Review Synthesis for Micro-Review Summarization

Thanh-Son Nguyen
School of Information Systems
Singapore Management University
tsnguyen.2013@phdis.smu.edu.sg

Hady W. Lauw
School of Information Systems
Singapore Management University
hadywlauw@smu.edu.sg

Panayiotis Tsaparas*
Dept. of Computer Science
University of Ioannina
Greece
tsap@cs.uoi.gr

ABSTRACT

Micro-reviews is a new type of user-generated content arising from the prevalence of mobile devices and social media in the past few years. Micro-reviews are bite-size reviews (usually under 200 characters), commonly posted on social media or check-in services, using a mobile device. They capture the immediate reaction of users, and they are rich in information, concise, and to the point. However, the abundance of micro-reviews, and their telegraphic nature make it increasingly difficult to go through them and extract the useful information, especially on a mobile device. In this paper, we address the problem of summarizing the micro-reviews of an entity, such that the summary is *representative*, *compact*, and *readable*. We formulate the summarization problem as that of synthesizing a new "review" using snippets of full-text reviews. To produce a summary that naturally balances compactness and representativeness, we work within the Minimum Description Length framework. We show that finding the optimal summary is NP-hard, and we consider approximation and heuristic algorithms. We perform a thorough evaluation of our methodology on real-life data collected from Foursquare and Yelp. We demonstrate that our summaries outperform individual reviews, as well as existing summarization approaches.

Categories and Subject Descriptors

H.4 [**Information Systems Applications**]: Miscellaneous; H.2.8 [**Database Applications**]: Data Mining

General Terms

Algorithms, Experimentation

Keywords

micro-review summarization; review synthesis

*This work has been supported by the Marie Curie Reintegration Grant project titled JMUGCS which has received research funding from the European Union.

WSDM'15, February 2–6, 2015, Shanghai, China.

ACM 978-1-4503-3317-7/15/02 ...$15.00.
http://dx.doi.org/10.1145/2684822.2685321.

1. INTRODUCTION

The confluence of the two fast-moving trends, the increasing penetration of mobile devices and the prevalence of social media, has given rise to a new breed of mobile social media platforms. Exemplars of such platforms include the well-known check-in services Foursquare and Facebook Places. In such services, users interact by sharing their experiences on various venues and services (restaurants, pubs, salons, etc.) with their friends and the public, providing invaluable decision aids to future customers.

A byproduct of this behavior is *micro-reviews*: concise, bite-sized reviews produced on micro-blogging platforms and location-based social networks. For instance, in Foursquare, micro-reviews, more popularly known as *tips*, are limited to 200 characters, and they are written by users when checking in at a particular point of interest. These tips serve several purposes: They may offer opinion on some aspects of the restaurant ("*Great place to stop by for a quick bite or for a good cup of coffee. Bustling atmosphere and reasonable prices.*"); They may give recommendations on what to order ("*We particularly love the coffee gelato flavor from Eataly. Light and creamy.*"); They may be actual "tips" or suggestions ("*If you live in the neighborhood shop after 8:30PM for minimal human traffic and shorter lines.*").

There are some characteristic differences between micro-reviews and regular reviews found on sites such as Yelp. Micro-reviews are concise, making a crisp point about a specific aspect, whereas reviews typically cover various aspects comprehensively. Micro-reviews tend to be spontaneous, giving a real-time reaction to the author's current experience, whereas reviews tend to be contemplative and reflective. Micro-reviews are often accompanied by check-ins, which lend them a degree of authenticity. Micro-reviews are growing faster than regular reviews. For instance, Foursquare content (micro-reviews) is growing 65% per year on a base of 35 million users, while Yelp content (reviews) is growing 41% per year, on a larger base of 108 million users[1].

Micro-reviews are thus an important source of information for users seeking information for making decisions. However, their increased popularity has led to an abundance of content. It is common for popular venues to have several hundreds of micro-reviews. While the concise and telegraphic nature of an individual micro-review makes it easy to convey a specific point, the very same property makes it difficult to go through a *collection* of micro-reviews to extract useful information, especially on a mobile device. This is because

[1] `http://www.fastcompany.com/3015168/foursquares-tips-growing-faster-than-yelps-reviews`

the collection consists of a large volume of fragmented opinions, all by different authors, expressing views that in some cases are highly repetitive. There exists useful information in the collection, but it is scattered across multiple micro-reviews, and as a result, diluted and obscure to the reader. To make the collective wisdom of micro-reviews useful, we need to piece it together into a single coherent piece of text.

We therefore consider the following problem. Given a collection of tips about an entity, produce a text summary of the information content of the tips. Ideally, this summary should capture most, if not all, of the points made by the tips in the collection, in a concise and coherent fashion that is easy to consume on a mobile device. Inspired by the highly complementary nature of micro-reviews and reviews, we propose to use review content for this task. While micro-reviews are good at identifying the salient points about an entity, a review is often a coherent, well-written piece of text, produced by an author who seeks to comprehensively describe her experience with the entity. We propose to *synthesize* a new "review", by taking the "best" parts of some reviews, and putting them together into a text summary.

Overview of our approach. We now give a high-level overview of our methodology. Given a set of tips about an entity (e.g., a restaurant), and a set of reviews about the same entity, we seek to construct a *readable*, *compact* and *representative* summary of the tips, using the review text. We assume that each review can be split into multiple coherent *snippets*, e.g., paragraphs. The summary we construct will be a collection of snippets (possibly from multiple reviews) that best capture the information content of the tips.

The representativeness and compactness objectives are often conflicting. A highly representative summary contains more and longer snippets, making it less compact. A highly compact summary may under-represent the information content of the tips. To model this trade-off holistically, in Section 3.2, we formulate our problem within the Minimum Description Length (MDL) framework, where we view the tips as being encoded by the snippets, and we seek to find a collection of snippets that produce the encoding with the minimum number of bits. We show that finding the optimal summary is NP-hard (Section 3.3). We establish a connection between our problem and the Uncapacitated Facility Location Problem, and show that there exists an algorithm with $(1 + \log n)$-approximation ratio for n tips. We also consider different heuristic algorithms for optimizing the MDL cost (Section 4). In Section 5, to investigate the efficacy of our algorithms, we compare them empirically to several baselines on real data from Foursquare and Yelp, in terms of representativeness, compactness, and readability.

Contributions. In this work, we make the following contributions. *First*, we introduce the problem of micro-review summarization, by synthesizing a summary from review snippets. We formulate constructing a compact and representative summary as a novel combinatorial optimization problem within the MDL framework. *Second*, we prove that finding the optimal summary is NP-hard. We show that our problem can be formulated algorithmically as an instance of Uncapacitated Facility Location Problem, for which there exists a $(1 + \log n)$-approximation greedy algorithm. *Third*, in addition to the greedy algorithm, we consider several heuristic algorithms. We demonstrate their efficacy on real-life datasets with respect to existing reviews, as well as summaries generated by existing techniques.

2. RELATED WORK

We now relate our work to the existing literature, broadly categorized into works that focus on summarization, mining reviews, and minimum description length.

Document Summarization. The objective of document summarization is to reduce one or more text documents into a compressed text. Broadly speaking, there are two categories of approaches. The first consists of *extractive methods*, which select text snippets (e.g., sentences, paragraphs) from the documents to be summarized. This selection may be based on clustering [20], or ranking [17]. The second consists of *abstractive methods*, which build some form of semantic representation, and then generate new or edited pieces of text to express that representation [5, 4].

If we consider micro-reviews simply as text, our problem can be seen as an instance of document summarization. Our work is related to extractive summarization, but with a key difference that we select text snippets, not from the corpus to be summarized (i.e., tips), but from an independent corpus of a different type (i.e., reviews). To validate our approach, we will compare against exemplars of both extractive (i.e., MEAD [20]) and abstractive (i.e., OPINOSIS [5]) methods in our experiments (see Section 5).

There are also other forms of summaries. For instance, reviews may be summarized in terms of the statistical distributions of sentiments for various features or aspects [9, 27], or by listing just the "key phrases" [6, 16]. These are orthogonal directions to our goal of producing a flowing text as a summary.

Review Mining. Our work is related to several lines of work in mining reviews. While we synthesize a "review" to create a summary of micro-reviews, there is a previous effort to create a synthetic review [23] to simulate a *fake review*. *Review ranking* seeks to rank reviews based on some notion of "quality" [14]. *Review selection* [12] seeks to select a specified number K of reviews based on some criteria, and it is commonly formulated as a variation of the maximum coverage problem [24, 11].

There exists previous work on review selection for covering micro-review content [18]. The problem of review selection is fundamentally different from micro-review summarization. Review selection imposes the restriction of selecting multiple *full* reviews from the existing corpus, while summarization aims at creating a single piece of text. Furthermore, in review selection, any individual review is not necessarily representative of all the points raised by tips, while the set of reviews as a whole is not necessarily compact, since the same points may be covered in multiple reviews. By synthesizing a "review", we do not run into the above issues, as it is not necessary to select whole reviews, but instead only the snippets that best represent the micro-reviews. To validate this point, in the experiments (see Section 5), we will compare to the review selected by [18], and more generally to all existing reviews in the dataset as well.

While reviews have been studied extensively, relatively little attention has been paid to micro-reviews. Most of the previous work on Foursquare, for example, look at it as a location-based social network, paying attention to the factors of locations and social links, rather than to the textual content of the micro-reviews. For instance, the aspects that have been studied include geographic analysis [19] and location-based recommendation [25].

Minimum Description Length. Minimum description length (MDL), introduced by Rissanen [21], is a well-established principle for model selection [7]. MDL itself is a general framework. The specification of the model space, and the manner in which the model describes the data, vary across applications. For instance, [1] employs MDL to model the interaction between two types of objects (expressed as an adjacency matrix) to find cross-associations. We also employ MDL to model the "interaction" between review snippets and micro-reviews, but our objective is different in selecting review snippets that summarize micro-reviews.

The optimization problem we define within the MDL framework is an instance of the Uncapacitated Facility Location Problem (UFLP) (see Section 3.3). There are also works on metric UFLP [22, 10, 13], but the metric assumption does not apply to our case.

3. PROBLEM FORMULATION

In this section, we formulate the micro-review summarization problem as a combinatorial optimization problem within the Minimum Description Length framework.

3.1 Preliminaries

Given a specific entity of interest (e.g., a restaurant), we are given as input a set of n micro-reviews (or tips) $\mathcal{T}$ for that entity. Each tip $t \in \mathcal{T}$ is modeled as a bag of words $\{w_1, w_2, \ldots, w_{|t|}\}$, where each word is drawn from a vocabulary W. This vocabulary is the universe of all the terms that appear in any tip or review.

In addition, we are given a set of m full-text reviews $\mathcal{R}$ for the same entity. We view each review $R \in \mathcal{R}$ as a collection of *snippets* $\{r_1, r_2, \ldots, r_{|R|}\}$. Each snippet $r \in R$ is a contiguous piece of text within R. In this work, we treat each review paragraph as a snippet. Snippets of different granularity can also be defined, such as, sentences, or text windows of a pre-specified length. We opt to work with paragraphs because they correspond to thematic units of variable length defined by the author herself, which are usually self-contained and discuss a coherent atomic idea of the author. Similar to tips, each snippet r is modeled as a bag of words drawn from the vocabulary W. The union of snippets from all reviews in $\mathcal{R}$ is denoted $\mathcal{U}_{\mathcal{R}}$.

A summary S is a set of review snippets, i.e., $S \subseteq \mathcal{U}_{\mathcal{R}}$. Given $\mathcal{T}$ and $\mathcal{R}$, our objective is to find the "best" summary of $\mathcal{T}$. Customarily, a summary is good if it can represent the underlying content being summarized (representativeness), and it can do so with a significantly shorter length than the full content (compactness). The two requirements, representativeness and compactness, are inherently conflicting. A longer summary may capture the underlying content better than a shorter summary. However, a summary that is too long is no longer a "summary". The goal is to find a "sweet spot" that balances the representativeness and compactness in a holistic way so as to obtain the best possible summary.

3.2 Problem Definition

To identify this "optimal" summary, we turn to Minimum Description Length (MDL) [21], a parameter-free framework for model selection. MDL deals with the issue of how to choose a model that can describe the data as concisely as possible [7]. A very complex model may be able to describe the data concisely, but the model itself would be very expensive to describe. In contrast, a simple model is easy to describe, but then describing the data becomes expensive. Importantly, MDL is parameter-free. It automatically determines the best model that balances both the cost of the model and the cost of describing the data using that model.

In our case, the data to describe are the tips in $\mathcal{T}$. A model is a summary $\mathcal{S}$, consisting of a collection of review snippets, and an assignment of each tip to one of the selected snippets. The snippet describes, or summarizes, the tips assigned to it. Let S denote the set of snippets in the summary, and let T_r denote the set of tips assigned to a snippet $r \in S$. The summary $\mathcal{S}$ is defined as the pair $\mathcal{S} = (S, \{T_r\}_{r \in S})$.

The quality of a solution is evaluated by a cost function $cost(\mathcal{T}, \mathcal{S})$ which is the cost to describe the data in $\mathcal{T}$ using the model $\mathcal{S}$. This cost function is decomposed into two parts: the model cost $model(\mathcal{S})$ which is the cost to describe the model $\mathcal{S}$, and the data cost $data(\mathcal{T}|\mathcal{S})$ which is the cost to describe the data in $\mathcal{T}$ given the model $\mathcal{S}$. A solution with low cost balances between having a complex descriptive model (high model cost) which describes accurately the data (low data cost), and a simple model (low model cost) which yields a complex description of the data (high data cost).

MDL has a natural information-theoretic interpretation, as a lossless encoding mechanism for the underlying data. The MDL cost function can be interpreted as the cost of communicating the data between two parties. In this case, the sender sends the model to the receiver, and then the description of the data using the model. The cost is computed as the number of bits needed to transmit the data.

For our problem, we are interested in encoding documents, which are "bags of words", that is, multisets of words. Any single document, or any corpus (collection) of documents, $\mathcal{D}$, defines a *language model* $M_D = (D, P_D)$, which consists of the vocabulary D of the document, and a probability distribution P_D over the words of the vocabulary. The probability $P_D(w)$ of word $w \in D$ is (usually) defined as the fraction of times that word w appears in $\mathcal{D}$.

It is well known in information theory [2] that given a domain D and a distribution P_D over this domain, the optimal encoding of D assigns a codeword of length $-\log P_D(w)$ to every element $w \in D$. This optimal encoding can be asymptotically achieved using the Huffmann encoding. Therefore, a language model $M_D = (D, P_D)$ defines an encoding of the words in the vocabulary D, and an encoding of words defines a language model. We will use the two interchangeably. We use $\text{bits}_D(w) = -\log P_D(w)$ to denote the length of the encoding of word w in the language model M_D. We also refer to this as the *cost* of the encoding. For a bag of words s from the vocabulary D, the cost of the encoding of s is

$$\text{bits}_D(s) = -\sum_{w \in s} \log P_D(w) = \sum_{w \in s} \text{bits}_D(w)$$

where, the sum over the set s, accounts for the multiple occurrences of the words in s.

We can now describe the MDL formulation of our problem, which describes the process of encoding and transmitting the set of tips $\mathcal{T}$ using the model defined by the summary $\mathcal{S}$. First, we assume that both the sender and the recipient already share the knowledge of the global vocabulary W and the language model (code) M_W for the vocabulary W. This model may be derived from any known corpus of the English language, but in our work we assume that it is

defined by the collection of reviews $\mathcal{R}$. Using this common information, we can define the model and data costs.

Model Cost. We begin by describing the summary $\mathcal{S} = (S, \{T_r\}_{r \in S})$ to the recipient, as follows.

1. First, we communicate the number of tips $n = |\mathcal{T}|$, which is the same for any model, and does not affect model selection.

2. We then communicate the number of snippets $k = |S|$ in the summary. Since $k \in [1, n]$, this can be done using $\log n$ bits.

3. We then communicate which tips are assigned to each snippet. For every snippet r, the tips in T_r will be transmitted together in sequence; therefore, we only need to communicate the transition points when we switch from one snippet to the next. For k snippets, there are $(k-1)$ transition points, and each transition is a value between 1 to n. This can be done using $(k-1) \times \log n$ bits.

4. Finally, we need to transmit the snippets $r \in S$. We use the model M_W to encode the snippets, resulting in $\sum_{r \in S} \text{bits}_W(r)$ number of bits.

Putting everything together, the cost for transmitting the model is computed as follows.

$$\begin{aligned} model(S) &= \log n + (k-1)\log n + \sum_{r \in S} \text{bits}_W(r) \\ &= \sum_{r \in S} (\log n + \text{bits}_W(r)) \end{aligned} \quad (1)$$

Data Cost. Given our model $\mathcal{S}$, we now encode the tips in $\mathcal{T}$ with the corresponding snippets. Let $r \in S$ denote one of the snippets. The snippet r is a bag of words, and defines a language model $M_r = (W, P_r)$. We will use this model to encode the set of tips in T_r associated to r by our model.

To compute the encoding cost for T_r, we need to address the following issue. Since the snippet r contains a subset of the words in W, for any word $w \notin r$ we have $P_r(w) = 0$ and thus the encoding cost is infinite. Therefore, we need to "smooth" the language model M_r, such that all terms in W would have non-zero probabilities. There are a number of smoothing methods [15]. We adopt the Laplace or additive smoothing, which adds $\alpha|W|$ number of word occurrences to r, and shares this count uniformly among all the words in the vocabulary. This method belongs to the class of Bayesian smoothing, specifically with uniform Dirichlet priors [26]. The smoothed generation probability of a word is as follows.

$$P_r(w) = \frac{\text{tf}_{r,w} + \alpha}{|r| + \alpha|W|} \quad (2)$$

In this equation, $\text{tf}_{r,w}$ is the number of occurrences of the word w in the snippet r, while α is the smoothing coefficient. Larger α tends towards a more even distribution over words. In the extremes, for $\alpha = 0$ we obtain the original probability, while for $\alpha \to \infty$ we obtain the uniform distribution.

Given the definition of $P_r(w)$ we can now define the encoding cost of tip t by snippet r as follows.

$$\text{bits}_r(t) = -\sum_{w \in t} \log P_r(w)$$

The encoding cost of the set of all tips is defined as follows.

$$data(\mathcal{T}|\mathcal{S}) = \sum_{r \in S} \sum_{t \in T_r} \text{bits}_r(t) \quad (3)$$

Given the definition for the model and data cost, the total cost for the summary $\mathcal{S}$ of the set of tips $\mathcal{T}$ is Equation 4.

$$\begin{aligned} cost(\mathcal{T}, \mathcal{S}) &= model(\mathcal{S}) + data(\mathcal{T}|\mathcal{S}) \\ &= \sum_{r \in S} \left[(\log n + \text{bits}_W(r)) + \sum_{t \in T_r} \text{bits}_r(t) \right] \end{aligned} \quad (4)$$

This equation clearly shows the trade-off between the model cost and the encoding cost. A greater number of snippets, or longer snippets, contribute to a more complex model $\mathcal{S}$ with higher model cost. However, it has the potential to decrease the encoding cost. Conversely, a very simple model may have a low model cost, but high encoding cost.

We are now ready to formally state our problem.

PROBLEM 1 (MICRO-REVIEW SUMMARIZATION (MIRS)). *Given a set of tips or micro-reviews $\mathcal{T}$, a set of reviews $\mathcal{R}$, find a summary $\mathcal{S}$, such that $cost(\mathcal{T}, \mathcal{S})$ is minimized.*

3.3 Complexity and Approximability

We now study the MIRS problem theoretically. We show that the problem is NP-hard. However, using a connection between MIRS and the Uncapacitated Facility Location Problem we can show that there exists a greedy algorithm with a $(1 + \log n)$-approximation ratio.

LEMMA 1. *The* MIRS *problem is NP-hard.*

PROOF SKETCH. The proof is based on a reduction from vertex cover (known to be NP-hard). Vertex cover seeks the minimum set of vertices in a graph, such that all edges in the graph are incident on at least one of the vertices in this set. In particular, we consider vertex cover on a regular graph [3]. In a d-regular graph, all vertices have degree exactly d.

We show that vertex cover on a d-regular graph $G(V, E)$ is a special instance of the MIRS problem with $\alpha = 0$. For each edge $e \in E$, we create a tip t_e, containing a unique word w_e for this edge (e.g., the edge ID). The size of the vocabulary $|W|$ is thus the same as the number of edges $|E|$. For each vertex $v \in V$, we create a review with one snippet r_v, containing d words corresponding to the d edges of v.

Since every vertex v has exactly d edges, correspondingly every snippet r_v contains exactly d words. Therefore, in the language model M_{r_v} we have $P_{r_v}(w_e) = 1/d$ for any word $w_e \in r_v$. This means that using any r_v to encode t_e, when e is incident on v, requires a constant number of $\text{bits}_{r_v}(t_e) = \log d$ bits. Since $\alpha = 0$, it costs infinitely high for r_v to encode t_e, when e is not incident on v.

Since every edge is incident on exactly two vertices, correspondingly every word occurs in exactly two snippets. Therefore, all words in W have the same frequency, and therefore, in the model M_W, we have that $P_W(w_e) = 1/n$ for all $w_e \in W$. In turn, this means that $\text{bits}_W(r) = d \log n$ for any r. Therefore, minimizing $cost(\mathcal{T}_E, \mathcal{S})$, where $\mathcal{T}_E$ is the set of tips corresponding to E, is equivalent to finding the set V_S with the minimum number of snippets (vertices) that collectively contain all the words (cover all the edges). □

Since the MIRS problem is NP-hard, we look for algorithms with known approximation guarantees. We can prove the following lemma.

LEMMA 2. *There exists a* $(1+\log n)$*-approximation algorithm for the* MIRS *problem.*

PROOF. We will prove the lemma by showing that the MIRS is an instance of the UNCAPACITATED FACILITY LOCATION PROBLEM (UFLP) [8]. For UFLP, we are given a set of facilities $\mathcal{U}_\mathcal{R}$ and a set of customers $\mathcal{T}$. We also know the cost f_r for opening each facility $r \in \mathcal{U}_\mathcal{R}$, as well as the cost c_{rt} to serve customer $t \in \mathcal{T}$ from facility r. The goal is to determine which subset of facilities to open (i.e., $y_r = 1$ if facility r is opened, and 0 otherwise), and which customers to service from each opened facility (i.e., $x_{rt} = 1$ if customer t is serviced from facility r, and 0 otherwise), so as to minimize the total cost $\sum_{r\in\mathcal{U}_\mathcal{R}} [f_r \cdot y_r + \sum_{t\in\mathcal{T}} c_{rt} \cdot x_{rt}]$.

It is easy to see that in the case of MIRS, the snippets are the facilities, and the tips are the customers. The cost of opening a facility r is the cost of encoding the review snippet r: $f_r = \log n + \text{bits}_W(r)$. The cost of servicing a customer t at facility r is the encoding cost of a tip t using the snippet r: $c_{rt} = \text{bits}_r(t)$. Here, $y_r = 1$ if $r \in S$, and $y_r = 0$ if $r \notin S$; $x_{rt} = 1$ if $t \in T_r$, and $x_{rt} = 0$ if $t \notin T_r$.

There is a body of work on approximation algorithms for the UFLP problem [22, 10], however most work is focused on the case where the service cost, c_{rt}, between customers and facilities defines a distance metric. This does not apply to MIRS, where $\text{bits}_r(t)$ is not metric (it is easy to see that it is not even reflexive). One known approximation algorithm for the non-metric UFLP is the greedy algorithm for MINIMUM WEIGHT SET COVER (MWSC) [8]. We describe the algorithm in Section 4.1. This algorithm has a provable approximation ratio of $1 + \log n$, where n is the number of tips, or customers. □

4. ALGORITHMS

We now propose algorithms for the MIRS problem. We assume that the encoding cost of every snippet f_r, $\forall r \in \mathcal{U}_\mathcal{R}$, and the encoding cost of any tip using any snippet c_{rt}, $\forall r \in \mathcal{U}_\mathcal{R}, t \in \mathcal{T}$ have been pre-computed. The output of the algorithms is a summary $\mathcal{S} = (S, \{T_r\}_{r\in S})$.

4.1 Greedy Synthesis

This is the approximation algorithm for the non-metric UFLP, which finds a solution to an instance of the MINIMUM WEIGHT SET COVER (MWSC) problem. The MIRS can be cast as an instance of MWSC, as follows. For every pair (r, T_r), consisting of a snippet $r \in \mathcal{U}_\mathcal{R}$ and a subset of tips $T_r \subseteq \mathcal{T}$, we define a set that "covers" the elements in T_r, with weight $f_r + \sum_{t\in T_r} c_{rt}$. Solving MWSC by finding the sub-collection of all such sets that cover all the tips in $\mathcal{T}$ with the smallest total weight also provides a solution to the corresponding MIRS instance. While enumerating all possible pairs of (r, T_r) explicitly may be intractable, [8] shows that, for each r, it is sufficient to consider those pairs (r, T_r^k), for $k = 1, \ldots, |\mathcal{T}|$, where T_r^k denotes the first k tips in a linear order of non-decreasing c_{rt}.

The pseudocode of the GREEDY SYNTHESIS algorithm is shown in Algorithm 1. In each step, we pick the pair (r, T_r) that is most effective, i.e., having the lowest average cost (line 3 in the algorithm). This can be done in $O(|\mathcal{U}_\mathcal{R}| \times |\mathcal{T}|)$ time. Once such a pair is identified, r is included in the output S, and the tips in T_r are removed from further consideration (line 4). This process is repeated until all the tips in $\mathcal{T}$ have been covered. Finally, we assign each tip to the "closest" snippet in S with the lowest encoding cost. This step is needed since the greedy selection may not have associated a tip with the lowest encoding cost snippet in S.

Algorithm 1: Greedy Synthesis

```
1  Initialize S = ∅; T = 𝒯; U = 𝒰_ℛ
2  while T ≠ ∅ or U ≠ ∅ do
3      Find the pair (r, T_r), where r ∈ U and T_r ⊆ T,
       which minimizes (f_r + Σ_{t∈T_r} c_rt) / |T_r|
4      Update S = S ∪ r; U = U \ r; and T = T \ T_r
5  return S and ∀r ∈ S, T_r = {t ∈ 𝒯 | r = arg min_{r'∈S} c_{r't}}
```

Algorithm 2: Partitional Synthesis

```
1  S_1 = {r}, where r = arg min_{r'∈𝒰_ℛ} f_{r'} + Σ_{t∈𝒯} c_{r't}.
2  C_1 = cost(S, 𝒯)
3  for k = 2, ..., |𝒯| do
4      Let S_k be k random snippets drawn from 𝒰_ℛ.
5      repeat
6          for r ∈ S_k do
7              T_r = {t ∈ 𝒯 | r = arg min_{r'∈S_k} c_{r't}}.
8          for T_r do
9              r* = arg min_{r'∈ℛ} f_{r*} + Σ_{t∈T_r} c_{r*t}.
10             replace r with r* in S_k.
11     until C_k = cost(S_k, {T_r}_{r∈S_k}) does not change
12     if C_k > C_{k-1} then
13         break
14 return S_{k-1} and {T_r}_{r∈S_{k-1}}
```

4.2 Partitional Synthesis

In GREEDY SYNTHESIS, the snippets already selected affect the choice of the next snippet, but previous decisions are never reconsidered or changed. We now consider a heuristic that considers the solution that tries to identify a local minimum in the MDL cost function. The heuristic is motivated by the observation that given a summary with k snippets, the assignment of tips to the snippets defines a partition of the tips into k clusters. The intuition is to search the space of possible tip partitions and snippet selections to find one with the lowest MDL cost.

This algorithm, which we name PARTITIONAL SYNTHESIS, is described by Algorithm 2. It considers different values for k (the number of snippets), starting from $k = 1$ and going potentially up to n. We try to find the best solution with k snippets through an iterative process reminiscent of k-means clustering. Starting with a random selection of k snippets, we assign each tip to the snippet that best encodes it (lines 6–7 of the algorithm). In turn, for each collection of tips, we find the snippet that encodes this collection with the lowest cost (lines 8–10 of the algorithm). This iterative process is conducted until the total cost does not further improve, thus reaching a local optimum. To ensure that we do not select a poor solution due to bad choice of the initial snippets, for a given k we repeat the process with M random initializations, and we pick the best solution. We keep increasing the value of k as long as we obtain a solution with a lower MDL cost. If for some k there is no further improvement, we terminate the algorithm and return

the current best solution. The complexity of this process is linear with respect to its variables, i.e., $O(k \times M \times |\mathcal{U}_{\mathcal{R}}| \times |\mathcal{T}|)$.

4.3 Hierarchical Synthesis

Motivated by the parallels between our summarization problem and clustering, we consider an algorithm that constructs a partition of the tips in a top-down hierarchical fashion. This algorithm, which we call HIERARCHICAL SYNTHESIS, is described in Algorithm 3. Starting from an existing number of partitions (initially 1), we split an existing partition into two. To determine which partition to split, we rank the existing partitions in decreasing order of average encoding cost. We then try to split the highest-ranked partition T_r associated with snippet r (lines 5–6 of the algorithm). The split is conducted using the PARTITIONAL SYNTHESIS as a subroutine with $k = 2$ (line 7). If the split is sucessful, resulting in a lower cost, we replace r with the two new snippets r_1 and r_2, and proceed to the next iteration (lines 8–10). Otherwise, we try to split the next highest-ranked snippet/partition that has not been tried. If none of the existing partitions can be split to improve the cost, the algorithm terminates and returns the current best solution. The complexity of this algorithm is similar to PARTITIONAL SYNTHESIS, but in practice it is faster, since when going from $k-1$ to k, we only need to split one partition into two.

Algorithm 3: Hierarchical Synthesis

```
1  S_1 = {r}, where r = arg min_{r' ∈ U_R} f_{r'} + Σ_{t∈T} c_{r't}.
2  C_1 = cost(S, T)
3  for k = 2, ..., |T| do
4      repeat
5          Let r be the next un-tried snippet in S_{k-1} with
           highest (f_r + Σ_{t∈T_r} c_{rt}) / |T_r|.
6          Let T_r be {t ∈ T | r = arg min_{r' ∈ S_{k-1}} c_{r't}}.
7          Find new snippets r_1 and r_2 using PARTITIONAL
           SYNTHESIS to split T_r into 2 partitions.
8          if successful split then
9              S_k = (S_{k-1} \ r) ∪ {r_1, r_2}
10             break
11     until all snippets in S_{k-1} have been tried
12     if no split then
13         break
14 return S_{k-1} and {T_r}_{r ∈ S_{k-1}}
```

5. EXPERIMENTS

Our objective is to investigate the effectiveness of our methodology in producing summaries that are representative, compact, and readable. We note that computational efficiency is not a major concern, as this is expected to be an offline batch operation, and the proposed heuristics are efficient. GREEDY SYNTHESIS completes in seconds on a machine with Intel Xeon CPU @ 2.90GHz. PARTITIONAL SYNTHESIS and HIERARCHICAL SYNTHESIS with a hundred random initializations complete in a few minutes. If necessary, these random trials are embarrassingly parallelizable.

5.1 Dataset

As input, we require paired sources of micro-reviews and reviews concerning the same entities. For this, we turn to Foursquare and Yelp. For reviews, we crawl Yelp to collect all the reviews of the top 110 restaurants in New York City with the most number of reviews as of March 2012. For micro-reviews, we crawl Foursquare to collect all the tips of the same 110 restaurants. Because some restaurants in Foursquare have too few tips, we filter out 8 restaurants with less than 50 tips each, and retain the remaining 102 restaurants for experiments. The statistics of this dataset are shown in Table 1. On average, a restaurant has 145 tips. Meanwhile, the average number of reviews per restaurant is 947. Since each review contains multiple snippets (i.e., paragraphs), it results in an average of 3K snippets per restaurant. Each restaurant constitutes a distinct instance of the micro-review summarization problem.

	Min	Max	Average	Median
#tips	51	498	145	133
#reviews	584	3,460	947	782
#snippets	1,263	12,298	3,117	2,612

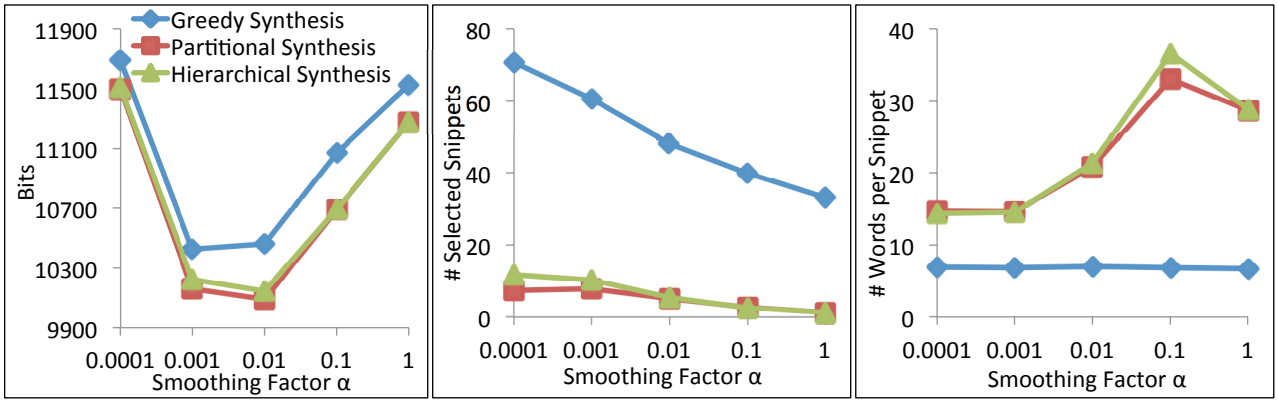
Table 1: Statistics of 102 Restaurants in the Dataset

5.2 Comparison of Proposed Algorithms

We first compare the performance of the three proposed algorithms in Section 4, both in terms of the MDL cost optimization, as well as in terms of the nature of snippets selected, for different values of the smoothing factor α.

Figure 1(a) shows the average MDL cost per restaurant (in bits) achieved by each algorithm. Fewer bits are better. PARTITIONAL SYNTHESIS and HIERARCHICAL SYNTHESIS achieve lower (better) MDL costs than GREEDY SYNTHESIS. The first two approaches are heuristics that explore the solution space by adjusting the selected snippets and tip assignments to lower the MDL cost. In contrast, GREEDY SYNTHESIS selects the snippets one at a time, each time selecting the best snippet in terms of the MDL cost. Since every snippet selection is final, GREEDY SYNTHESIS cannot lower its cost by changing a previously picked snippet. PARTITIONAL SYNTHESIS is also slightly better than HIERARCHICAL SYNTHESIS, as the former has more flexibility in exploring the space of possible partitions for finding the best one, while HIERARCHICAL SYNTHESIS is restricted to always splitting one partition into two at any one time.

We then examine the selected snippets, and we observe that there is a qualitative difference in the kinds of snippets selected by the different algorithms. Figure 1(b) shows the average number of snippets picked by each algorithm, while Figure 1(c) shows the average length of those snippets in terms of the number of words. GREEDY SYNTHESIS picks many more snippets, but those snippets tend to be shorter. At each step, GREEDY SYNTHESIS selects the snippet that can encode a number of tips with the smallest average cost. The model cost of a snippet is effectively amortized over the number of tips covered (line 3). Therefore, the tendency is to pick very short snippets, whose cost can be averaged across a small number of tips. As a result, GREEDY SYNTHESIS has to pick many of these short snippets to encode all the tips. In contrast, PARTITIONAL SYNTHESIS and HIERARCHICAL SYNTHESIS consider the cost of the summary as a whole, instead of looking at each snippet independently. This results in a solution with fewer snippets that are more substantial (longer), and can encode multiple tips.



a. MDL Cost b. Number of Selected Snippets c. Average Length of Snippet

Figure 1: Comparison of Proposed Synthesis Algorithms

We observe that for all algorithms, as α increases, initially the MDL cost decreases, and then increases again. On one hand, with a smaller α, more bits are required to encode a word in a tip that is "missing" from the corresponding snippet. Therefore, the tendency is to pick more snippets, so that at least one snippet would contain some rare words that appear in a tip. With more snippets, each snippet only needs to represent a small number of tips, favoring shorter snippets that are more similar to tips. On the other hand, with a larger α, fewer bits are required to encode a "missing" word. The tendency is to pick fewer snippets that can represent more tips, which lowers the model cost, but increases the encoding cost. The trade-off between encoding and model costs as α changes causes the U-shaped trend in Figure 1(a).

The best α seems to be 0.01, where PARTITIONAL SYNTHESIS and HIERARCHICAL SYNTHESIS reach the minimum, and GREEDY SYNTHESIS is close to the minimum. Subsequently, we will use $\alpha = 0.01$ as the default value.

5.3 Comparison with Existing Reviews

To validate the utility of synthesizing a "review", instead of just selecting one of the existing reviews, we now compare the summaries produced by our algorithms against the collection of existing reviews in the dataset.

Representativeness. To evaluate representativeness, we map it to the notion of relevance in IR. Intuitively, a good summary should be a highly relevant document to any of the tips (when the latter are used as a queries). We thus propose to evaluate representativeness within a retrieval framework. We create a corpus consisting of the reviews and the summary we want to evaluate, and we use the tips as queries against this corpus. We consider our summary to be good if it is highly ranked for most of the tips. For the IR component in our evaluation, we adopt the vector space model [15]. Each document in our corpus (i.e., a review, or the summary) is represented by a $tf \cdot idf$ vector, with dimensionality equal to the vocabulary size. The tf value of a word is the count of occurrences of the word in the document. The idf is defined as $\log \frac{N}{df}$, where N is the total number of reviews, and df is the number of reviews that contain the word. Each query (i.e., a tip) is also represented by a $tf \cdot idf$ vector, where idf is derived from reviews. If a query term does not appear in any review, its idf is set to $\log N$, as if $df = 1$. The relevance of a document to a query is the cosine similarity between their $tf \cdot idf$ vectors.

For each restaurant, we issue every tip in turn as a query, and assign a rank to every document. We order the documents according to their average rank, and then compute the *representativeness score*, which is expressed as percentile rank. The best document will have 100%, which implies that it outperforms all the other reviews.

Table 2 shows the percentile rank of our summaries, as compared to all existing reviews. In this experiment we construct the corpus for each restaurant by adding the summary to be evaluated together with the reviews for this restaurant. The percentile rank is averaged across all restaurants in the dataset. Table 2 shows that our three algorithms produce summaries with very high percentile ranks, around 99.9%. The last column of the table shows the percentage of restaurants for which our summaries obtain the highest rank. Both GREEDY SYNTHESIS and PARTITIONAL SYNTHESIS have higher representativeness scores than all existing reviews for 97% of the restaurants, whereas HIERARCHICAL SYNTHESIS obtains 93%. Since our summaries are at the top most of the time, this explains why the above percentile ranks are very close to 100% (top rank).

While our summaries outperform the existing reviews for the vast majority of query tips, it is also instructive to see how other methods of selecting a review perform on the same task. The middle three rows of Table 2 are different ways to identify the "best" review. In this case the representativeness score is computed for a corpus consisting of only the reviews for a restaurant, not including the summaries.

Lowest MDL Review selects the review with the lowest MDL score. This review scores very high percentile rank of almost 93%, which is still lower than our summaries, validating the need for synthesizing a "review", instead of selecting just one review. Its very high percentile rank also validates our MDL formulation in identifying a good review.

EffMaxCover Review is the review selected by the algorithm in [18], where the goal is to select the review that covers as many tips as possible, subject to an efficiency constraint (we follow the same settings as used in [18]). While it still attains a high percentile rank of 79%, it does not perform as well as our summaries. This is expected as it is designed for a different problem (review selection) with a different concern (efficiency constraint).

Method	Representativeness (percentile rank among reviews)	Highest Rank (percentage of restaurants)
Greedy Synthesis	99.97%	97.06%
Partitional Synthesis	99.99%	97.06%
Hierarchical Synthesis	99.89%	93.14%
Lowest MDL Review	92.94%	5.88%
EffMaxCover Review	79.36%	0.98%
Most Useful Review	60.76%	0.00%
Shortest Review	8.97%	0.00%
Median Review	51.91%	0.00%
Longest Review	84.33%	5.88%

Table 2: Comparison with Reviews: Representativeness

Method	Compactness	
	(# words)	(percentile rank among reviews)
Greedy Synthesis	337.5	8.8%
Partitional Synthesis	104.8	52.9%
Hierarchical Synthesis	114.9	49.7%
Lowest MDL Review	66.3	51.0%
EffMaxCover Review	114.7	50.5%
Most Useful Review	327.0	19.5%
Shortest Review	1.6	100.0%
Median Review	106.3	50.3%
Longest Review	833.2	0.2%

Table 3: Comparison with Reviews: Compactness

Most Useful Review selects the review with the highest usefulness votes given by Yelp users. It has relatively low percentile rank of around 60%. There are many factors affecting how users cast their usefulness votes, which are not always correlated to the comprehensiveness of the review, which could explain the low representativeness score.

For completeness, we also include several reviews selected based on length (number of words) alone. The results are quite expected. *Shortest Review* is not representative, with low percentile rank of around 9%. Unsurprisingly, *Median Review* with median length also has percentile rank close to the median, around 52%. Helped by its length, *Longest Review* has high representativeness score of 84%, but this comes at the cost of the compactness.

Compactness. Another concern is compactness. This can be measured in a more straightforward manner, by counting the number of words. We assign each review a percentile rank, which measures the percentage of reviews are at least as long (no better) as the review at hand. The last three rows of Table 3 show that, as expected, *Shortest Review* has 100% percentile rank (most compact), with only 1.6 words on average. Some reviews contain only one or two words (e.g., "*Amazing!*"). *Longest Review* has 833 words (least compact), whereas *Median Review* has 106 words.

Our objective is not to create the shortest summary (which is trivial), but rather a representative summary of short length. A reasonable target is to create a summary of length comparable to the median length. Table 3 shows that indeed both Partitional Synthesis and Hierarchical Synthesis produce summaries that are very close to the median length. Partitional Synthesis is slightly shorter, with 104.8 words, whereas Hierarchical Synthesis is slightly longer, with 114.9 words. As previously explained, Greedy Synthesis generates many more snippets, resulting in a longer summary of 337.5 words. *EffMaxCover Review* is also around the median, whereas *Lowest MDL Review* is more compact, and *Most Useful* review is longer.

5.4 Comparison with Baselines

We now compare our summaries with those generated by existing text summarization methods, which summarize the tips directly without relying on reviews.

As baselines, we compare against two popular methods, for which a public implementation is available: Opinosis[2] [5], which is an example of abstractive summarization, and Mead[3] [20], which is an example of extractive summarization. For both, we use their default settings. As in [5], for Mead, we turn off the effect of sentence position in text, which is not relevant to our case. Both Opinosis and Mead require as input the expected length of the summary. For a fair comparison, we use the length of the summary produced by Partitional Synthesis, our best-performing technique, as an input parameter to Opinosis and Mead. For Mead, we specify the same number of words. Opinosis outputs a ranked set of sentences that meet some criteria, so we create a summary by selecting the top sentences until we reach the word threshold, or until we exhaust all the sentences.

For completeness, we include two versions of our algorithm based on partitional synthesis. The original Partitional Synthesis uses review snippets to summarize tips. Another variant, which we call Partitional Synthesis with Tips, does not rely on reviews at all, and instead creates a summary using tips (as snippets) to represent tips.

Benchmarking against Reviews. First, we conduct the same experiment as in Section 5.3. Table 4 shows their representativeness scores. Evidently, the Partitional Synthesis based on review snippets is the best. It is better than the tip-based variant, which suggests that using review snippets is more effective than using tips only.

Both our variants are better than the baselines. Opinosis has percentile rank of around 86%, and achieves the top rank only for 23% of the restaurants. Mead has slightly better percentile rank, with 90%, but worst top rank, with 12%.

Method	Representativeness (percentile rank among reviews)	Highest Rank (percentage of restaurants)
Partitional Synthesis	99.99%	97.06%
Partitional Synthesis with Tips	98.86%	57.84%
Opinosis	86.01%	23.53%
Mead	90.78%	12.75%

Table 4: Comparison with Baselines: Representativeness (with respect to reviews)

Table 5 shows the comparison in terms of compactness. As expected, Partitional Synthesis and Mead have very similar lengths, both around 100 words. Partitional Synthesis with Tips produces slightly shorter summary for the same setting of $\alpha = 0.01$. Opinosis is much shorter, with around 42 words. This is because Opinosis tends to generate very short sentences. Even after we use all the output sentences, we may not attain the same length as the others. To show that Opinosis is not disadvantaged, we

[2] `kavita-ganesan.com/opinosis-summarizer-library`
[3] `http://www.summarization.com/mead/`

also show the average number of sentences for various methods. Opinosis uses the most sentences, but because they are shorter, it results in fewer words overall. The behavior and the performance of Opinosis is reasonable, since its main focus is on generating short blurbs from very similar sentences, rather than a full-fledged summary.

Method	Compactness		#sentences
	(# words)	(percentile rank among reviews)	
Partitional Synthesis	104.8	52.9%	9.2
Partitional Synthesis with Tips	82.8	61.6%	8.2
Opinosis	41.6	83.6%	10.8
Mead	104.5	53.3%	5.4

Table 5: Comparison with Baselines: Compactness (with respect to reviews)

Head-to-head Comparison. The previous comparison is indirect, since it compares each method against all reviews, but not against each other. Here, we perform a head-to-head comparison, by repeating the same retrieval experiment for the different summarization techniques. In this case the corpus consists of only the summaries generated by Partitional Synthesis, Partitional Synthesis with Tips, Opinosis, and Mead, and for each query (tip), we rank these four summaries. Table 6 shows the comparison in terms of representativeness.

Partitional Synthesis has the highest percentile rank with 93%, as compared to 69% for Partitional Synthesis with Tips, which in turn has higher percentile rank than the baselines Opinosis with 47% and Mead with 41%. Partitional Synthesis emerges at the top rank for 75% of restaurants, significantly higher than the other methods.

Method	Representativeness (percentile rank among reviews)	Highest Rank (percentage of restaurants)
Partitional Synthesis	92.89%	75.49%
Partitional Synthesis with Tips	69.36%	18.63%
Opinosis	47.30%	3.92%
Mead	40.69%	1.96%

Table 6: Comparison with Baselines: Representativeness (head-to-head)

Readability. A summary is ultimately meant for human consumption. We thus want to evaluate the readability of our summaries. Since there is no good way to assess readability automatically, we rely on a user study. For this study, we use the 20 restaurants with the highest number of tips. We use five human judges, who are not related to this work, and for each restaurant, we show each human judge the four summaries by Partitional Synthesis, Partitional Synthesis with Tips, Opinosis, and Mead respectively, in random order, without identifying the methods. Each human judge is requested to give a rating from 1 to 5 to each summary, where 1 (lowest) indicates a badly-written piece of text that is not readable, and 5 (highest) indicates a very well-written piece of text that is highly readable. We then compute the average rating given by the judges.

Table 7 shows the readability scores of the four methods. The score of 4.23 (out of 5) for Partitional Synthesis indicates that the human judges find the summaries well-written and highly readable. This is also higher than the score of 3.99 obtained by Partitional Synthesis with Tips. Paired samples t-test shows that the difference is statistically significant at 5% level.

Method	Readability Score
Partitional Synthesis	4.23
Partitional Synthesis with Tips	3.99
Opinosis	2.07
Mead	3.47

Table 7: Readability Scores

Both our variants score higher than the baselines. Mead, which selects sentences from tips, has a lower readability score of 3.47. Opinosis has a below-average score of 2.07. This is expected since its summary consists of a collection of blurbs. In addition, Opinosis relies on generating new sentences, which is a hard task.

The overall ordering between methods is consistent among all the judges. We also conduct a correlation analysis on how the independent judges agree on the ratings of individual summaries. For this, we measure the Pearson's correlation coefficient between any two judges, which ranges from -1 (anti-correlated) to 1 (perfectly correlated). Across the ten pairs of judges, the correlation coefficient ranges from 0.4 to 0.7, with an average of 0.6. These correlation values are high, indicating significant agreement between the judges.

Case Study. In Figure 2, we show example summaries for the restaurant *Eataly*[4]. The summary by Partitional Synthesis (Figure 2(a)) covers the various aspects of the restaurant: the grocery store, the food (pasta, cheeses), the wine, and the gelato. Overall it has consistency and continuity, and it successfully selects atomic snippets to cover specific aspects (e.g., the part about the gelato comes from a single snippet). The summary by Partitional Synthesis with Tips (Figure 2(b)) is compact and dense, but is not as descriptive and narrative. The summary of Opinosis (Figure 2(c)) contains useful information and keywords, but it is not presented in a flowing, articulate manner. In the summary of Mead (Figure 2(d)) the individual sentences (extracted from tips) are readable, but they tend to capture peculiarities (e.g., Jimmy Fallon, Sammy Hagar), rather than things that are pertinent to the place. The gelato is described in merely two words: *Gelato Great!*.

6. CONCLUSION

We introduce the problem of summarizing micro-reviews. Our proposed approach is to synthesize a summary from review snippets. The goal is a summary that is representative of the micro-reviews, yet compact and readable. To balance the conflicting objectives of representativeness and compactness holistically, we formulate the problem within the Minimum Description Length (MDL) framework. Minimizing the MDL cost is NP-hard. Through a connection to Uncapacitated Facility Location Problem (UFLP), we establish an approximation guarantee of $1 + \log n$, where n is the number of micro-reviews. We also propose three heuristic algorithms to solve the problem. Experiments on Foursquare and Yelp datasets show that our methodology results in highly representative and compact summaries.

[4]`https://foursquare.com/v/eataly-nyc/4c5ef77bfff99c74eda954d3`

I love this place. i come here to get anything and everything Italian. Like many others have noted, this place is confusing. It's part food court, part grocery store, part coffee shop, part bookstore. Eataly tries to be a one-stop shop for all things Italian. They've got everything you'd want for a good Italian dinner - fresh meat, seafood, pasta, cheese, etc. Loved the fresh pasta in the Pizza and Pasta section. Also great place to shop for everything Italian even if a little pricey. We put our names in at the pasta place and went to have wine while we waited. Order a bottle of wine, nibble on the cheese, bread, meats, and olives. The gelato is also really good. I got the pistachio gelato, and it was some of the best I've had. When I get gelato, I want it to really taste like the flavor that it is, and this gelato did not disappoint. Amazing. The food was great! it was very crowded, but worth it.

(a) Partitional Synthesis

Try the pizza, the pasta, the wine, everything's great here 50,000 square feet of pure Italian with a rooftop beer garden, a cooking school, bakery, coffee shop, fresh pasta counter, a butcher, & any pantry item you'll need to play chef at home. disregard the duplicate venues, this is the right one! This place is great! Some of the best gelato I've ever tasted! Find one of the only American foods in this fine Italian markets.

(b) Partitional Synthesis with Tips

fresh pasta , butcher , any pantry item you 'll need to play chef at home. a better value.../: it 's a fun place to browse if your in the area. the bakery and with $ 2,80 you get the best onions focaccia you 've ever had. the pizza and the fettuccine con coda alla vaccinara are both superb. sunday april 3rd , renee and the derelicts redux performed great live music. cooking school , bakery , coffee shop , fresh pasta counter , butcher. they make the whole eataly experience even better and much more fun. fresh pasta. they will be back on the 17th. good food. great italian. the food is great. prime rib sandwich. artisanal italian food and wine marketplace. gelato is amazing. cooking school , coffee. redux performed great. great live music. new york. hard to move. authentic italian. hot chocolate. amazing food. delicious food. many people. fresh vegetables. amazing place. reasonable prices.

(c) Opinosis

send your photos of food to: posteat.ly check out beer garden 50,000 square feet of pure Italian with a rooftop beer garden, a cooking school, bakery, coffee shop, fresh pasta counter, a butcher, & any pantry item you'll need to play chef at home. Go to the pasta restaurant and get the cracked pepper pa Try the entrance on 23rd to avoid the line to get in crazy place with amazing Italian food & products. Amazing place for a seafood, charcuterie, cheeses, caviar, wine & champagne lunch Can be obez at this place I love everything about this place...panini, piazza for the cheese board, lavazza espresso bar, beer garden, the market...their homemade mozzarella is amazing! too crowded in lunch time, but it worth it. at least try the gelato for desert after having a bite at one of the food tents of the mad square park.

(d) Mead

Figure 2: Summaries for *Eataly*

7. REFERENCES

[1] D. Chakrabarti, S. Papadimitriou, D. S. Modha, and C. Faloutsos. Fully automatic cross-associations. In *KDD*, 2004.

[2] T. M. Cover and J. A. Thomas. *Elements of information theory*. John Wiley & Sons, 2012.

[3] U. Feige. Vertex cover is hardest to approximate on regular graphs. Technical report, Citeseer, 2003.

[4] K. Filippova. Multi-sentence compression: Finding shortest paths in word graphs. In *COLING*, 2010.

[5] K. Ganesan, C. Zhai, and J. Han. Opinosis: a graph-based approach to abstractive summarization of highly redundant opinions. In *COLING*, 2010.

[6] K. Ganesan, C. Zhai, and E. Viegas. Micropinion generation: an unsupervised approach to generating ultra-concise summaries of opinions. In *WWW*, 2012.

[7] P. D. Grünwald, I. J. Myung, and M. A. Pitt. *Advances in minimum description length: Theory and applications*. MIT Press, 2005.

[8] D. S. Hochbaum. Heuristics for the fixed cost median problem. *Mathematical Programming*, 1982.

[9] M. Hu and B. Liu. Mining and summarizing customer reviews. In *KDD*, 2004.

[10] K. Jain and V. V. Vazirani. Approximation algorithms for metric facility location and k-median problems using the primal-dual schema and lagrangian relaxation. *JACM*, 2001.

[11] T. Lappas, M. Crovella, and E. Terzi. Selecting a characteristic set of reviews. In *KDD*, 2012.

[12] T. Lappas and D. Gunopulos. Efficient confident search in large review corpora. In *ECML/PKDD*, 2010.

[13] N. Lazic, B. J. Frey, and P. Aarabi. Solving the uncapacitated facility location problem using message passing algorithms. In *AISTATS*, 2010.

[14] Y. Lu, P. Tsaparas, A. Ntoulas, and L. Polanyi. Exploiting social context for review quality prediction. In *WWW*, 2010.

[15] C. D. Manning, P. Raghavan, and H. Schütze. *Introduction to Information Retrieval*, volume 1. Cambridge University Press, 2008.

[16] X. Meng and H. Wang. Mining user reviews: from specification to summarization. In *ACL-IJCNLP*, 2009.

[17] R. Mihalcea and P. Tarau. TextRank: Bringing order into texts. In *ACL*, 2004.

[18] T.-S. Nguyen, H. W. Lauw, and P. Tsaparas. Using micro-reviews to select an efficient set of reviews. In *CIKM*, 2013.

[19] A. Noulas, S. Scellato, C. Mascolo, and M. Pontil. An empirical study of geographic user activity patterns in foursquare. *ICWSM*, 2011.

[20] D. R. Radev, H. Jing, M. Styś, and D. Tam. Centroid-based summarization of multiple documents. *Information Processing & Management*, 2004.

[21] J. Rissanen. Modeling by shortest data description. *Automatica*, 1978.

[22] D. B. Shmoys, É. Tardos, and K. Aardal. Approximation algorithms for facility location problems. In *ACM Symposium on Theory of Computing*, 1997.

[23] H. Sun, A. Morales, and X. Yan. Synthetic review spamming and defense. In *KDD*, 2013.

[24] P. Tsaparas, A. Ntoulas, and E. Terzi. Selecting a comprehensive set of reviews. In *KDD*, 2011.

[25] M. Ye, P. Yin, and W.-C. Lee. Location recommendation for location-based social networks. In *SIGSPATIAL*, 2010.

[26] C. Zhai and J. Lafferty. A study of smoothing methods for language models applied to information retrieval. *TOIS*, 2004.

[27] L. Zhuang, F. Jing, and X.-Y. Zhu. Movie review mining and summarization. In *CIKM*, 2006.