the set with the maximum overall confidence, among those that satisfy the first 2 conditions.

Ambiguous attributes: For certain attributes, the number of negative opinions may be only marginally higher than the number of positive ones (or vice versa), leading to a weak consensus. In order to identify such attributes, we define the $weight$ of an attribute α to be proportional to the $strength$ of its respective consensus (defined in Eq. (4)). Formally, given a review corpus $\mathcal{R}$ and an attribute α, we define $w(\alpha, \mathcal{R})$ as follows:

$$w(\alpha, \mathcal{R}) = \frac{d(\alpha \rightarrow C_{\mathcal{R}}(\alpha), \mathcal{R})}{|\mathcal{R}|} \tag{6}$$

Observe that, since $0 \leq d(\alpha \rightarrow C_{\mathcal{R}}(\alpha) \leq |\mathcal{R}|$, we know that $w(\alpha, \mathcal{R})$ takes values in $[0, 1]$. Conceptually, a low weight shows that the reviews on the specific attribute are mixed. Therefore, a set of reviews that contains only positive (or negative) opinions will not deliver a complete picture to the user. To address this, we relax the 2nd condition as follows: if the weight of an attribute α is less than some pre-defined lower bound b (i.e. $w(\alpha, \mathcal{R}) < b$), then the reported set $\mathcal{R}^*$ will be allowed to include reviews that contradict the (weak) consensus on α. In addition, $\mathcal{R}^*$ will be $required$ to contain at least one positive and one negative review with respect to α. The value of b depends on our concept of a weak consensus. For our experiments, we used $b = 0.5$.

6.1 A Combinatorial Solution

Next, we propose a combinatorial solution for the Review Selection problem. We show that the problem can be mapped to the popular Weighted Set Cover problem [3,5] (`WSC`), from which we can leverage solution techniques. Formally, the `WSC` problem is defined as follows:

Routine 3. `Transformation Routine`

Input: Set of attributes $\mathcal{A}$, Set of reviews $\mathcal{R}$
Output: Collection of subsets $\mathcal{S}$, $cost[s] \forall s \in \mathcal{S}$

for (every review $r \in \mathcal{R}$) **do**
  $s \leftarrow \emptyset$ // New empty set
  for (every attribute $\alpha \in \mathcal{A}$) **do**
    if $pol(\alpha, r) = +1$ **then** $s \leftarrow s \cup \{\alpha^+\}$
    else if $pol(\alpha, r) = -1$ **then** $s \leftarrow s \cup \{\alpha^-\}$
  $cost[s] \leftarrow (1 - conf(r, \mathcal{R}))/2$
  $\mathcal{S}.add(s)$
return $\mathcal{S}, cost[\,]$

[Weighted Set Cover Problem]: We are given a universe of elements $\mathcal{U} = \{e_1, e_2, \ldots, e_n\}$ and a collection $\mathcal{S}$ of subsets of $\mathcal{U}$, where each subset $s \in \mathcal{S}$ has a positive cost $cost[s]$. The problem asks for a collection of subsets $\mathcal{S}^* \subseteq \mathcal{S}$, such that $\bigcup_{s \in \mathcal{S}^*} \{s\} = \mathcal{U}$ and the cost $\sum_{s \in \mathcal{S}^*} cost[s]$ is minimized.

Given a review corpus $\mathcal{R}$, Routine (3) is used to generate a collection of sets $\mathcal{S}$, including a set s for every review $r \in \mathcal{R}$. The produced sets consist of elements from the same universe and have their respective costs, as required by the `WSC` problem.

Algorithm 1. `Greedy-Reviewer`

Input: $\mathcal{S}, \mathcal{A}^* \subseteq \mathcal{A}$, lower bound b
Output: weighted set-cover $\mathcal{S}^*$

$\mathcal{U} \leftarrow \emptyset$
for every attribute $\alpha \in \mathcal{A}^*$ **do**
  if $w(\alpha, \mathcal{R}) < b$ **then** $\mathcal{U} \leftarrow \mathcal{U} \cup \{\alpha^+\} \cup \{\alpha^-\}$
  else if $C_{\mathcal{R}}(\alpha) = +1$ **then** $\mathcal{U} \leftarrow \mathcal{U} \cup \{\alpha^+\}$
  else $\mathcal{U} \leftarrow \mathcal{U} \cup \{\alpha^-\}$
$\mathcal{S}^* \leftarrow \emptyset$ `// The set-cover`
$\mathcal{Z} \leftarrow \emptyset$ `// The still-uncovered part of` $\mathcal{U}$
while ($\mathcal{S}^*$ is not a cover of $\mathcal{U}$) **do**
  $s \leftarrow \underset{s' \in \mathcal{S},\ s' \cap \mathcal{U} = \emptyset}{\operatorname{argmin}} \left(\frac{cost[s']}{|s' \cap \mathcal{Z}|} \right)$
  $\mathcal{S}^*.add(s)$
return $\mathcal{S}^*$

The `Greedy-Reviewer` Algorithm: Next, we present an algorithm that can efficiently solve the Review Selection problem. The input consists of the collection of sets $\mathcal{S}$ returned by the transformation routine, a query of attributes $\mathcal{A}^* \subseteq \mathcal{A}$, and a number $b \in [0, 1]$, used to determine if the consensus on an attribute is weak (as described earlier in this section). The algorithm returns a subset $\mathcal{S}^*$ of $\mathcal{S}$. The pseudocode is given in Algorithm (1).

The Algorithm begins by populating the universe $\mathcal{U}$ of elements to be covered (lines 2-6). For each attribute $\alpha \in \mathcal{A}^*$, if the consensus on the attribute is weak ($w(\alpha, \mathcal{R}) < b$), two elements α^+ and α^- are added to $\mathcal{U}$. Otherwise, if the consensus is strong and positive (negative), an element α^+ (α^-) is added.

The universe of elements $\mathcal{U}$, together with the collection of sets $\mathcal{S}$, constitute an instance of the `WSC` Problem. The problem is known to be NP-Hard, but can be approximated by a well-known Greedy algorithm, with an $\ln n$ approximation ratio [5]. First, we define 2 variables $\mathcal{S}^*$ and $\mathcal{Z}$ to maintain the final solution and the still-uncovered subset of $\mathcal{U}$, respectively. The greedy-choice is conducted in lines 9-11: the algorithm selects the set that minimizes the quotient of the cost, over the still-uncovered part of $\mathcal{U}$ that is covered by the set. Since there is a 1-to-1 correspondence between sets and reviews, we can trivially obtain the set of selected reviews $\mathcal{R}^*$ from the reported set-cover $\mathcal{S}^*$ and return it to the user.

7 Experiments

In this section, we present the experiments we conducted toward the evaluation of our search framework. We begin with a description of the used datasets. We then proceed to discuss the motivation and setup of each experiment, followed by a discussion of the results. All experiments were run on a desktop with a Dual-Core 2.53GHz Processor and 2G of RAM.

7.1 Datasets

• `GPS`: For this dataset, we collected the complete review corpora for 20 popular GPS Systems from Amazon.com. The average number of reviews per item was 203.5. For each review, we extracted the stars rating, the date the review was submitted and the review content.

• `TVs`: For this dataset, we collected the complete review corpora for 20 popular TV Sets from Amazon.com. The average number of reviews per item was 145. For each review, we extracted the same information as in the `GPS` dataset.

• `Vegas-Hotels`: For this dataset, we collected the review corpora for 20 popular Las Vegas Hotels from yelp.com. Yelp is a popular review-hosting website, where users can evaluate business and service providers from different parts of the United States. The average number of reviews per item was 266. For each review, we extracted the content, the stars rating and the date of submission.

• `SF-Restaurants`: For this dataset, we collected the reviews for 20 popular San Francisco restaurants from yelp.com. The average number of reviews per item was 968. For each review, we extracted the same information as in the `Vegas-Hotels` dataset. **The data is available upon request.**

7.2 Qualitative Evidence

We begin with some qualitative results, obtained by using the proposed search framework on real data. For lack of space, we cannot present the sets of reviews reported for numerous queries. Instead, we focus on 2 indicative queries, 1 from

SF-Restaurants and 1 from Vegas-Hotels. For reasons of discretion, we omit the names of the specific items. For each item, we present the query, as well as the relevant parts of the retrieved reviews.

SF-Restaurants

Item 1, Query: {*food, service, atmosphere, restrooms*} 3 Reviews:

- *"...The dishes were creative and delicious ... The only drawback was the single unisex restroom."*

- *"Excellent food, excellent service. Only taking one star for the size and cramp seating. The wait can get long, and i mean long..."*

- *"... Every single dish is amazing. Solid food, nice cozy atmosphere, extremely helpful waitstaff, and close proximity to MY house..."*

Item 2, Query: {*location, price, music*}, 2 Reviews:

- *"...Great location, its across from 111 Minna. Considering the decor, the prices are really reasonable...."*

- *"..Another annoying thing is the noise level. The music is so loud that it's really difficult to have a conversation..."*

Vegas-Hotels

Item 3, Query: {*pool, location, rooms*}, 1 Review:

- *"...It was also a fantastic location, right in the heart of things...The pool was a blast with the eiffel tower overlooking it with great frozen drinks and pool side snacks. The room itself was perfectly fine, no complaints."*

Item 4, Query: {*pool, location, buffet, staff* }, 2 Reviews:

- *"This is one of my favorite casinos on the strip; good location; good buffet; nice rooms; nice pool(s); huge casino..."*

- *"...The casino is huge and there is an indoor nightclub on the ground floor. All staff are professional and courteous..."*

As can be seen from the results, our engine returns a compact set of reviews that accurately captures the consensus on the query-attributes and, thus, serves as a valuable tool for the interested user.

7.3 Skyline Pruning for Redundant Reviews

In this section, we present a series of experiments for the evaluation of the redundancy filter described in Section 5.

Number of Pruned Reviews: First, we examine the percentage of reviews that are discarded by our filter: for every item in each of the 4 datasets, we find the set of reviews that represents the skyline of the item's review corpus. We then calculate the average percentage of pruned reviews (i.e. reviews not included in the skyline), taken over

all the items in each dataset. The computed values for TVs, GPS, Vegas-Hotels and SF-Restaurants were $0.4, 0.47, 0.54$ and 0.79, respectively. The percentage of pruned reviews reaches up to 79%. This illustrates the redundancy in the corpora, with numerous reviewers expressing identical opinions on the same attributes. By focusing on the skyline, we can drastically reduce the number of reviews and effectively reduce the query response time.

Evolution of the Skyline: Next, we explore the correlation between the size of the skyline and the size of the review corpus, as the latter grows over time. First, we sort the reviews for each item in ascending order, by date of submission. Then, we calculate the cardinality of the skyline of the first K reviews. We repeat the process for $K \in \{50, 100, 200, 400\}$. For each value of K, we report the average percentage of the reviews that is covered by the skyline, taken over all the items in each dataset. The results are shown in Table 1.

Table 1. Skyline Cardinality Vs. Total #Reviews

	Avg #Reviews in the Skyline (Per Item)			
#Reviews	TVs	GPS	Vegas-Hotels	SF-Restaurants
50	0.64	0.53	0.47	0.35
100	0.56	0.47	0.44	0.28
200	0.55	0.43	0.4	0.24
400	0.55	0.43	0.39	0.19

The table shows that the introduction of more reviews has a decreasing effect on the percentage of the corpus that is covered by the skyline, which converges after a certain point. This is an encouraging finding, indicating that a compact skyline can be extracted regardless of the size of the corpus.

Running Time: Next, we evaluate the performance of the ReviewSkyline algorithm (Section 5). We compare the required computational time against that of the state-of-the-art Branch-and-Bound Algorithm (BnB) by Papadias et al. [16]. Our motivation is to show how our specialized algorithm compares to one made for the general problem.

The results, shown in Table 2, show that ReviewSkyline achieved superior performance in all 4 datasets. BnB treats each corpus as a very-high dimensional dataset, assuming a new dimension for every distinct opinion. As a result, the computational time is dominated by the construction of the required R-tree structure, which is known to deteriorate for very high dimensions [20]. ReviewSkyline avoids these shortcomings by taking into consideration the constrained nature of the review space.

Table 2. Avg Running Time Skyline Computation (in seconds)

	TVs	GPS	Vegas-Hotels	SF-Restaurants
ReviewSkyline	0.2	0.072	0.3	0.11
BnB	24.8	39.4	28.9	116.2

Scalability: In order to demonstrate the scalability of `ReviewSkyline`, we created a benchmark with very large batches of artificial reviews. As a seed, we used the reviews corpus for the "slanted door" restaurant from the `SF-Restaurants` dataset, since it had the largest corpus across all datasets (about 1400 reviews). The data was generated as follows: first, we extracted the set $\mathcal{Y}$ of distinct opinions (i.e. attribute-to-polarity mappings) from the corpus, along with their respective frequencies. A total of 25 distinct attributes were extracted from the corpus, giving us a set of 50 distinct opinions. In the context of the skyline problem, this number represents the dimensionality of the data.

Each artificial review was then generated as follows: first, we flip an unbiased coin. If the coin comes up heads, we choose an opinion from $\mathcal{Y}$ and add it to the review. The probability of choosing an opinion from $\mathcal{Y}$ is proportional to its frequency in the original corpus. We flip the coin 10 times. Since the coin is unbiased, the expected average number of opinions per review of is 5, which is equal to the actual average observed in the corpus. We created 6 artificial corpora, where each corpus had a population of p reviews, $p \in \{10^4, 2 \times 10^4, 4 \times 10^4, 8 \times 10^4, 16 \times 10^4\}$. We compare `ReviewSkyline` with the `BnB` Algorithm, as we did in the previous experiment. The Results are shown in Figure (4). The entries on the x-axis represent the 5 artificial corpora, while the values on the y-axis represent the computational time *(in logarithmic scale)*. The results show that `ReviewSkyline` achieves superior performance for all 5 corpora. The algorithm exhibited great scalability, achieving a low computational time even for the largest corpus (less than 3 minutes). In contrast to `ReviewSkyline`, `BnB` is burdened by the construction and poor performance of the R-tree in very high-dimensional datasets.

7.4 Query Evaluation

In this section, we evaluate the search engine described in Section 6. Given the set of attributes $\mathcal{A}$ of an item, we choose 100 subsets of $\mathcal{A}$, where each subset contains exactly k elements. The probability of including an attribute to a query is proportional to the attribute's frequency in the corpus. The motivation is to generate more realistic queries,

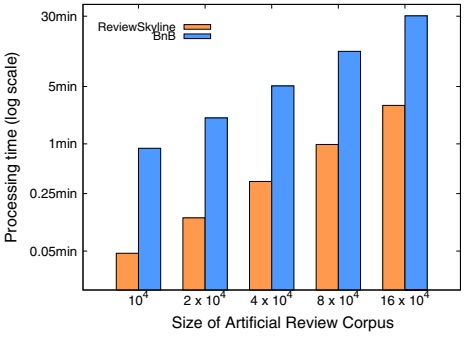
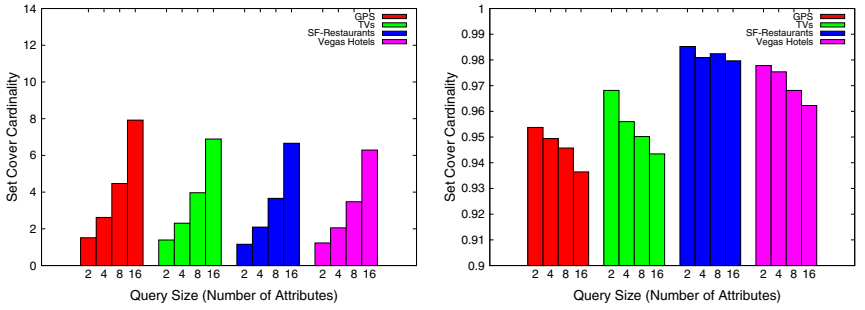


Fig. 4. Scalability of `ReviewSkyline` and `BnB`



(a) Avg. Result size (b) Avg. Confidence per review

Fig. 5. Figures (a) and (b) show the average number of reviews included in the result and the average confidence per reported review, respectively

since users tend to focus on the primary and more popular attributes of an item. We repeat the process for $k \in \{2, 4, 8, 16\}$, for a total of $100 \times 4 = 400$ queries per item.

Query size Vs. Result size: First, we evaluate how the size of the query affects the cardinality of the returned sets. Ideally, we would like to retrieve a small number of reviews, so that a user can read them promptly and obtain the required information. Given a specific item I and a query size k, let $Avg[I, k]$ be the average number of reviews included in the result, taken over the 100 queries of size k for the item. We then report the mean of the $Avg[I, k]$ values, taken over all 20 items in each dataset. The results are shown in Figure 5(a): The reported sets were consistently small, with less than 8 reviews were enough to cover queries containing up to 16 different attributes. Such compact sets are desirable since they can promptly be read by the user.

Query Size Vs. Confidence: Next, we evaluate how the size of the query affects the average confidence of the selected reviews. The experimental setup is similar to that of the previous experiment. However, instead of the average result cardinality, we report the average confidence per selected review. Figure 5(b) shows the very promising results. An average confidence of 0.93 or higher was consistently reported for all query sizes, and for all 4 datasets. Combined with the findings of the previous experiment, we conclude that our framework produces compact sets of high-quality reviews.

8 Conclusion

In this paper, we formalized the Confident Search paradigm for large review corpora. Taking into consideration the requirements of the paradigm, we presented a complete search framework, able to efficiently handle large sets of reviews. Our framework employs a principled method for evaluating the confidence in the opinions expressed in reviews. In addition, it is equipped with an efficient method for filtering redundancy. The filtered corpus maintains all the useful information and is considerably smaller, which makes it easier to store and to search. Finally, we formalized and addressed the problem of selecting a minimal set of high-quality reviews that can effectively cover any

query of attributes submitted by the user. The efficacy of our methods was demonstrated through a rigorous and diverse experimental evaluation.

References

1. Archak, N., Ghose, A., Ipeirotis, P.: Show me the money! Deriving the pricing power of product features by mining consumer reviews. In: SIGKDD (2007)
2. Börzsönyi, S., Kossmann, D., Stocker, K.: The skyline operator. In: ICDE (2001)
3. Caprara, A., Fischetti, M., Toth, P.: Algorithms for the set covering problem. Annals of Operations Research (1996)
4. Chomicki, J., Godfrey, P., Gryz, J., Liang, D.: Skyline with presorting. In: ICDE (2003)
5. Chvatal, V.: A greedy heuristic for the set-covering problem. Mathematics of Operations Research (1979)
6. Dave, K., Lawrence, S., Pennock, D.M.: Mining the peanut gallery: opinion extraction and semantic classification of product reviews. In: WWW 2003 (2003)
7. Ghani, R., Probst, K., Liu, Y., Krema, M., Fano, A.: Text mining for product attribute extraction. SIGKDD Explorations Newsletter (2006)
8. Hu, M., Liu, B.: Mining and summarizing customer reviews. In: SIGKDD (2004)
9. Hu, M., Liu, B.: Mining opinion features in customer reviews. In: AAAI (2004)
10. Jindal, N., Liu, B.: Opinion spam and analysis. In: WSDM 2008 (2008)
11. Ku, L.-W., Liang, Y.-T., Chen, H.-H.: Opinion extraction, summarization and tracking in news and blog corpora. In: AAAI Symposium on Computational Approaches to Analysing Weblogs, AAAI-CAAW (2006)
12. Liu, J., Cao, Y., Lin, C.-Y., Huang, Y., Zhou, M.: Low-quality product review detection in opinion summarization. In: EMNLP-CoNLL (2007)
13. Min Kim, S., Pantel, P., Chklovski, T., Pennacchiotti, M.: Automatically assessing review helpfulness. In: EMNLP 2006 (2006)
14. Pang, B., Lee, L.: Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales. In: ACL (2005)
15. Pang, B., Lee, L., Vaithyanathan, S.: Thumbs up? Sentiment classification using machine learning techniques. In: EMNLP (2002)
16. Papadias, D., Tao, Y., Fu, G., Seeger, B.: Progressive skyline computation in database systems. ACM Trans. Database Syst. (2005)
17. Popescu, A.-M., Etzioni, O.: Extracting product features and opinions from reviews. In: HLT 2005 (2005)
18. Riloff, E., Patwardhan, S., Wiebe, J.: Feature subsumption for opinion analysis. In: EMNLP (2006)
19. Turney, P.D.: Thumbs up or thumbs down? semantic orientation applied to unsupervised classification of reviews. In: ACL (2002)
20. Weber, R., Schek, H.-J., Blott, S.: A quantitative analysis and performance study for similarity-search methods in high-dimensional spaces. In: VLDB 1998 (1998)
21. Zhang, Z., Varadarajan, B.: Utility scoring of product reviews. In: CIKM (2006)
22. Zhuang, L., Jing, F., Zhu, X., Zhang, L.: Movie review mining and summarization. In: CIKM (2006)