Review Selection Using Micro-Reviews

Thanh-Son Nguyen, Hady W. Lauw, *Member, IEEE*, and Panayiotis Tsaparas, *Member, IEEE*

Abstract—Given the proliferation of review content, and the fact that reviews are highly diverse and often unnecessarily verbose, users frequently face the problem of selecting the appropriate reviews to consume. *Micro-reviews* are emerging as a new type of online review content in the social media. Micro-reviews are posted by users of check-in services such as Foursquare. They are concise (up to 200 characters long) and highly focused, in contrast to the comprehensive and verbose reviews. In this paper, we propose a novel mining problem, which brings together these two disparate sources of review content. Specifically, we use coverage of micro-reviews as an objective for selecting a set of reviews that cover efficiently the salient aspects of an entity. Our approach consists of a two-step process: matching review sentences to micro-reviews, and selecting a small set of reviews that cover as many micro-reviews as possible, with few sentences. We formulate this objective as a combinatorial optimization problem, and show how to derive an optimal solution using Integer Linear Programming. We also propose an efficient heuristic algorithm that approximates the optimal solution. Finally, we perform a detailed evaluation of all the steps of our methodology using data collected from Foursquare and Yelp.

Index Terms—Micro-review, review selection, coverage

1 Introduction

Today, we can find ample review content in various Web sources. For instance, Yelp.com is a popular site for restaurant reviews, assisting diners to plan restaurant visits. While useful, the deluge of online reviews also poses several challenges. Readers are inundated by the information overload, and it is becoming increasingly harder for them to weed out the reviews that are worthy of their attention. This is worsened by the length and verbosity of many reviews, whose content may not be wholly relevant to the product or service being reviewed. Reviewers often digress, detailing personal anecdotes that do not offer any insight about the item or place being reviewed. Furthermore, it is getting increasingly more difficult to determine whether a review has been written by a genuine customer, or by a spammer.[1] Identifying and selecting high quality, authentic reviews is a hard task, and it has been the focus of substantial amount of research [7], [11], [14], [16], [29].

With the recent growth of social networking and micro-blogging services, we observe the emergence of a new type of online review content. This new type of content, which we term *micro-reviews*, can be found in micro-blogging services that allow users to "check-in", indicating their current location or activity. For example, at Foursquare, users check in at local venues, such as restaurants, bars or coffee shops. After checking in, a user may choose to leave a message, up to 200 characters long, about their experience, effectively a *micro-review* of the place. In addition to Foursquare, there are also alternative sources for micro-reviews in several domains. For instance, Facebook Places, Jiepang (in Chinese) and VK (in Russian) feature similar services, while GetGlue (now tvtag) allows users to check in to TV shows, movies, or sports events, and Flicktweets to post micro-reviews on movies. Following the Foursquare terminology, we will refer to all micro-reviews as *tips*.

In the case of restaurants, tips are frequently recommendations (e.g., what to order), opinions (what is great or not), or actual "tips". For example, the following are some example Foursquare tips for a popular burger joint in New York: *"Order the Trifecta—a shack stack, cheese fries and a milk shake and then see if you don't get winded on your way home. Head there early to avoid the line."* (a recommendation), *"'SHROOM BURGER!!! Its the ONLY good veggie burger in the city!"* (an opinion), and *"If you only want ice cream there is a short line. The A line is for food and shakes and is long. The B line has little or no waiting for ice cream & floats."* (an actual tip).

Micro-reviews serve as an alternative source of content to reviews for readers interested in finding information about a place.[2] They have several advantages. *First*, due to the length restriction, micro-reviews are *concise and distilled*, identifying the most salient or pertinent points about the place. *Second*, because some micro-reviews are written on site, right when the user has checked in, they are *spontaneous*, expressing the author's immediate and unadulterated reaction to her experience. *Third*, because most authors check in by mobile apps, these authors are likely at the place when leaving the tips, which makes the tips more likely to be *authentic*. Micro-blogging sites also have the ability, if necessary, to filter out tips without an accompanying check-in, thus, boosting the authenticity of the tips.

Micro-reviews and reviews nicely complement each other. While reviews are lengthy and verbose, tips are short

1. http://www.businessweek.com/magazine/a-lie-detector-test-for-online-reviewers-09292011.html

2. http://www.fastcompany.com/3015168/foursquares-tips-growing-faster-than-yelps-reviews

• *T.-S. Nguyen and H.W. Lauw are with the School of Information Systems, Singapore Management University, 178902, Singapore, Singapore. E-mail: tsnguyen.2013@phdis.smu.edu.sg, hadywlauw@smu.edu.sg.*
• *P. Tsaparas is with the Department of Computer Science, University of Ioannina, Ioannina, Greece. E-mail: tsap@cs.uoi.gr.*

Manuscript received 1 Apr. 2014; revised 21 Aug. 2014; accepted 22 Aug. 2014. Date of publication 0 . 0000; date of current version 0 . 0000.
Recommended for acceptance by P.G. Ipeirotis.
For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org, and reference the Digital Object Identifier below.
Digital Object Identifier no. 10.1109/TKDE.2014.2356456

and concise, focusing on specific aspects of an item. At the same time, these aspects cannot be properly explored within 200 characters. This is accomplished in full-blown reviews which elaborate and contemplate on the intricacies of a specific characteristic. Marrying these two different reviewing approaches can yield something greater than the sum of their parts: detailed reviews that focus on aspects of a venue that are of true importance to users.

We consider the following problem. Given a collection of reviews, and a collection of tips about an item, we want to select a small number of reviews that best *cover* the content of the tips. This problem is of interest to any online site or mobile application that wishes to showcase a small number of reviews. For example, review sites such as Yelp, which recently introduced tips as part of their mobile application, would benefit from such a review selection mechanism. Similarly for review aggregation sites such as Google Local. The need for concise and comprehensive content becomes especially more pronounced for the mobile applications of such sites, where the screen real-estate is limited, and the user attention span is shorter.

The problem of review selection has been studied in the past [13], [14], [29]. In all prior work this is modeled as a *coverage problem*, where the selected reviews are required to cover the different aspects of the item (e.g., product attributes), and the polarity of opinions about the item (positive and negative). To extract the aspects covered by a review and the sentiment polarity off-the-shelf tools for supervised techniques are usually applied. Such approaches, although generally successful, cannot generalize to arbitrary domains. Unsupervised techniques, e.g., topic modeling, have also been applied (e.g., [18]), however they suffer from the broadness of the topic definition.

We view tips as a crowdsourced way to obtain the aspects of an item that the users care about, as well as the sentiment of the users. By covering the tips, we effectively identify the review content that is important, and the aspects of the item upon which the reviews need to expand and elaborate. In our formulation, which we outline below, we make sure that the selected reviews are compact, that is, the content does not diverge from what is important about the reviewed item. We view this as an important constraint, especially for viewing on mobile devices, where screens are small, and time is short.

Contributions. Although the content of micro-blogging sites has been studied extensively, micro-reviews is a source of content that has been largely overlooked in the literature. In this paper we study micro-reviews, and we show how they can be used for the problem of review selection. To the best of our knowledge we are the first to mine micro-reviews such as Foursquare tips and combine them with full-text reviews such as Yelp reviews. Our work introduces a novel formulation of review selection, where the goal is to maximize coverage while ensuring efficiency, leading to novel coverage problems. The coverage problems we consider are of broader interest, and they could find applications to different domains. We consider approximation and heuristic algorithms, and study them experimentally, demonstrating quantitatively and qualitatively the benefits of our approach. We also propose an Integer Linear Programming (ILP) formulation, and provide an optimal algorithm. This allows us to quantify the approximation quality of the greedy heuristics. We investigate the number of reviews needed to obtain perfect coverage through an alternative formulation inspired by set cover.

2 Overview

We begin with a high-level overview of our approach. For an entity (e.g., a restaurant), we assume we are given as input a collection of reviews $\mathcal{R}$ and a collection of tips $\mathcal{T}$ about the entity. Our goal is to select a subset of reviews $\mathcal{S} \subseteq \mathcal{R}$ that covers the set of tips $\mathcal{T}$ as concisely (efficiently) and as thoroughly as possible.

To perform the selection, we need to determine when a review $R \in \mathcal{R}$ covers a tip $t \in \mathcal{T}$. We refer to this procedure as *matching* reviews and tips. Given the matching, we then select a small subset of reviews that cover as many tips as possible. We refer to the number of covered tips as the selection *coverage*. We also introduce the notion of selection *efficiency*, which captures the principle that the selected set should not contain too many sentences that do not cover any tip.

2.1 Matching Reviews and Tips

Reviews and tips are of different granularity. A tip is short and concise, usually making a single point, while a review is longer and multi-faceted, discussing various aspects of an entity. Intuitively, a review covers a tip, if the point made by the tip appears within the text of the review. To make this more precise, we break a review into sentences, which are semantic units with granularity similar to that of the tips.

We view a review R as a set of sentences $R = \{s_1, \ldots, s_{|R|}\}$, and we use $\mathcal{U}_s$ to denote the union of all review sentences from the reviews in $\mathcal{R}$. We define a matching function $\mathcal{F} : \mathcal{U}_s \times \mathcal{T} \rightarrow \{0, 1\}$, where for a sentence $s \in \mathcal{U}_s$ and a tip $t \in \mathcal{T}$ we have:

$$\mathcal{F}(s,t) = \begin{cases} 1, & \text{if s and t are similar,} \\ 0, & \text{otherwise.} \end{cases}$$

We want to match a sentence s and a tip t if they convey a similar meaning, and therefore one can be seen as covering the content of the other. We consider the following three criteria for making the matching decision. The first criterion considers the sentence and the tip as bags of words. If they share a substantial subset of textual content then we assume that they convey a similar meaning. In this case we say that they have high *syntactic similarity*. The second criterion considers the concept that is discussed. A sentence and a tip may discuss the same concept (e.g., a menu dish), but use different words (e.g., soup vs. broth). In this case we say that they have high *semantic similarity*. Finally, reviews as well as tips, express the opinions of their respective authors. Hence, in addition to sharing similar keywords and concepts, we would also like a matching sentence-tip pair to share the same sentiment (positive or negative). In this case we say that they have high *sentiment similarity*. In Section 4, we elaborate further on each of the above three types of similarity, and how they can be defined and measured. We also describe how to combine them into a single matching function $\mathcal{F}$.

2.2 Selection Coverage

If a sentence s and a tip t are matched, then we say that s covers t. We will say that a review R covers a tip t if there is a sentence $s \in R$ that is matched to the tip t. Given the collection of reviews $\mathcal{R}$ and the collection of tips $\mathcal{T}$, and the matching function $\mathcal{F}$, we define for each review R the set of tips $\mathcal{T}_R$ that are *covered* by at least one sentence of review R. Formally:

$$\mathcal{T}_R = \{t \in \mathcal{T} : \exists s \in R, \mathcal{F}(s,t) = 1\}.$$

We say that R *covers* the tips in $\mathcal{T}_R$. We define the coverage $\mathrm{Cov}(R)$ of review R as the fraction of tips $|\mathcal{T}_\mathcal{R}|/|\mathcal{T}|$ covered by the review R.

We can extend this definition to the case of a *collection* of reviews. For a set of reviews $\mathcal{S} \subseteq \mathcal{R}$, we define the coverage of the set $\mathcal{S}$ as:

$$\mathrm{Cov}(\mathcal{S}) = \frac{|\cup_{R\in\mathcal{S}} \mathcal{T}_R|}{|\mathcal{T}|}$$

that is, the fraction of tips covered by the set $\mathcal{S}$.

2.3 Selection Efficiency

Some reviews may have high coverage, but at the same time they are too verbose, containing many sentences that are not relevant to any tip at all. We would like to avoid such reviews in our selection, so we introduce the concept of *efficiency*. For a review R, let R^r be the set of "relevant" sentences which cover at least one tip, i.e., $R^r = \{s \in R : \exists t \in \mathcal{T}_R, \mathcal{F}(s,t) = 1\}$. We define the *efficiency* $\mathrm{Eff}(R)$ of the review R as the fraction of "relevant" sentences in R. Formally:

$$\mathrm{Eff}(R) = \frac{|R^r|}{|R|}.$$

Extending the definition of efficiency to a collection of reviews is a little more involved. We need a way to aggregate the efficiency of the individual reviews. We propose three possible definitions.

- *Minimum efficiency*. In this case, the efficiency of a set of reviews $\mathcal{S}$ is defined as the minimum efficiency of any review in the set. Formally:
$$\mathrm{Eff}_{\min}(\mathcal{S}) = \min_{R\in\mathcal{S}} \mathrm{Eff}(R).$$
- *Average efficiency*. In this case, the efficiency of a set $\mathcal{S}$ is defined as the average efficiency of the reviews in the set. Formally:
$$\mathrm{Eff}_{\mathrm{avg}}(\mathcal{S}) = \frac{\sum_{R\in\mathcal{S}} \mathrm{Eff}(R)}{|\mathcal{S}|}.$$
- *Bag efficiency*. In this case, we view a collection of reviews $\mathcal{S}$ as a single review $R_\mathcal{S}$ consisting of the union of the sentences of the reviews. We then define the efficiency of the collection as the efficiency of $R_\mathcal{S}$. Formally, we have $R_\mathcal{S} = \cup_{R\in\mathcal{S}} R$, and $\mathrm{Eff}_{\mathrm{bag}}(\mathcal{S}) = \mathrm{Eff}(R_\mathcal{S})$.

$\mathrm{Eff}_{\min}$ is useful for imposing a stringent condition on the efficiency of the reviews in the set $\mathcal{S}$. For instance, by requesting that the minimum efficiency is above some threshold, we gain a guarantee that all reviews in the set obey the threshold. The other two definitions $\mathrm{Eff}_{\mathrm{avg}}$ and $\mathrm{Eff}_{\mathrm{bag}}$ are more flexible, because they consider the set $\mathcal{S}$ as a whole. This allows us to select some reviews with high coverage but slightly lower efficiency, if we can balance this choice with other reviews with high efficiency in the set. $\mathrm{Eff}_{\mathrm{bag}}$ is different from $\mathrm{Eff}_{\mathrm{avg}}$ in that it effectively gives longer reviews a higher weight when computing the aggregate efficiency of a set.

2.4 Problem Statement

Ideally, there would be a small number of reviews with perfect coverage and efficiency. In practice, such an ideal set rarely exists, if ever. We formulate the selection problem as an optimization problem where we seek the best possible solution. However, optimizing both coverage and efficiency is a bi-criterion optimization problem, with no single optimal solution. We need to select one of the two metrics to optimize.

In most cases, perfect efficiency is not essential. There may exist a few sentences in a review that do not cover any tip on their own accord, but their presence may improve the readability of the review. It suffices to ensure that the efficiency does not fall below a certain minimum acceptable threshold. Therefore, we opt to view our problem as a maximization problem, where we constrain the efficiency, and we ask for a solution with maximum coverage.

Problem 1 (EffMaxCoverage). *Given a set of reviews $\mathcal{R}$, a set of tips $\mathcal{T}$, the matching function $\mathcal{F}$ between review sentences and tips, and parameters α and K, select a set $\mathcal{S}$ of K reviews such that the coverage $\mathrm{Cov}(\mathcal{S})$ of the set is maximized, while the efficiency of the set is at least α, that is $\mathrm{Eff}(\mathcal{S}) \geq \alpha$.*

In the above formulation, by setting the desired number K of reviews, we can ensure a concise representation of the tips. An alternative formulation is to not limit the number of reviews K, and instead try to obtain perfect coverage with the minimum number of reviews. This problem can be stated as follows.

Problem 2 (EffSetCover). *Given a set of reviews $\mathcal{R}$, a set of tips $\mathcal{T}$, the matching function $\mathcal{F}$ between review sentences and tips, and a parameter α, select a set $\mathcal{S} \subseteq \mathcal{R}$ of reviews which covers all the tips in $\mathcal{T}$, such that the efficiency of the set is at least α ($\mathrm{Eff}(\mathcal{S}) \geq \alpha$), and the size of the set $\mathcal{S}$ is minimized.*

We note that for the applications we consider (e.g., checking reviews on mobile devices), space and time resources are limited, so we need to select a small number of reviews to show to the user. We consider the EffSetCover formulation in order to quantify the minimal number of reviews necessary to cover all tips, and understand the tradeoff between coverage and efficiency. We compare the two alternative formulations experimentally in Section 5.3.1.

3 Algorithms

Ideally, we would like to solve the EffMaxCoverage problem optimally. In Section 3.1, we will show that it can be expressed as an Integer Linear Programming problem, for which there are known algorithms for deriving an optimal solution. However, the ILP formulation, while optimal, may not be tractable for cases where the number of reviews is very large. This is because EffMaxCoverage is NP-hard. This result follows from the fact that in the special case where $\alpha = 0$ (no efficiency constraint) the EffMaxCoverage is the same as the MaxCoverage problem, which is known to be NP-hard. Therefore, we also need to look for approximation, or heuristic algorithms, which we discuss in Section 3.2.

3.1 Finding the Optimal Solution

Our problem definition differs depending on the choice of the efficiency function. In the case that we use the $\mathrm{Eff}_{\min}$ function requiring that $\mathrm{Eff}_{\min}(\mathcal{S}) \geq \alpha$ each of the selected reviews must have individual efficiency at least α. Therefore, this is equivalent to the MaxCoverage problem, where the universe of available reviews is restricted to the subset of reviews that have efficiency at least α. There is a known ILP formulation [31] for the MaxCoverage that we can use for obtaining an optimal solution.

No similar equivalence could be established for the other two efficiency functions, $\mathrm{Eff}_{\mathrm{avg}}$ and $\mathrm{Eff}_{\mathrm{bag}}$; therefore, new solutions need to be developed. We will now show how to adapt the ILP formulations for MaxCoverage to take into account these efficiency constraints. We begin by describing the ILP formulation for the MaxCoverage problem. Let x_i be a binary integer variable associated with each review R_i, with $x_i = 1$ denoting that R_i is in the selected set, and $x_i = 0$ otherwise. Let y_j be a binary integer variable associated with each tip t_j, with $y_j = 1$ denoting that the tip t_j is covered by at least one of the reviews in the selected set, and $y_j = 0$ otherwise.

We express the problem as a set of constraints:

$$\text{maximize} \sum_{j=1}^{m} y_j \tag{1}$$

$$\text{subject to} \sum_{i=1}^{n} x_i \leq K \tag{2}$$

$$\sum_{i:t_j \in \mathcal{T}_{R_i}} x_i \geq y_j \quad \forall t_j \in \mathcal{T} \tag{3}$$

$$x_i = \{0, 1\} \tag{4}$$

$$y_j = \{0, 1\}. \tag{5}$$

Equation (1) is an objective function that maximizes the number of tips covered. Constraint 2 ensures that the number of selected reviews does not exceed K. Constraint 3 ensures that if $y_j = 1$ (i.e., tip t_j is covered), then at least one review that covers t_j must be selected. Constraints 4 and 5 require that the variables take 0 or 1 values.

The above ILP formulation captures the optimization objective for EffMaxCoverage, but it does not take into account the efficiency constraints. It can be used in the case of the $\mathrm{Eff}_{\min}$ function, by pre-filtering reviews that do not meet the efficiency threshold. However, for the $\mathrm{Eff}_{\mathrm{avg}}$ and $\mathrm{Eff}_{\mathrm{bag}}$ functions, we need additional constraints. Fortunately, these efficiency constraints can be expressed as linear constraints as well, and thus fit naturally into the ILP framework.

- *Average efficiency.* Constraint 6 below ensures that the average precision of the selected reviews will be at least α.

$$\frac{\sum_{i=1}^{n} \frac{|R_i^r|}{|R_i|} x_i}{\sum_{i=1}^{n} x_i} \geq \alpha \Leftrightarrow \sum_{i=1}^{n} \left(\frac{|R_i^r|}{|R_i|} - \alpha \right) x_i \geq 0. \tag{6}$$

- *Bag Efficiency.* Constraint 7 ensures that the bag precision of the selected reviews will be at least α.

$$\frac{\sum_{i=1}^{n} |R_i^r| x_i}{\sum_{i=1}^{n} |R_i| x_i} \geq \alpha \Leftrightarrow \sum_{i=1}^{n} \big(|R_i^r| - \alpha |R_i| \big) \; x_i \geq 0 \tag{7}$$

We use *ILP-EffMaxCover*, indexed by the efficiency function that we consider, to refer to the ILP formulations. With the constraints in place, the formulations can be solved by existing solvers. In our experiments, we employ the publicly available lp_solve library.[3]

3.2 Greedy Selection

Since our problems are NP-hard, finding the optimal solution is not tractable for very large problem sizes. Therefore, we look for more efficient alternatives.

It is well known that due to the submodularity property of the coverage function, the greedy algorithm that always selects the review whose addition maximizes the coverage produces a solution with approximation ratio $(1 - \frac{1}{e})$ for the MaxCoverage problem, where e is the base of the natural logarithm [23]. That is, the coverage of the greedy algorithm is at least a $(1 - \frac{1}{e})$ fraction of the coverage of the optimal algorithm. Therefore, we obtain the following lemma.

Lemma 1. *The greedy algorithm for the* EffMaxCoverage *problem with the* $\mathrm{Eff}_{\min}$ *efficiency function has approximation ratio* $(1 - \frac{1}{e})$.

We could not determine an approximation bound for the other two variants of the efficiency function.

We now present a greedy algorithm for the EffMaxCoverage problem which, as a special case, includes the greedy approximation algorithm for $\mathrm{Eff}_{\min}$.

The algorithm, shown in Algorithm 1, proceeds in iterations each time adding one review to the collection $\mathcal{S}$. At each iteration, for each review R we compute two quantities. The first is the *gain* $\mathrm{gain}(R)$, which is the increase in coverage that we obtain by adding this review to the existing collection $\mathcal{S}$. The second quantity is the cost $\mathrm{cost}(R)$ of the review R, which is proportional to the *inefficiency* $1 - \mathrm{Eff}(R)$ of the review, that is, the fraction of sentences of R that are not matched to any tip. We select the review R^* that has the highest gain-to-cost ratio, and guarantees that the efficiency of the resulting collection is at least α, where α is a parameter

3. http://lpsolve.sourceforge.net/5.5/

provided in the input. The intuition is that reviews with high gain-to-cost ratio cover many additional tips, while introducing little irrelevant content, and thus they should be added to the collection.

Algorithm 1. *Greedy-EffMaxCover* algorithm.

```
Input:  Set of reviews $\mathcal{R}$ and tips $\mathcal{T}$; Efficiency function Eff;
        Integer budget value $K$, parameters $\alpha$, $\beta$.
Output: A set of reviews $\mathcal{S} \subseteq \mathcal{R}$ of size $K$.
 1:  $\mathcal{S} = \emptyset$
 2:  while $|\mathcal{S}| < K$ do
 3:      for all $R \in \mathcal{R}$ do
 4:          $\text{gain}(R) = \text{Cov}(\mathcal{S} \cup R) - \text{Cov}(\mathcal{S})$
 5:          $\text{cost}(R) = \beta(1 - \text{Eff}(R)) + (1 - \beta)$.
 6:      end for
 7:      $\mathcal{E} = \{R \in \mathcal{R} : \text{Eff}(\mathcal{S} \cup R) \geq \alpha\}$
 8:      if ($\mathcal{E} == \emptyset$) or ($\max_{R\in\mathcal{E}} \text{gain}(R) == 0$) then
 9:          break
10:      end if
11:      $R^* = \arg\max_{R\in\mathcal{E}} \text{gain}(R)/\text{cost}(R)$
10:      $\mathcal{S} = \mathcal{S} \cup R^*$
11:      $\mathcal{R} = \mathcal{R} \setminus R^*$
12:  end while
13:  return $\mathcal{S}$
```

The cost of the review is parameterized by a value $\beta \in [0, 1)$, provided as part of the input, which controls the effect of efficiency in our selection of the review R^*. More specifically, the cost of a review is defined as follows:

$$\text{cost}(R) = \beta(1 - \text{Eff}(R)) + (1 - \beta).$$

When $\beta = 0$, the review selection is not affected by the efficiency of the reviews, but only by the coverage. For β close to 1 the effect of the efficiency on the review selection is maximized. Values in-between regulate the effect of efficiency in our selection. The higher the value of β, the higher the value of coverage that is needed for a low-efficiency review to be included in the set. For example, for $\beta = 1$, a review R_1 with efficiency 0.5 needs to have at least 250% times more coverage to be picked over another review R_2 with efficiency 0.8. For $\beta = 0.5$, R_1 only needs 25% more additional coverage to be picked over R_2.

We obtain different algorithms for different choices of the efficiency function. We study these different variations in detail in the experimental analysis. Note also that by varying the parameters α and β we can obtain some existing algorithms as special cases. For $\alpha = 0$ and $\beta = 0$ we obtain the greedy algorithm for the MaxCoverage problem. We refer to this algorithm as *Greedy-MaxCover*. For $\beta = 0$ we obtain the greedy approximation algorithm for the case of the $\text{Eff}_{\min}$ efficiency function.

3.3 The EffSetCover Problem

Similar to the EffMaxCoverage problem, the EffSetCover is also NP-hard, since the special case where $\alpha = 0$ is equivalent to the SetCover problem which is known to be NP-hard [31]. For the SetCover problem we can use the ILP formulation below to obtain an optimal solution. We can add the efficiency constraints 6 and 7 to obtain a solution to the EffSetCover problem.

$$\text{minimize} \sum_{i=1}^{n} x_i \tag{8}$$

$$\text{subject to} \sum_{i:t_j \in \mathcal{T}_{R_i}}^{n} x_i \geq 1 \quad \forall t_j \in \mathcal{T} \tag{9}$$

$$x_i = \{0, 1\}. \tag{10}$$

4 Matching Reviews and Tips

As mentioned in Section 2, for matching reviews and tips, we consider three types of similarity. In this section, we define each type in detail, as well as how they can be measured. We also describe how to combine them into the matching function $\mathcal{F}$.

Syntactic similarity (SynSim). A review sentence and a tip are syntactically similar if they share important keywords. A well-established model for keyword similarity is the vector space model [20]. Each review sentence s, and each tip t, are associated with vectors $\mathbf{s}$ and $\mathbf{t}$ respectively. The dimensionality of the vectors is the size of the vocabulary. Each vector entry signifies the importance of the corresponding word. The degree of similarity between the sentence and the tip is then measured as the cosine similarity [20]. Therefore we have:

$$\text{SynSim}(s, t) = \text{cosine}(\mathbf{s}, \mathbf{t}).$$

To compute the importance weights for the words we form a corpus of documents, where each document represents an entity (e.g., restaurant) and it consists of all the tips about this entity. We use the standard *tf-idf* [20] scheme for determining the importance of a word.

Semantic similarity (SemSim). A review sentence and a tip are semantically similar, when they are describing the same concept, even if they do not use exactly the same keywords. For instance, when discussing ramen noodles, some may choose to use "broth", while others use "soup", although both refer to the same concept. There are two main challenges in determining semantic similarity: first, identifying automatically concepts that are important to each entity; second, finding the words that are used to describe the concepts in text. To deal with these challenges, we seek an unsupervised approach that can work across different domains. Inspired by the work in text mining, we propose to discover the latent concepts from text using topic modeling.

While there are several potential topic models, here we describe an approach based on the well-known Latent Dirichlet Allocation (LDA) [1]. For illustration, in Table 1, we show an example of topics discovered from the Foursquare tips of the restaurant *Shake Shack*,[4] located at Madison Square Park in New York. Due to space limitation, we show five out of 20 topics learned from the restaurant's tips. The topics reflect: (1) the main menu of burgers, shakes, and

4. https://foursquare.com/v/shake-shack/40e74880f964a520150a1fe3

TABLE 1
Example Topics for *Shake Shack, Madison Square Park*

Topic #	Top 5 keywords
1	burger, shack, shake, fri, chees
2	line, wait, burger, worth, it', long
3	custard, frozen, flavor, awesom, eat
4	burger, spot, foodspot, shroom, shack
5	park, madison, squar, stand, locat

cheese fries; (2) the waiting time; (3) frozen custard; (4) the mushroom burger vegetarian option; and (5) the location at Madison Square Park. This small example serves to demonstrate that the topics do reflect the pertinent concepts in each restaurant.

LDA associates each tip t with a probability distribution θ_t over the topics, which captures which topics are most important for t. Given the topics, and the corresponding language model for each topic as it is learnt from the tips, we can estimate the topic distribution θ_s for each review sentence s, which captures how well a sentence s reflects the topics being discussed in the corpus of tips. To measure the semantic similarity between a review sentence and a tip, we measure the similarity of the topic distributions θ_s and θ_t. A commonly used distance measure between two probability distributions is the Jensen-Shannon Divergence (JSD) [20]. Intuitively, a sentence and a tip are semantically similar if their topic distributions can describe each other well. The lower the divergence, the greater is the similarity. Therefore, we have:

$$\mathrm{SemSim}(s,t) = 1 - \mathrm{JSD}(\theta_s, \theta_t).$$

Sentiment similarity (SentSim). A matching pair of review sentence and tip should also represent the same sentiment. Sentiment extraction from text is an active area of research [25]. Here, we cast the problem as a classification problem, where the goal is to predict the sentiment (positive or negative) of a sentence or a tip. We thus have two classes c^+ and c^-. We use a maximum entropy classifier (MEM) [19], which has been demonstrated to work well for sentiment classification in text [25], using N-gram features (both letter N-grams and word N-grams). To illustrate these features, in Table 2, we show several features found to be important (high feature weight) for each class. Letter N-grams are prefixed by a dash. For the positive class, we have words such as "so" and "best", as well as the superlative suffix (letter N-gram) "-est". For the negative class, we have negations, such as "not", or words with negative connotation, such as "over" or "but".

Given a document d (a sentence or a tip), the MEM classifier outputs conditional probabilities $P(c^+|d)$ and $P(c^- \mid d)$ for the positive and negative classes, where $P(c^+ \mid d) + P(c^- \mid d) = 1$. Given the classifier output for a document d, we transform the probability $P(c^+ \mid d) \in [0,1]$ into a polarity value $\mathrm{polarity}(d) = 2P(c^+ \mid d) - 1$, ranging from -1 (extremely negative) to 1 (extremely positive). For $P(c^+ \mid d)$ close to 1/2, the polarity is close to zero, which agrees with our intuition that in these cases the document has neutral polarity. We define the sentiment similarity between a sentence s and a tip t as the product of their polarities: it

TABLE 2
Example Important Features for Sentiment Classification

Sentiment	Features
Positive	so, best, all, -est, -lov
Negative	-ted, not, over, not worth, but

approaches 1 when the sentence and the tip polarities are similar; it approaches -1 when their polarities are opposite; it approaches 0 when the tip or the sentence polarity is neutral. Therefore, we have:

$$\mathrm{SentSim}(s,t) = \mathrm{polarity}(s) \times \mathrm{polarity}(t).$$

Matching function. Having defined the three main criteria for matching (syntactic, semantic, and sentiment), we would like to combine them to determine whether a review sentence s and a tip t match or not. One principled way to combine the three criteria is through a supervised binary classification framework, with two classes *match* and *non-match*, using the three features we defined above: syntactic similarity $\mathrm{SynSim}(s,t)$, semantic similarity $\mathrm{SemSim}(s,t)$, and sentiment similarity $\mathrm{SentSim}(s,t)$. For a sentence-tip pair (s,t) the classifier estimates the matching probability $P(s,t)$. The binary mapping function $\mathcal{F}(s,t)$ is thus defined in terms of the matching probability, using on a threshold η, as follows:

$$\mathcal{F}(s,t) = \begin{cases} 1, & \text{if } P(s,t) > \eta, \\ 0, & \text{otherwise.} \end{cases}$$

We discuss the choice of η in the experiments.

5 Experiments

The objective of the experiments is to showcase the effectiveness of the proposed approach in finding a set of reviews that cover as many tips as possible, in an efficient manner. First, we will describe the real-life dataset used in the experiment. This is followed by an evaluation of the matching process described in Section 4. We then investigate how the coverage algorithms proposed in Section 3 behave under different parameter settings, as well as how they compare against the baselines.

5.1 Dataset

The experiments require data coming from two different sources (reviews and micro-reviews), concerning the same set of entities. We pick the domain of restaurants, because it is a popular domain where there are active platforms for reviews as well as for micro-reviews. For reviews, we crawl Yelp.com to obtain the reviews of the top 110 restaurants in New York City with the highest number of reviews as of March 2012. For micro-reviews, we crawl the popular check-in site Foursquare.com to obtain the tips of the same 110 restaurants. However, some of the restaurants in Foursquare.com have too few tips, which may not adequately reflect the restaurant's information. Therefore, we retain only the 102 restaurants with at least 50 tips each. For these 102 restaurants, we have a total of 96,612 reviews, with a minimum of 584, and a maximum of 3,460 per restaurant.

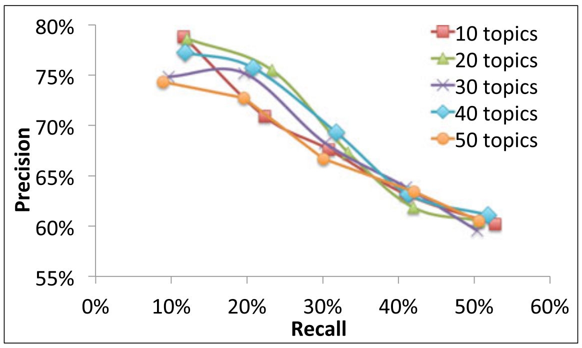

Fig. 1. Matching: Precision-recall curve.

TABLE 3
Performance of Matching Classifier

Threshold	Matching Pairs		Coverable Tips
η	Precision	Recall	
0.70	78.6%	12.1%	72.3%
0.65	75.5%	23.3%	83.5%
0.60	67.4%	33.2%	89.7%
0.55	61.9%	41.8%	93.4%
0.50	60.6%	50.4%	95.9%
All	42.9%	100.0%	100.0%

We also have a total of 14,740 tips, with a minimum of 51, and a maximum of 498 per restaurant. Note that we get the *full* set of reviews and tips of each restaurant at the time of extraction, and that these are the realistic sizes of the real-world data. It is also important to note that every restaurant is a distinct instance of the coverage problem.

5.2 Matching

Matching between a review sentence and a tip is by itself a challenging problem. Our objective in this experiment is to establish that we achieve a reasonable level of quality in matching, such that the reviews selected by the coverage algorithms would be a good reflection of the covered tips.

To build the matching classifier, we generate the three real-valued features described in Section 4. For semantic similarity, we train LDA [1] topic models using the MALLET toolbox [21]. Because topic modeling is probabilistic, we average the semantic similarity over ten runs. To determine the sentiment polarity of each sentence and tip, we train a sentiment classifier using the Stanford Classifier toolkit [19] with textual features (word and letter n-grams).

To train the matching classifier, we sample 20 entities, and for each entity we sample 50 sentence-tip pairs sharing at least one common word. We assume no match otherwise. For these 1,000 pairs, we get three judges to label whether the pairs match in meaning, and take the majority label as the ground truth. Finally, we use the real-valued features and the majority labels to train the matching classifier using the MEM classifier from [19]. Based on the feature weights learned by the classifier, we find that among the three features, semantic similarity is the most important, followed by syntactic, and lastly sentiment.

To validate the effectiveness of the matching classifier, we conduct a five-fold validation, with 80:20 split between training and testing in each fold. As metrics, we use precision and recall at the pair level. Precision is the fraction of true matching pairs within the set of classified matching pairs. Recall is the fraction of true matching pairs found by the classifier within the set of all true matching pairs. Because the objective of matching is to determine which review sentences match a tip, it is important to have high precision, so we can be confident that the reviews selected by the algorithms actually reflect the underlying tips.

Number of topics. We study the performance of matching classifier as we vary the number of topics used for the semantic similarity. In Fig. 1, we plot the precision-recall curve for $\eta = 0.65$ (discussed below). It appears that the effect of the number of topics on precision and recall is not significant. The performance for 20-40 topics is better than for 10 (which may underfit), or for 50 (which may overfit). The results for 20 topics are slightly better, especially in terms of precision, which is our main concern. Subsequently, we show the results for 20 topics.

Threshold η. We experiment with different values for the threshold η on the probability of matching $P(s,t)$. Table 3 shows the precision and recall of the matching classifier at different values of η. If we were to skip the matching classification, and simply take all the pairs with at least one common word as matching, we get a precision of only 43%, which means more than half of all matching pairs would be incorrect. As we increase the threshold η, the precision improves significantly. If we would like at least three-quarters of matching pairs to be correct, we need to set the threshold at 0.65 of higher. At this threshold, the recall is low at 23%, but this can be compensated by the fact that a tip may be covered by many different sentences.

The last column shows the percentage of tips that are covered by at least one sentence in some review. At 0.65, we cover 83.5 percent of all tips, a substantial subset. In the following, we present results for $\eta = 0.65$.

We also experimented with temporal similarity (how close in time a tip and a review were posted) as an additional feature for the matching classifier. We found that it has essentially no effect on the accuracy, which remains practically identical for $\eta = 0.65$.

To get an intuitive sense of the matching quality, we show some examples of matching pairs for the burger joint *Shake Shack* in Table 4. The first pair discuss how good the fries are. The second pair discuss the long waiting times, while the third pair discuss the mushroom burger. These examples showcase how the features, i.e., syntactic, semantic, and sentiment similarity, help to identify relevant matching pairs.

5.3 Coverage and Efficiency

Given the matching between review sentences and tips, we now investigate the effectiveness of our algorithms in terms of coverage and efficiency. First, we will compare the EFF-MAXCOVERAGE formulation with the EFFSETCOVER formulation. Then, we will compare the proposed greedy algorithm against the optimal solution, and then against the baselines. Finally, we will show a case study to provide an intuitive

TABLE 4
Example Matching Pairs for *Shake Shack*, *Madison Square Park*

ID		Review Sentence—Tip Matching Pair	$P(s,t)$
1	Review	The fries were really good too.	0.80
	Tip	Great french fries	
2	Review	The Bad: One burger will not do it alone The Ugly: The line of people the length of Great Wall of China Total wait time on the first visit: 1 hr. 25 min.	0.78
	Tip	Go here at odd eating hours or in bad weather. Otherwise, you'll be in a very long line for the best burger in the world.	
3	Review	Shakes, fries, burgers, and my favorite, the portabello "burger" (doesn't actually have any meat, just deep fried mushroom and cheeeeese!	0.72
	Tip	Mushroom burger. Transfat-free fries. healthy-junk food heaven. concrete shake is wooohhhaaaa!	

sense of the kind of results that our algorithm produces compared to those of the baselines.

5.3.1 EffMaxCoverage vs. EffSetCover

In this section we compare the EffMaxCoverage and EffSetCover formulations. For the EffMaxCoverage problem, we set $K = 5$, a number of reviews that can be consumed quickly, while providing sufficient information about an item. Our goal is to compare this set against one that covers all the tips, both in terms of size and coverage. To avoid introducing the effects of specific algorithmic techniques into the comparison, we compare the optimal solutions, using the ILP formulation for both problems.

The first issue that we need to address is that an optimal solution does not always exist. Table 5 shows the percentage of the 102 entities, for which *ILP-EffMaxCover* can discover an optimal solution for the three efficiency functions (average, bag, and minimum). We focus on $\alpha \geq 0.5$, which is the more interesting range, since, as we will show in Section 5.3.4, the baseline *MaxCover* (maximizing coverage without efficiency constraint) has an efficiency of 0.43. The table shows that at efficiency threshold $\alpha = 0.5$, there exists an optimal solution for 90 percent of the entities. Of the 10 entities for which an optimal solution cannot be obtained, nine are due to the two-hour cutoff that we impose on the lp_solve package in order to avoid infinite running time for large problem sizes. For the last entity, there is no optimal solution that can satisfy the constraints. As expected, as we increase α, the percentage of optimal solutions decreases monotonically, as there are fewer and fewer entities that can satisfy the increasingly stringent efficiency constraint.

Table 6 shows the percentage of entities, for which *ILP-EffSetCover* can discover an optimal solution. The ILP solver terminates under two hours in all cases, therefore the cases where there is no optimal solution are due to the non-existence of such a solution. Note that the percentages are much lower than those for *ILP-EffMaxCover*, especially for larger α. This implies that requiring perfect coverage is too stringent, since in most cases such a solution cannot be found.

Table 7 shows the average number of reviews produced by the *ILP-EffSetCover* algorithm for the entities for which an optimal solution exists. From the table it is clear that in order to cover all the tips, we require a substantial number of reviews (in the order of twenties and thirties). This number becomes higher for larger values of α indicating that we cannot reduce the amount of content by imposing stricter efficiency constraints. For the application scenarios we consider, where a user wants to quickly make a decision by reading little content on a screen with limited real estate, this number of reviews is too high.

We now investigate the level of coverage achieved by *ILP-EffMaxCover* as a fraction of the full coverage. As a representative, we use *ILP-EffMaxCover*$_{\text{avg}}$, but similar conclusions can be drawn for other efficiency functions. Fig. 2 plots the coverage of *ILP-EffMaxCover*$_{\text{avg}}$ at $\alpha = 0.5$, for different values of K. We divide the entities into two groups. The first group (corresponding to the top red line in Fig. 2) consists of 63 entities for which a perfect coverage of all tips exists. It shows that as K increases the solution for *ILP-EffMaxCover*$_{\text{avg}}$ eventually converges to the perfect coverage (1.0) around $K = 30$. Interestingly, even for smaller values of K, which are of interest to our applications, the coverage is still very high, e.g., 0.8 coverage at $K = 5$. The second group (corresponding to the bottom blue line in Fig. 2) consists of 29 entities for which no perfect coverage exists. Even in such cases, the *ILP-EffMaxCover* is able to obtain a satisfactory solution (coverage between 0.6 and 0.7).

In conclusion, we find that the EffMaxCoverage formulation is the appropriate formulation for the problem we consider. Obtaining a set of reviews with full coverage is often

TABLE 5
ILP-EffMaxCover: Entities with Optimal Solutions

	Efficiency Threshold α					
	0.5	0.6	0.7	0.8	0.9	1.0
ILP-EffMaxCover$_{\text{avg}}$	90%	90%	82%	76%	47%	44%
ILP-EffMaxCover$_{\text{bag}}$	90%	90%	82%	76%	47%	44%
ILP-EffMaxCover$_{\text{min}}$	90%	90%	82%	76%	47%	44%

TABLE 6
ILP-EffSetCover: Entities with Optimal Solutions

	Efficiency Threshold α					
	0.5	0.6	0.7	0.8	0.9	1.0
ILP-EffSetCover$_{\text{avg}}$	71%	47%	28%	6%	2%	0%
ILP-EffSetCover$_{\text{bag}}$	63%	39%	21%	5%	1%	0%
ILP-EffSetCover$_{\text{min}}$	6%	3%	0%	0%	0%	0%

TABLE 7
ILP-EffSetCover: Optimal Number of Reviews

	Efficiency Threshold α					
	0.5	0.6	0.7	0.8	0.9	1.0
ILP-EffSetCover$_{\text{avg}}$	26	28	28	31	18	-
ILP-EffSetCover$_{\text{bag}}$	30	30	41	48	32	-
ILP-EffSetCover$_{\text{min}}$	12	9	-	-	-	-

not feasible, and when feasible it results in a large number of reviews. Moreover, EFFMAXCOVERAGE, even for small values of K, produces a set of reviews with significant coverage of the tips.

5.3.2 *EFFMAXCOVERAGE: ILP vs. Greedy*

While ILP produces an optimal solution, it suffers from high running time. On the other hand, the greedy algorithm is much more efficient, but it produces an approximate solution. We will now measure experimentally how closely the greedy algorithm can approximate the optimal solution by ILP. In particular, we will measure two quantities. The first quantity, *coverage approximation ratio*, takes the ratio of the coverage by *Greedy-EffMaxCover* to that by *ILP-EffMaxCover*. The second quantity, *efficiency approximation ratio*, is a similar measurement on the efficiency. Both measurements are averaged across the 92 entities (i.e., the 90 percent of entities with optimal results at $\alpha = 0.5$ in Table 5). These ratios measure how closely the greedy algorithm approximates the optimal solutions.

Table 8 shows that both the coverage and efficiency approximation ratios are very close to 1 for any α. This means that in practical terms, the greedy solution of *EffMaxCover* is essentially identical to the ILP solution.

Table 9 compares the time taken by the greedy and ILP versions (on a regular PC with Intel Core i5 3.20 GHz CPU and 4 GB RAM). Greedy is easily two orders of magnitude faster than ILP. Even this speedup is a severe underestimation, since it does not account for the cases where ILP cannot complete its run within the two-hour cut-off time. Such cases involve larger problem sizes. The entities that ILP can solve within the cut-off time have on average 606 reviews and 107 tips. The unsolvable entities have on average 1,505 reviews and 246 tips. The greedy algorithm can still solve the latter very efficiently (within 25 ms).

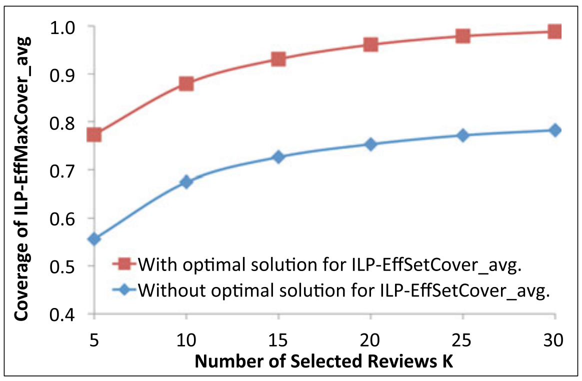

Fig. 2. Coverage of *ILP-EffMaxCover*$_{\text{avg}}$ ($\alpha = 0.5$).

TABLE 8
Greedy vs. ILP: Approximation Ratios ($K = 5$)

	Coverage Approx. Ratio			Efficiency Approx. Ratio		
α	Eff$_{\text{avg}}$	Eff$_{\text{bag}}$	Eff$_{\text{min}}$	Eff$_{\text{avg}}$	Eff$_{\text{bag}}$	Eff$_{\text{min}}$
0.5	0.96	0.96	0.98	1.04	1.01	1.02
0.6	0.95	0.97	0.99	1.03	1.00	1.00
0.7	0.97	0.97	1.00	1.01	1.00	1.00
0.8	0.98	0.98	1.00	1.00	1.00	1.00
0.9	0.97	0.98	1.00	1.00	1.00	1.00
1.0	1.00	1.00	1.00	1.00	1.00	1.00

In subsequent sections, we will focus on the understanding and evaluation of the greedy algorithm.

5.3.3 Greedy-EffMaxCover*: Parameter Analysis*

There are two ways in which *Greedy-EffMaxCover* controls the efficiency of the selected set of reviews. The first is by the threshold α, which guarantees the efficiency of the set is at least α. The second is by the parameter β which controls the sensitivity of the selection process to the efficiency of the next review to be added to the set. In the following, we study the effect of the parameters α and β.

EffMaxCover: Varying α. To isolate the effect of α, we fix $\beta = 0$, making the cost a constant, independent of the efficiency. We vary α from 0.5 to 1.0 for $K = 5$, and show the coverage and efficiency in Table 10. Table 10 shows that as α increases, as expected, efficiency monotonically increases. However, coverage monotonically decreases, since the constraint on α disqualifies some of the higher-coverage, but lower-efficiency reviews. Furthermore, as α increases, the percentage of entities for which a result that satisfies all the constraints can be found, also decreases. For large values of α, i.e., $\alpha > 0.8$, the percentage of entities with a solution drops below 85%, which is too limiting. Therefore, subsequently, we will search for a suitable setting in the range $0.5 \le \alpha \le 0.8$.

Among the different ways of aggregating efficiency for *Greedy-EffMaxCover*, we observe Eff$_{\text{avg}}$ and Eff$_{\text{bag}}$ behave similarly, and slightly differently from Eff$_{\text{min}}$. The Eff$_{\text{min}}$ tends to have higher efficiency but lower coverage. This is due to the fact that every selected review has to meet the efficiency threshold, reducing the set of available candidate reviews. In contrast, the other two algorithms consider the efficiency of the set as a whole, and they may select reviews with high coverage that have efficiency below α, if the reviews already in the set have high efficiency. If there is a

TABLE 9
Greedy vs. ILP: Running Time in Milliseconds ($K = 5$)

	Greedy-EffMaxCover			*ILP-EffMaxCover*		
α	Eff$_{\text{avg}}$	Eff$_{\text{bag}}$	Eff$_{\text{min}}$	Eff$_{\text{avg}}$	Eff$_{\text{bag}}$	Eff$_{\text{min}}$
0.5	2.6	2.2	1.0	127.8	432.0	3.9
0.6	1.5	1.4	0.6	202.2	384.5	3.4
0.7	0.9	0.9	0.2	228.7	164.3	1.0
0.8	0.5	0.5	0.1	109.2	82.8	0.6
0.9	0.3	0.2	0.1	55.4	32.5	0.2
1.0	0.2	0.1	0.1	6.7	6.4	0.2

TABLE 10
Greedy-EffMaxCover ($K = 5, \beta = 0$)

	% Entities	Coverage			Efficiency		
α	w. Results	Eff_{avg}	Eff_{bag}	Eff_{min}	Eff_{avg}	Eff_{bag}	Eff_{min}
0.5	99%	0.68	0.67	0.65	0.54	0.55	0.61
0.6	99%	0.64	0.63	0.60	0.63	0.63	0.69
0.7	91%	0.62	0.61	0.56	0.72	0.72	0.78
0.8	85%	0.55	0.54	0.52	0.81	0.81	0.84
0.9	56%	0.58	0.58	0.47	0.91	0.90	0.99
1.0	53%	0.47	0.47	0.47	1.00	1.00	1.00

strict requirement on efficiency, Eff_{min} is probably the better choice. Otherwise, Eff_{avg} and Eff_{bag} are slightly better with their higher coverage. We will use Eff_{avg} in the subsequent analysis, due to its slightly higher coverage.

EffMaxCover: Varying β. We now study the effect of β on the performance of the algorithm, for different values of α. Fig. 3 shows how the coverage and efficiency change as β increases from 0 to 1 for *Greedy-EffMaxCover* with Eff_{avg}. The curves for the other efficiency variants Eff_{bag} and Eff_{min} are similar and not shown here due to space limitation. We plot the curves for the values of α between 0.5 to 0.8, for which at least 85% of entities satisfy the constraints (see the earlier discussion on Table 10).

At $\beta = 0$, the cost is a constant, and we rely entirely on α to maintain efficiency. As we increase β, the greedy selection of reviews will increasingly be sensitive to the cost (loss in efficiency). Fig. 3 shows that for all values of α, as β increases, the efficiency increases while the coverage decreases. Interestingly, the gain in efficiency outpaces the loss in coverage. For example, for $\alpha = 0.5$, from $\beta = 0$ to $\beta = 1$, efficiency increases from 0.54 to 0.76 (efficiency gain of 0.22), while the coverage reduces from 0.68 to 0.62 (coverage loss of 0.06). This shows β is an effective way to gain efficiency with minimal loss in coverage.

In order to have a single metric that balances coverage and efficiency, inspired by the F1 measure in information retrieval, we use the harmonic mean:

$$\text{HMean}(\mathcal{S}) = \frac{2 \times \text{Cov}(\mathcal{S}) \times \text{Eff}_{\text{avg}}(\mathcal{S})}{\text{Cov}(\mathcal{S}) + \text{Eff}_{\text{avg}}(\mathcal{S})}.$$

Fig. 3c plots the harmonic mean, which consistently reaches the peak at around $\beta = 0.9$ for all values of α. For $\alpha = 0.5$ and $\beta = 0.9$, we can guarantee that the efficiency is at least 50%, and the harmonic mean is as high as possible. Subsequently, we will use this setting for *Greedy-EffMaxCover*.

5.3.4 EffMaxCover *vs. Baselines*

We now compare of the proposed approach *EffMaxCover* to baseline approaches. As previously discussed in our comparison we will consider the *Greedy-EffMaxCover*$_{\text{avg}}$ algorithm with $\alpha = 0.5$ and $\beta = 0.9$.

Baselines. Our primary baseline is *MaxCover*, which also has the objective of maximizing coverage, but does not consider the efficiency constraint. Because *MaxCover* is not constrained in the review selection, it obtains a relatively high coverage of 0.72, which is the upper bound for *EffMaxCover*. *MaxCover*'s efficiency is only 0.43, and this is the lower bound for *EffMaxCover* that searches for a more efficient set of reviews.

We also consider the following additional baselines. *MaxLength* selects the longest K reviews, with the intuition that longer reviews may cover more tips. *MinLength* selects the shortest K reviews (with at least five sentences), with the intuition that shorter reviews may be more efficient. *Useful* selects the K reviews with the highest number of usefulness votes as voted by Yelp users (the vote is indicated in each review). To emphasize the statistical significance of the results, we also compare to the performance of *Random*, which selects K reviews randomly. For *Random*, we average the coverage and efficiency across 1,000 random runs, and plot the median, as well as the min and max.

Varying K. Fig. 4a shows how coverage varies with K for various methods. As expected, *MaxCover* has the highest coverage, followed closely by the *EffMaxCover* variants. *MaxLength* and *Useful* also do better than *Random*, but worse than *EffMaxCover*. *MinLength* has the lowest coverage, as it has very few sentences to capture the tips.

Fig. 4b shows that the efficiency of *EffMaxCover* algorithms is by far superior to all the baselines. This underlines the effectiveness of *EffMaxCover* in finding efficient reviews. The efficiency tends to decrease slightly with increasing K, which is expected as it gets increasingly more difficult to find high-coverage and high-efficiency reviews after each selection. Interestingly, the efficiency of *MaxLength* and *Useful* falls below that of *Random*, which could be due to the

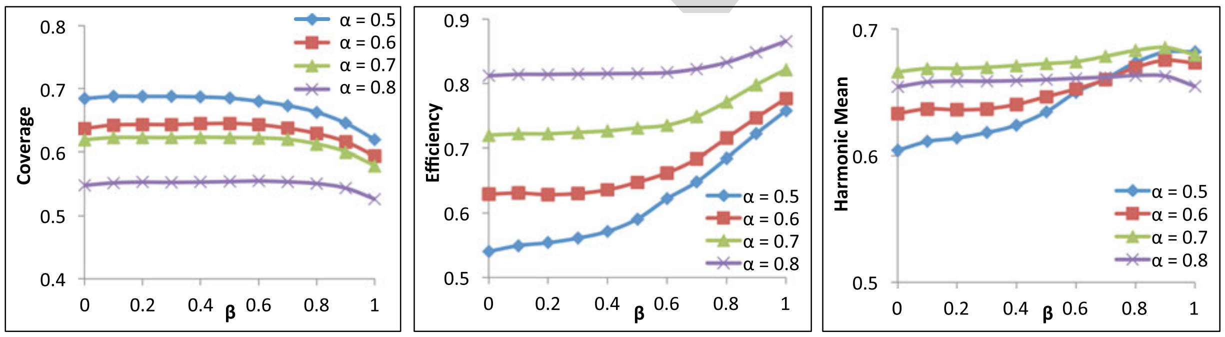

Fig. 3. Comparison of Coverage & Efficiency for varying β for $\alpha \in [0.5, 0.8]$ for *Greedy-EffMaxCover* with Eff_{avg}.

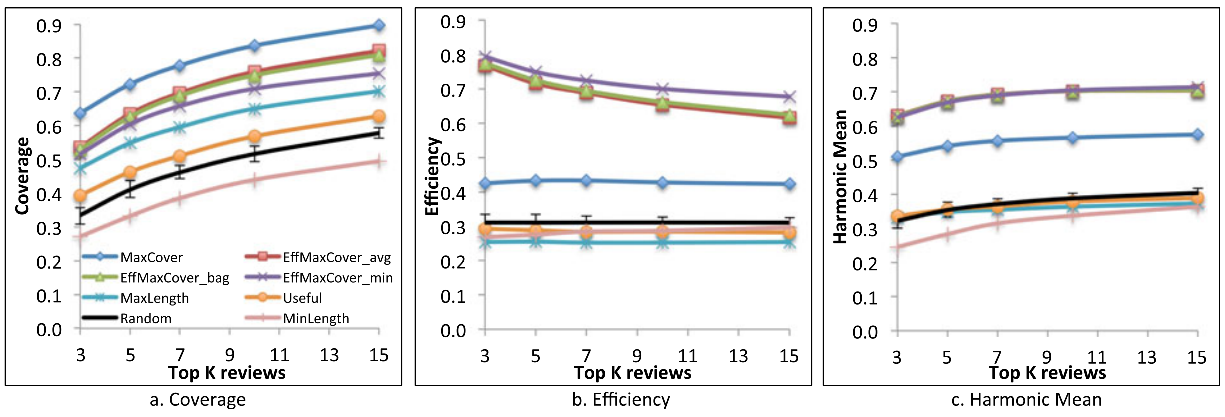

Fig. 4. Comparison of Coverage & Efficiency for varying K for $\alpha = 0.5$, $\beta = 0.9$.

length of the reviews, resulting in having many sentences that may not represent any tip. *MinLength* is more efficient than *MaxLength*, but is also worse than *Random*. This suggests that being short alone is not sufficient if the reviews do not also capture the tips well.

To emphasize the efficacy of *EffMaxCover* at achieving both coverage and efficiency, we plot the harmonic mean in Fig. 4c. It shows that the three *EffMaxCover* variants outperform the rest significantly, followed by *MaxCover*. *MaxLength* and *Useful* are no better than *Random*, whereas *MinLength* is the worst.

Qualitative analysis. We also conduct a qualitative analysis involving three judges who are not related to this paper. To each judge, we show the top three reviews selected by an algorithm for a sample of 20 restaurants, and ask the judge to choose which aspects are mentioned in the reviews from a manually hand-picked list of aspects. Because the objective is to investigate the trade-off between coverage and efficiency, we focus the comparison on two methods: *EffMaxCover*$_{\text{avg}}$ as a representative of the *EffMaxCover* variants, and *MaxCover*, as the closest competitor.

Table 11 shows that on average, the judges identify 5.1 aspects for *MaxCover*, and 3.7 aspects for *EffMaxCover*$_{\text{avg}}$. This lower coverage of aspects is expected, and consistent with the previous experiments. On the other hand, the reviews selected by *EffMaxCover*$_{\text{avg}}$ are much more compact, with an average of 26.8 sentences total in three reviews, compared to the lengthy 121 sentences by *MaxCover*. This suggests a gain in efficiency. If we look at the density of information covered (the ratio of aspects covered per sentence), the third column of Table 11 shows that *EffMaxCover*$_{\text{avg}}$ has much higher density of 0.14 aspects per sentence, than 0.04 by *MaxCover*.

5.4 Case Study

To illustrate the different types of reviews selected by the various criteria, as a case study, we show an example of the top review selected by each algorithm for the venue *Shake Shack* in Table 12. This is a burger joint located in Madison Square Park in New York.

TABLE 11
User Study Comparing EffMaxCover and MaxCover

Algorithm	*Aspects*	*Sentences*	*Aspects per sentence*
EffMaxCover$_{\text{avg}}$	3.7	26.8	**0.14**
MaxCover	5.1	121.0	0.04

The top review selected by the greedy version of *EffMaxCover* (all three efficiency variants selected the same) is compact and informative, describing the main attributes of the place: the long wait at peak hours, the location at Madison Square Park, the dishes (burgers, fries), as well as the affordable prices. If one is to select only one review to show on a mobile device, this review conveys a lot of information with a small footprint.

MaxCover's top review also covers these attributes, but with a significantly longer review. Parts of the review are not to the point. For instance, the mentions of "hearing voices inside my head" and "Disney princess" do not concern the restaurant directly. The same can be said for *MaxLength*, which discusses the same aspects, but using many more words. These very long reviews require more effort to read on a mobile device.

The other two reviews, while also compact, are not informative. *Useful*'s top review is written in a sarcastic tone, including irrelevant mentions (e.g., "I hate when the Mets lose."). *MinLength*'s top review is too short and only covers the generics, without getting into helpful details such as dishes or waiting time.

6 Related Work

Mining reviews. Recently, there is a line of work that deals with the selection of a "good" set of reviews. In [14], the objective is to select a set of reviews that cover all attributes (for a given set of attributes). In [29], the objective is refined to also include both the positive and negative aspects of each attribute. The work in [13] further seeks to preserve the underlying distribution of positive and negative comments in the reviews. In [32], the objective is to cover diversified opinion clusters. Related to review selection, [28] considers the problem of selecting a good set of photos based on quality, diversity, and coverage.

Our work is along the same lines, but is distinct in two ways. *First*, in terms of formulation, we seek to represent micro-reviews, rather than attributes. *Second*, in terms of approach, we introduce the efficiency requirement to the coverage formulation. To compare against approaches that focus on coverage but not efficiency, we

TABLE 12
Top Review for *Shake Shack, Madison Square Park*

Algorithm	*Top Review*
EffMaxCover	While in Union Square, I decided to deviate from my usual plan of lunch at Wildwood BBQ, and chose to venture off to the Shake Shack. I tried to go last saturday, but the line was at least a 45 minute wait so I decided to go at a non-peak hour so the line wouldn't be as long. The Shake Shack is in a gorgeous location, right in the heart of Madison Square Park in Gramercy. The Shake Shack...its litterally a shack, don't expect Tao. But the burgers are one of the best in the city and are most deffinetly on par with those of the Burger Joint. While J.G. Melon has the best fries between the Burger Joint, and the Shake Shack, the Shake Shack's fries weren't too shabby. The best part, lunch cost me $8.75 (one burger, an order of fries, and a coke). Not too bad for an upscale burger. What really makes it though is the location, eating one of the best burgers in the city while gazing at the beauty of Madison Square Park, is just phenominal. The Shake Shack is a great place for lunch, it won't cost you an arm and a leg, and you'll eat well.
MaxCover	Psh, who travels to Manhattan for a burger at 11 ? Alright fine, I did exactly just that. Now, for a few years, I've been hearing murmurs of Shake Shack cranking out one of the best burgers in town. I told my doctor I was hearing voices inside my head, but he told me to just go to Shake Shack. I guess he's a fan too. So went I did. Accompanied by the abundance of squirrels, birds, and pigeons (I can't group them in the same category), I waited in line, feeling like a Disney princess. Since nobody else is crazy enough to get burgers 11 in the morning in the freezing cold, the line was pretty short. (I wasn't as crazy as the guy that downed two shakes in the time I was there though) "A single cheeseburger and a Shack burger pleas–oh and fries, cheese fries... please !" A wand-like device was handed to us. Shrugging, we picked our seats. Not every table comes with a heat lamp, but they do nothing anyways. Except for burning the top of heads. We waited a short while for our food to arrive– approximately ten minutes. A few minutes later, I found myself looking at a hot-dog-at-a-baseball-game-esque box. Our treasures laid within. Food: 10/10 Though the burgers were smaller than I had expected, I appreciated the balance of each ingredient. The ratios were perfect. The bun wasn't too small– there was just enough meat peeking out, so you don't feel jipped for paying more than 3 bucks for a single burger. Both burgers weren't pink in the middle–I had expected med rare. The seasoning was fantastic, enough so you don't feel like you're chewing on dead cow. Single cheeseburger- Interestingly enough, despite the fact that Shake Shack is a chain, not every burger is created equally. Meaning, my boyfriend's Shake burger was a tiddle bit rarer than mine. Mine was still juicy, but lost a bit of its tenderness. The lettuce and tomato was extremely fresh– I loved how they give you the lettuce by the leaf, and like the crazy shredded massacres that they plop down in other establishments. Cheese ? Deliciously melty. It makes me feel okay about being American. A solid burger, a solid burger. Shack burger- Deeeeeeeeeeeeelicious. Oh my goodness, I saw stars. The sauce tasted a bit naughty, in the fattening but tasty sort of way. Props for a nice thickness in the patties– nothing like those wimpy thin, thin patties. This is the stuff. Cheese fries- I love fries. Therefore, when you give me fries, smothered in fake, plastic-like cheese, I will love them even more. You can't really go wrong, people. Delightfully crunchy, pretty good for crinkle cut fries (they're usually soft and floppy). Though it's not the cheapest option, it sure tastes a lot better than any other fast food establishment I've been in. I'm more than happy to shell out the bucks, knowing that I'll be guaranteed these delicious things. Wowza. I'm coming for you Shake Shack. Hide 'yo burgers.
MaxLength	Very solid fast food burger ***Short Review Came here for dinner Oct'11. Very good fast food burger. I feel it tops 'Five Guys' and 'In N Out'. What makes it stand out are great beef patties & well thought out use of cheese & sauce. Beef patties are relatively thick & juicy. Burger doesn't fall apart and isn't too messy. Price of the double patty 'Shack Burger' at $7 isn't cheap, but I feel the value is reasonable for what you get. Fries are decent but not exceptional. My coffee & vanilla shake was awesome. Price isn't cheap, and value is probably about average. Cost of a double patty 'Shack Burger', regular fries, and a 'Fair Shake' set me back around $16 after taxes, & before tip. Waiting for the burger here can take some time. It took me approximately 30 mins from waiting in line to getting my food. Ambiance is outdoors seating. Its fine as long as the weather's tolerable. ***Detailed Review I really enjoyed Shake Shack. I feel it takes on some of the qualities of better fast food burgers, improves on them, and adds a bit of it's own charm. What's also great is their commitment to excellent shakes, offering beer, and some promising desserts (which unfortunately I didn't get the chance to try). This is one of my favorite fast food burgers right now, certainly much better than both 5 Guys and 'In N Out'. It's also a lot more enjoyable than many of the gourmet burgers, often which are more expensive than Shake Shack. On to the food; 'Shack Burger' American cheese, lettuce, tomato, & shack sauce. (Double patty, approx $7) (Dish Rating, 75%) At first glance, this burger resembles somewhat of a hybrid between In N Out & 5 guys. Like 5 guys, you get two larger, thicker (relative to a fast food burger) beef patties. The patties are thicker & juicier than 5 guys. The meat at 5 guys is cooked throughout, but Shake Shack does it closer to medium. Inside of patty is a little pink. Meat is quite tasty with a distinct char. Size of patty isn't small either, maybe around 4oz each, or 8oz total. When biting into the burger, the beef patty is the central focus. The grind of the beef is fairly coarse, so the texture is more chewy although not difficult to chew. Biting into the juicy, chewy meat is delicious affair. Like 'In N Out', there's also a heavier presence of cheese and sauce that gives the burger some extra flavor (if that's your preference). However, it excels further in that the sauce and cheese aren't so over-powering. Unlike In N Out, the sauce isn't quite as thick or sweet. The cheese isn't quite as strong or gooey. There is a great balance between cheese, sauce, and meat. Mostly with the sauce and cheese complementing the meat in a more balanced manner. Other than the cheese and sauce, it's Shake Shack's burger doesn't seem to focus too much on other condiments. There's a slice of tomato and some regular lettuce. I feel that's totally fine, as there's already enough good things going on, that there need not be much more complexity Bun used, I felt did it's job but wasn't in itself taking up much attention. It held the burger in place, wasn't heavy, and wasn't too messy to eat. It's difficult to pinpoint an inherent weakness for Shake Shack. As a fast-food burger, it's close to as good as it gets. Only 'Little Big Burger' in Portland, OR has a slight advantage, and that's because the meat quality is better, the beef patty is thicker & even juicier. They also allow you to cook the beef patty 'rare' or 'medium rare'. Nonetheless, Shake Shack isn't far behind. For a somewhat larger burger franchise, I feel it's hard to top. Fries here are probably closer to decent. They're a crinkle cut. Thankfully low on grease, starch, and seasoning. They're served nice and crispy. There's no inherent weakness, but nothing outstanding about them either. On this count, I feel Shake Shack also falls behind somewhat. 'Fair Shake' Vanilla shake spun 100% certified Arabica fairly traded coffee ($5.50) (Dish Rating, 76%) As far as shakes go, I only had the chance to try their vanilla coffee shake. Here I felt Shake Shack were truly exceptional. The flavor of the vanilla and coffee quite intense, and the consistency of the shake to be rich but not too heavy. It's difficult for me to articulate beyond that what makes a good shake, but I feel that it was exceptional. There are few places that can make a better shake. Some people might feel that $7 for a Shake Shack double patty burger is too expensive. At this price point, it's beginning to encroach on the cost of many gourmet burgers. Nonetheless, it's also a whole lot better than several gourmet burgers. It's probably not a great deal, but the quality of the burger justifies it's cost in my opinion. If you're looking for value based on quantity of food, or the lowest price possible, you might be better off looking elsewhere.
Useful	I hate Shake Shack. I hate the wait. I hate the hype. I hate it's proximity to my office. I hate my absolute inability to resist whenever anyone suggests it. I hate when I get a craving, and no one else has it. Because I hate standing in line, friendless. I hate the Outback-esque vibrating wand they hand you. I hate mosquitoes. I hate humidity. I hate you. I hate non-potato bread. Why does it exist? I hate that the burgers taste better at CitiField. I hate when the Mets lose. I hate that they lose a lot. I hate my gut, which Shake Shack is at least somewhat responsible for. I hate that a single cheeseburger is too small, and a double necessary. I hate that it would probably take me less time to trek up to the UWS location and back than it would to wait here in Madison Square Park. But I love life when I'm eating delicious burgers and cheese fries in the middle of the park. You just have to ask yourself if the wait is worth the 3 minute face-stuffing. And you know what? It often is.
MinLength	Sweet corn frozen custard? Mmmm. Mint honeydew? MMMM. All hail the Shake Shack and summer!

compare against a max coverage algorithm as a baseline in Section 5. There also exists a variant of max coverage called budgeted max coverage [10] where the constraint is a total cost that cannot be exceeded. Our coverage formulation is different in how both constraints of cost and count apply.

Another related coverage formulation is the red-blue set cover problem [2], [26], whereby the input is a collection of sets, and each set may contain a mix of red and blue elements. The objective is to select a sub-collection of sets that covers all blue elements, but covers as few red elements as possible. If we interpret a red element as an irrelevant sentence in a review, the objective is then to get a set cover while minimizing the *number* of irrelevant sentences. In our work, efficiency is used as a threshold constraint, rather than a minimization objective. Moreover, our efficiency definitions are not expressed in terms of a count, and instead are expressed in terms of fractions, which results in a significantly different formulation.

Related to the notion of finding a "good" set of reviews is the problem of determining the quality of each individual review [17]. Sites such as Amazon or Yelp allow users to rate each review by its helpfulness or usefulness. Most review ranking works rely on a supervised regression or classification approach, using the helpfulness votes as the target class [7], [11], [16]. One possible formulation to produce a set of reviews is to first rank all the reviews based on individual merits, and then selecting the top K. The weakness of this formulation is that it ignores the potential similarities among the top reviews. It may well be that the top few reviews all represent the same information. For comparison, we introduced a baseline called *Useful* in Section 5, which ranks reviews by its usefulness votes, and selects the top K.

Our work is also related to review summarization, where the goal is to gain a quick overview of the underlying corpus of reviews. Existing approaches vary in the kind of summary they produce. In [8], [34], the summary is a list of features, the statistics of positive and negative opinions, as well as some example sentences. In [5], [22], the summary is a list of short phrases. If we treat a review as a document, the summary could also take the form of a subset of sentences extracted from the underlying documents [15], or a list of sentences generated abstractively [4]. Different from these works, our objective is closer to micro-reviews summarization (using reviews).

Mining micro-reviews. Compared to reviews, there has not been as much interest in micro-reviews within the research community. One related work focuses on very short comments on eBay left by buyers about sellers [18], but the problem there was to extract aspects from the comments. There are also works [9], [12] on analyzing opinions in micro-blogging services such as Twitter. However, because Twitter is a general micro-blogging platform, these opinions are usually about more general concepts (e.g., brands, hashtags) rather than specific entities (e.g., products, restaurants). Unlike Foursquare tips, tweets are not attached to any entity, and it is difficult to separate "reviews" from other types of content.

Most of the previous work on Foursquare or other check-in services does not view them as a source of micro-reviews, but rather as location-based social networks (LBSN), and it addresses problems such as mining user profiles [30], movement patterns [24], privacy [27], or POI recommendation [3], [6], [33].

7 Conclusion

We introduce the use of micro-reviews for finding an efficient set of reviews, which is novel in the objective of micro-review coverage, as well as in the efficiency constraint. We describe an optimal algorithm based on Integer Linear Programming. Since the problem is NP-hard, we also propose a greedy algorithm, which is virtually identical to the optimal solutions in coverage and efficiency, but it is much faster computationally. Evaluation over a corpora of restaurants' reviews and micro-reviews shows that our approach outperforms the baselines in discovering review sets consisting of compact, yet informative reviews.

Acknowledgments

This research was supported by the Singapore National Research Foundation under its International Research Centre @ Singapore Funding Initiative and administered by the IDM Programme Office, Media Development Authority (MDA). This work has been supported by the Marie Curie Reintegration Grant project titled JMUGCS which has received research funding from the European Union. An earlier version of this paper appeared in CIKM 2013.

References

[1] D. M. Blei, A. Y. Ng, and M. I. Jordan, "Latent Dirichlet allocation," *J. Mach. Learn. Res.*, vol. 3, pp. 993–1022, 2003.
[2] R. D. Carr, S. Doddi, G. Konjevod, and M. V. Marathe, "On the red-blue set cover problem," in *Proc. 11th Annu. ACM-SIAM Symp. Discrete Algorithm*, 2000, pp. 345–353.
[3] C. Cheng, H. Yang, I. King, and M. R. Lyu, "Fused matrix factorization with geographical and social influence in location-based social networks," in *Proc. 26th AAAI Conf. Artif. Intell.*, 2012, p. 1.
[4] K. Ganesan, C. Zhai, and J. Han, "Opinosis: A graph-based approach to abstractive summarization of highly redundant opinions," in *Proc. 23rd Int. Conf. Comput. Linguistics*, 2010, pp. 340–348.
[5] K. Ganesan, C. Zhai, and E. Viegas, "Micropinion generation: An unsupervised approach to generating ultra-concise summaries of opinions," in *Proc. 21st Int. Conf. World Wide Web*, 2012, pp. 869–878.
[6] H. Gao, J. Tang, X. Hu, and H. Liu, "Exploring temporal effects for location recommendation on location-based social networks," in *Proc. 7th ACM Conf. Recommender Syst.*, 2013, pp. 93–100.
[7] A. Ghose and P. G. Ipeirotis, "Designing novel review ranking systems: Predicting the usefulness and impact of reviews," in *Proc. 9th Int. Conf. Electron. Commerce*, 2007, pp. 303–310.
[8] M. Hu and B. Liu, "Mining and summarizing customer reviews," in *Proc. 10th ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining*, 2004, pp. 168–177.
[9] B. J. Jansen, M. Zhang, K. Sobel, and A. Chowdury, "Twitter power: Tweets as electronic word of mouth," *J. Amer. Soc. Inf. Sci. Technol.*, vol. 60, no. 11, pp. 2169–2188, 2009.
[10] S. Khuller, A. Moss, and J. S. Naor, "The budgeted maximum coverage problem," *Inf. Process. Lett.*, vol. 70, no. 1, pp. 39–45, 1999.
[11] S.-M. Kim, P. Pantel, T. Chklovski, and M. Pennacchiotti, "Automatically assessing review helpfulness," in *Proc. Conf. Empirical Methods Natural Lang. Process.*, 2006, pp. 423–430.
[12] E. Kouloumpis, T. Wilson, and J. Moore, "Twitter sentiment analysis: The good the bad and the omg," in *Proc. 5th Int. Conf. Weblogs Social Media*, 2011, pp. 538–541.

[13] T. Lappas, M. Crovella, and E. Terzi, "Selecting a characteristic set of reviews," in *Proc. 18th ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining*, 2012, pp. 832–840.

[14] T. Lappas and D. Gunopulos, "Efficient confident search in large review corpora," in *Proc. Eur. Conf. Mach. Learn. knowl. Discovery Databases: Part II*, 2010, pp. 195–210.

[15] H. Lin and J. Bilmes, "Multi-document summarization via budgeted maximization of submodular functions," in *Proc. Human Lang. Technol.: Annu. Conf. North Amer. Chapter Assoc. Comput. Linguistics*, 2010, pp. 912–920.

[16] Y. Liu, X. Huang, A. An, and X. Yu, "Modeling and predicting the helpfulness of online reviews," in *Proc. 8th Int. Conf. Data Mining*, 2008, pp. 443–452.

[17] Y. Lu, P. Tsaparas, A. Ntoulas, and L. Polanyi, "Exploiting social context for review quality prediction," in *Proc. 19th Int. Conf. World Wide Web*, 2010, pp. 691–700.

[18] Y. Lu, C. Zhai, and N. Sundaresan, "Rated aspect summarization of short comments," in *Proc. 18th Int. Conf. World Wide Web*, 2009, pp. 131–140.

[19] C. Manning and D. Klein, "Optimization, maxent models, and conditional estimation without magic," in *Proc. Conf. North Amer. Chapter Assoc. Comput. Linguistics Human Lang. Technol.: Tuts.*, 2003, vol. 5, p. 8.

[20] C. D. Manning and H. Schütze, *Foundations of Statistical Natural Language Processing*. Cambridge, MA, USA: MIT Press, 1999.

[21] A. K. McCallum. (2002). Mallet: A machine learning for language toolkit [Online]. Available: http://mallet.cs.umass.edu

[22] X. Meng and H. Wang, "Mining user reviews: From specification to summarization," in *Proc. ACL-IJCNLP Conf. Short Papers*, 2009, pp. 177–180.

[23] G. L. Nemhauser, L. A. Wolsey, and M. L. Fisher, "An analysis of approximations for maximizing submodular set functions-I," *Math. Programm.*, vol. 14, no. 1, pp. 265–294, 1978.

[24] A. Noulas, S. Scellato, C. Mascolo, and M. Pontil, "An empirical study of geographic user activity patterns in foursquare," in *Proc. 5th Int. AAAI Conf. Weblogs Social Media*, 2011, pp. 570–573.

[25] B. Pang, L. Lee, and S. Vaithyanathan, "Thumbs up?: Sentiment classification using machine learning techniques," in *Proc. ACL-02 Conf. Empirical Methods Natural Lang. Process.*, 2002, pp. 79–86.

[26] D. Peleg, "Approximation algorithms for the label-coverMAX and Red-Blue set cover problems," *J. Discrete Algorithms*, vol. 5, no. 1, pp. 55–64, 2007.

[27] T. Pontes, M. Vasconcelos, J. Almeida, P. Kumaraguru, and V. Almeida, "We know where you live: privacy characterization of foursquare behavior," in *Proc. ACM Conf. Ubiquitous Comput.*, 2012, pp. 898–905.

[28] P. Sinha, S. Mehrotra, and R. Jain, "Summarization of personal photologs using multidimensional content and context," in *Proc. 1st ACM Int. Conf. Multimedia Retrieval*, 2011, p. 4.

[29] P. Tsaparas, A. Ntoulas, and E. Terzi, "Selecting a comprehensive set of reviews," in *Proc. 17th ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining*, 2011, pp. 168–176.

[30] M. A. Vasconcelos, S. Ricci, J. Almeida, F. Benevenuto, and V. Almeida, "Tips, dones and todos: Uncovering user profiles in foursquare," in *Proc. 5th ACM Int. Conf. Web Search Data Mining*, 2012, pp. 653–662.

[31] V. V. Vazirani, *Approximation Algorithms*. New York, NY, USA: Springer, 2004.

[32] W. Yu, R. Zhang, X. He, and C. Sha, "Selecting a diversified set of reviews," in *Proc. 15th Asia-Pacific Web Conf.*, 2013, pp. 721–733.

[33] Q. Yuan, G. Cong, Z. Ma, A. Sun, and N. M. Thalmann, "Time-aware point-of-interest recommendation," in *Proc. 36th Int. ACM SIGIR Conf. Res. Develop. Inf. Retrieval*, 2013, pp. 363–372.

[34] L. Zhuang, F. Jing, and X.-Y. Zhu, "Movie review mining and summarization," in *Proc. 15th ACM Int. Conf. Inf. knowl. Manage.*, 2006, pp. 43–50.

Thanh-Son Nguyen received the bachelor's degree from the University of Engineering and Technology, Vietnam National University, Hanoi. He is currently working toward the PhD degree in the Information Systems program at Singapore Management University. His research interests are in data mining, as well as natural language and text processing.

Hady W. Lauw received the bachelor's and PhD degrees from Nanyang Technological University. He is an assistant professor of information systems at Singapore Management University. Previously, he was a postdoctoral researcher at Microsoft Research, and then a scientist at A*STAR's Institute for Infocomm Research. His research interest is in data mining, focusing on Web and social media data. He is a member of the IEEE.

Panayiotis Tsaparas received the PhD degree from the University of Toronto in 2004. Since then he has worked at the University of Rome, La Sapienza, and the University of Helsinki as a postdoc, and at Microsoft Research as a researcher. Since 2011, he has been an assistant professor at the University of Ioannina. His research interests include data mining, information retrieval, and social network analysis. He is a member of the IEEE.