Noname manuscript No.
(will be inserted by the editor)

Micro-Review Synthesis for Multi-Entity Summarization

Thanh-Son Nguyen · Hady W. Lauw · Panayiotis Tsaparas

Received: date / Accepted: date

Abstract Location-based social networks (LBSNs), exemplified by Foursquare, are fast gaining popularity. One important feature of LBSNs is micro-review. Upon check-in at a particular venue, a user may leave a short review (up to 200 characters long), also known as a tip. These tips are an important source of information for others to know more about various aspects of an entity (e.g., restaurant), such as food, waiting time, or service. However, a user is often interested not in one particular entity, but rather in several entities collectively, for instance within a neighborhood or a category. In this paper, we address the problem of summarizing the tips of multiple entities in a collection, by way of synthesizing new micro-reviews that pertain to the collection, rather than to the individual entities per se. We formulate this problem in terms of first finding a representation of the collection, by identifying a number of "aspects" that link common threads across two or more entities within the collection. We express these aspects as dense subgraphs in a graph of sentences derived from the multi-entity corpora. This leads to a formulation of maximal multi-entity quasi-cliques, as well as a heuristic algorithm to find K such quasi-cliques maximizing the coverage over the multi-entity corpora. To synthesize a summary tip for each aspect, we select a small number of sentences from the corresponding quasi-clique, balancing conciseness and representativeness in terms of a facility location problem. Our approach performs well on collections of Foursquare entities based on localities and categories, producing more representative and diverse summaries than the baselines.

Keywords Micro-review synthesis · Multi-entity summarization · Maximal quasi-clique

Thanh-Son Nguyen
Singapore Management University, Singapore
E-mail: tsnguyen.2013@phdis.smu.edu.sg

Hady W. Lauw
Singapore Management University, Singapore
E-mail: hadywlauw@smu.edu.sg

Panayiotis Tsaparas
University of Ioannina, Greece
E-mail: tsap@cs.uoi.gr

1 Introduction

Location-based social networks (LBSNs), e.g., Foursquare, Yelp Check-ins, are currently undergoing tremendous growth. Particularly, we are interested in the reviewing feature of LBSNs. The reviews generated in LBSNs, which we term *micro-reviews*, are characteristically different from regular reviews, such as found in Yelp or TripAdvisor. One fundamental difference is length. Micro-reviews are shorter by design, e.g., Foursquare imposes a limit of 200 characters to each tip[1]. Hence, they are much more concise and to-the-point than reviews. Another difference is because leaving tips is usually done on-site upon check-in, they capture the user's reaction in the moment, as opposed to well after the fact. These material differences signify micro-reviews as an important content in their own right[2].

Micro-reviews are written by users for an individual venue. Let us take for an example *Ippudo*[3] in New York. On Foursquare, users have left hundreds of tips about various aspects, about signature dishes, e.g., *"Get the pork buns (it's true; better than Momofoku) and Akamaru Modern. You will leave a very happy person."*, waiting time, e.g., *"Go in the afternoon to avoid having to wait. Sat afternoon = 15 min wait."*, or reservation, e.g., *"No advanced reservations, do take same day but must walk in to make it"*. These bite-sized morsels of information collectively provide an overall picture to users interested in this particular venue.

LBSNs are increasingly popular as a travel tool to get a glimpse of what is available in a new unfamiliar locality [10, 43, 44]. Hence, there is a need to provide micro-review-like information, not just about a specific individual venue, but also for *a collection of venues*. For instance, a tourist may find herself in a particular neighborhood, such as the Lower Eastside of New York City, and wish to know about what is available in the neighborhood. Alternatively, she may have certain dietary restrictions, e.g., halal, kosher, vegan, and wish to know about venues that serve such foods. In these scenarios, the scope is not an individual venue or entity, but rather a collection of entities defined by some concept (e.g., locality, category).

Problem. In this paper, we propose to generate or synthesize micro-reviews for a collection of entities, by summarizing the micro-reviews of the underying entities within the collection. Let us take for example the ZIP code NY 10003, which covers the Lower Eastside / East Village neighborhood in New York City. There are more than twenty restaurants in this area, each described by its own set of micro-reviews. Upon "check-in" at this ZIP code, we would like to present a user with a list of micro-reviews pertaining to the ZIP code as a whole.

To illustrate this, Table 1 shows several examples of such micro-reviews (which we call summary tips) generated by our proposed method. From the first one t_1, we see that several restaurants serve ramen, including *Ippudo*, *Republic*, and *Yakitory Taisho*. Another big thing in this locality are pork dishes, with different specialties (buns, belly, pulled, etc.) available in various venues (shown in italics). In turn, t_3 and t_4 talk about pizza and chocolate respectively. Each summary tip encapsulates some pertinent aspect about two or more entities in this ZIP code. Taken together, they provide a rounder picture of what one could find in this neighborhood.

[1] Here, we use "micro-review" and "tip" interchangeably as we are mostly using Foursquare in our running examples.

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

[3] `https://foursquare.com/v/ippudo/4a5403b8f964a520f3b21fe3`

Table 1 Example Summary Tips for ZIP code NY 10003

ID	Micro-Review (*and the corresponding relevant entities*)
t_1	Amazing pork buns and ramen. Try Akamaru Modern Ramen! Best ramen ever. The miso ramen is tops. (*Ippudo, Republic, Yakitori Taisho*)
t_2	Get the pork belly sandwich. Try the pulled-pork. Try the pork buns. Love steamed Pork buns. Pork buns are great. (*Ippudo, Wafels & Dinges, Num Pang, Momofuku Ssam Bar, (and 7 more)*)
t_3	Best pizza in NYC. Best artichoke pizza. (*ABC Kitchen, Otto Enoteca Pizzeria, Max Brenner, (and 3 more)*)
t_4	Mexican Hot Chocolate - do it! Italian thick hot chocolate. Delicious hot chocolate! Must try the Chocolate Marshmallow Pizza! (*ABC Kitchen, Otto Enoteca Pizzeria, Max Brenner, (and 7 more)*)

Given a collection of entities, and the set of input tips for each entity, we would like to derive a summary for this collection, in the form of a specified number K of "synthesized" micro-reviews or summary tips. The given collection of entities may arise due to various application scenarios. In one scenario, a user in a specified area or neighborhood (e.g., a ZIP code) may wish to summarize nearby entities. In another scenario, a user may be looking for a specified category or dietary preference (e.g., Asian restaurant, Kosher or Halal food). Alternatively, a user may also wish to summarize a collection of entities that are relevant to a certain query. In the output summary, each summary tip captures some aspect that applies to multiple entities. Collectively, the K summary tips should represent as much information about the collection as possible.

Approaches. One possible approach is to pool together the tips from all entities in the collection, and to employ traditional summarization techniques. However, this approach suffers from drawbacks. For one, the summary tips may be *individualistic*, capturing an aspect specific to one entity. For another, there may be a lack of diversity, and the summary tips may be repetitive. The results also may be skewed towards "larger" entities with many more tips than others.

We advocate a *collectivist* approach, whereby the summary tips would emphasize the common threads across entities in the set. That motivates our three-step methodology, outlined in Section 3. In the first step, we transform tip sentences into a multi-entity graph, modeling sentences in tips as vertices and content similarity as edges. In the second step, in Section 4, we find K quasi-cliques that "tie" entities together. This is advantageous, because aspects just like subgraphs may overlap in content, and the graph representation allows us to specify connectivity constraints that take into account the multi-entity structure. For the third step, in Section 5, we then synthesize a summary tip from each subgraph of highly connected sentences capturing a coherent aspect, by selecting sentences from the subgraph so as to balance representativeness and conciseness.

Contributions. We make the following contributions. *First*, we introduce the problem of generating micro-reviews for a collection of entities from the micro-reviews of the underlying entities within the collection. We formulate this as finding K maximal quasi-cliques in a graph of tip sentences so as to maximize coverage over the multi-entity corpora, followed by synthesizing a new micro-review via sentence selection within each quasi-clique that is modeled as facility location problem. *Second*, we formulate the notion of "multi-entity quasi-clique", with dual

density thresholds, for enforcing connectivity constraints within each entity as well as across each pair of entities. *Third*, we show that the problem is computationally intractable, and develop a heuristic algorithm for finding a number of maximal multi-entity quasi-cliques based on the framework of greedy randomized adaptive search procedure. We demonstrate the efficacy of our approach against comparative baselines on a Foursquare dataset consisting of 102 restaurants in New York City (Section 6), involving different collections based on locality and category. Because we do not rely on specific features that are present only in Foursquare or in New York City, the proposed technique would generalize to other sources of micro-reviews for various cities as well.

2 Related Work

In this section, we discuss the related literature, which we broadly categorize into text mining, graph mining, and review mining.

Text mining. Condensing a body of text (a single document or multiple documents) into a more compact form is known as *summarization*. There are several main methodologies. One is *extractive*, which selects existing sentences. Two of the most well-known methods are MEAD [32] and TextRank [24]. MEAD selects text snippets in an incremental manner, according to a scoring function that takes into account a snippet's similarity to the centroid, its position in a document, as well as the overlap with previously selected snippets. TextRank conducts random walks on a graph of text snippets, and selects the snippets with the highest stationary probabilities. [41] is similar, but simultaneously considers both the rankings of words and sentences. Another methodology is *abstractive*, exemplified by [12, 11], which generates new sentences from an abstract representation. They construct a graph of words, incorporating word frequencies, POS tags, and word sequence within sentences. A summary is generated by selecting paths from this graph. In experiments (Section 6), we compare to these methodologies as baselines.

In working with multiple text corpora (in our case, one per entity), we fall under *multi-corpora text analysis*. Previous works are not meant for summarization. Some are based on topic modeling. [46] models topics with a common component across the corpora, and a unique component within each corpus. [36] models topics of different granularities: local topics that discriminate between parts within a document versus global topics that discriminate among documents. Yet others are focusing on finding contrastive viewpoints [30]. The objective is to align sentences across documents, either based on similarity [34] or contradiction [16].

Graph mining. Our multi-entity quasi-clique formulation is related to the problem of finding dense subgraphs in a large graph [38]. There are different applications of dense subgraph mining, for example, joint mining of protein-protein interaction data to find frequent cliques of proteins to detect the proteins that are likely to be functionally related [14], or mining important groups in a network [4], etc. There are various definitions of dense subgraphs, including based on average degree or edge density. We adopt the definition of quasi-clique. Given the generality of quasi-clique, there are various algorithms proposed in the literature, including local search [5], branch and bound [28], pruning [45, 21], mixed integer programming [29], etc. Our objective is not to enumerate all dense subgraphs in a graph [39], but rather finding a set of top K dense subgraphs.

Generally, with regards to multi-entity summarization, there are two limitations of the current graph mining approaches. First, our problem pertains to summarization, which requires a further step beyond dense subgraph discovery. These approaches would not produce the desired output without further processing. Second, our interest is in multi-entity summarization, and therefore our definition of dense subgraph has a requirement that involves connectivity within and across entities, whereas these approaches do not have multi-entity consideration. This motivates our definition of multi-entity quasi-cliques (see Section 4).

Specifically, redundancy-aware maximal cliques or RAMC by Wang et al. [42] is particularly related. It allows finding the top-K maximal cliques with diversity. The key difference is our multi-entity consideration with intra- and inter-densities, whereas RAMC is agnostic about entities and attempts to find dense subgraphs based on connectivity alone. To investigate the utility of the multi-entity consideration, we consider RAMC as a baseline in Section 6.

Review mining. In focusing on the reviewing feature of LBSNs, we are related to the broader area of review mining. Previous works address various problems, such as ranking reviews by quality [22], selecting a small subset of representative reviews [19,37,18,25], or synthesizing a review [35,26]. The crucial distinction is their focus on reviews for a *single entity*, as opposed to our *multi-entity* scenario.

Other works studying micro-reviews in Foursquare address different purposes. [6] identifies unexpected micro-reviews for a single venue. [40] predicts which micro-reviews would be popular (high number of likes). Yet several others focus not on the reviewing aspect, but rather on other features of Foursquare as a location-based social network, such as mobility [27], gamification [20] or privacy [31].

3 Overview

In this section we introduce the notation and terminology we use in the paper, and we provide an overview of the proposed multi-entity summarization system.

Let Ω denote the universal set of all entities of a particular domain (e.g., restaurants). Each entity $e \in \Omega$ is associated with a set of tips T_e. In turn, each tip $t \in T_e$ is a set of sentences $\{s_1, s_2, \ldots, s_{|t|}\}$, each of which is modeled as a bag of words drawn from a finite universal vocabulary. Without losing generality, here we adopt Foursquare's definition for each tip to have a limit of 200 characters.

A multi-entity collection $\mathcal{C} \subseteq \Omega$ is a subset of entities. Though it may be any arbitrary set of entities, in practice we expect $\mathcal{C}$ to be defined by some meaningful conceptual boundary. For instance, one type of boundary may be ZIP codes, i.e., every ZIP code defines a collection $\mathcal{C}$ consisting of entities located within that ZIP code. Other possible boundaries include cuisine types, dietary preferences, etc. It follows that $\mathcal{C}$ is associated with a multi-entity corpora of tips $\mathcal{T}_\mathcal{C} = \{T_e\}_{e \in \mathcal{C}}$, which is the union of partitions of the respective corpus of each underlying entity.

Problem: Multi-Entity Summarization. Given a multi-entity collection $\mathcal{C}$ and its corresponding corpora of tips $\mathcal{T}_\mathcal{C}$, and an integer K, we seek to derive a summary $R_\mathcal{C}$ of the content in $\mathcal{T}_\mathcal{C}$ consisting of K *summary tips*. The tips in $R_\mathcal{C}$ are not part of $\mathcal{T}_\mathcal{C}$, but rather synthesized tips.

The summary $R_\mathcal{C}$ is meant to describe the collection $\mathcal{C}$ as a whole, rather than the individual entities within $\mathcal{C}$. Therefore, it would not do to proportionally generate $\frac{K}{|\mathcal{C}|}$ tips separately from each $T_e \in \mathcal{T}_\mathcal{C}$ and stitch them together, as the

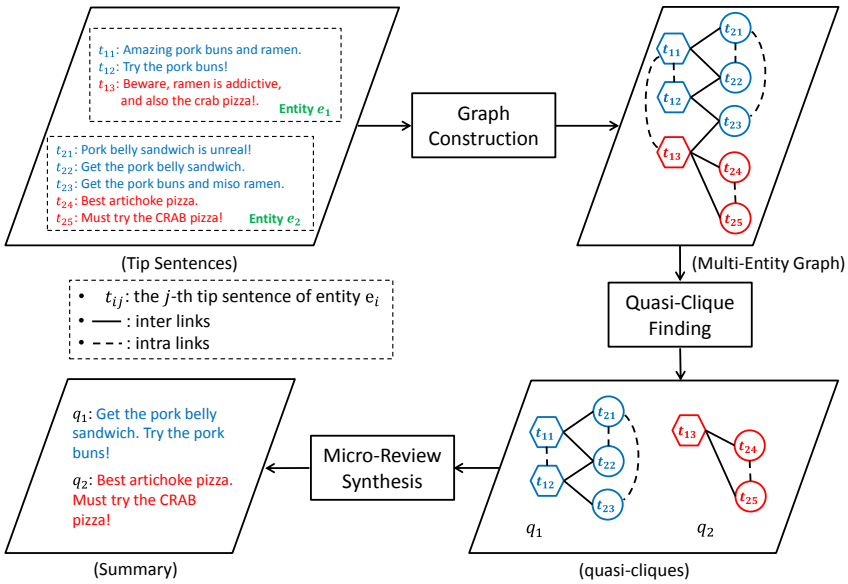

Fig. 1 Multi-Entity Summarization Framework

resulting summary might be either repetitive or incomplete (it is possible that $K \ll |\mathcal{C}|$). Rather, the summary $R_{\mathcal{C}}$ should represent the common threads among entities in $\mathcal{C}$ that define the inherent characteristic aspects of $\mathcal{C}$. Intuitively, the summary tips should cut across the different entities, rather than go deep into a single entity. Hence, we postulate that each tip in $R_{\mathcal{C}}$ should capture some aspect in common among two or more entities in $\mathcal{C}$.

System Overview. Fig. 1 illustrates the proposed multi-entity summarization framework. It consists of three main steps: The *graph construction*, the *quasi-clique finding*, and *micro-review synthesis*. The input is a set of tip sentences from different entities (e.g., restaurants). In the *graph construction* step, the tip sentences are transformed into a *multi-entity graph* where each sentence is a node, and there are edges between similar sentences (we define multi-entity graph in Section 4.1). The *quasi-clique finding* step takes the multi-entity graph as input, and finds K maximal multi-entity quasi-cliques (as defined in Definition 3). These quasi-cliques of sentences capture the common aspects across the different entities. After having the quasi-cliques, the *micro-review synthesis* step will generate a *summary tip* for each quasi-clique based on the tip sentences inside the quasi-clique. We now describe each step in more detail.

Graph Construction. In the graph construction step, we construct a multi-entity graph $G = (\mathcal{V}, \mathcal{E})$ from the input tip sentences. Let $n \geq 2$ be the number of entities (e.g., restaurants) in the collection $\mathcal{C}$. The set of vertices $\mathcal{V} = \{V_1, V_2, ..., V_n\}$ contains n partitions, one for each entity. Let $\mathcal{S}_{\mathcal{C}}$ denote the set of sentences in all tips in $\mathcal{T}_{\mathcal{C}}$. We view the sentence as the atomic unit of content. We create a vertex in the set V_i for every unique sentence of the entity e_i. Fig. 1

illustrates this using an example of two entities. t_{ij} refers to the j^{th} sentence of entity e_i. In Fig. 1, the vertices in V_1 are drawn as hexagons, and the vertices in V_2 are drawn as circles.

The set of edges $\mathcal{E}$ of G, can be partitioned into distinct subsets. We denote by $E_i \subset \mathcal{E}$ (for $i = 1, \ldots, n$), the subset of edges that connect two vertices in V_i. We call such edges connecting vertices within a partition: *intra-edges*. In Fig. 1, they are drawn as dotted lines. We denote by $E_{ij} \subset \mathcal{E}$ (for $i < j$), the subset of edges that connect one vertex in V_i to another vertex in V_j. We call such edges connecting vertices across two partitions: *inter-edges*. In Fig. 1, they are drawn as solid lines. In this work, we create an edge between two sentences, if the sentences are textually similar, that is, they use common words to describe the same aspect or characteristic of the entities. Specifically, we create an edge between two vertices if their corresponding cosine similarity is above a certain threshold (we study the appropriate threshold in Section 6). Aside from cosine similarity, there may be other approaches to determine similarity between sentences, such as based on semantics or topic modeling [3]. The framework in Fig. 1 is general in that such semantic-based similarity could be used to augment the graph construction without much changes to the subsequent steps in our framework.

Quasi-Clique Finding. This step discovers common aspects among entities in $\mathcal{C}$, by identifying quasi-cliques in the multi-entity graph. Informally, a multi-entity quasi-clique q is a subgraph of the graph containing nodes from multiple partitions that are densely connected with both intra and inter edges. The density of the intra and inter edges is controlled by two parameters α and β. We also require that the set is *maximal*, i.e., we cannot grow it into a larger quasi-clique. We discuss the exact definition of such quasi-clique in Section 4.

The set of nodes in the quasi-clique q corresponds to a set of sentences in $\mathcal{S}_\mathcal{C}$. We view this set of sentences as pertaining to some aspect that cuts across the entities in $\mathcal{C}$ (since q contains nodes from at least two entities). We say that q "covers" a sentence s if $s \in q$. We define the coverage of q as $\frac{|q|}{|\mathcal{S}_\mathcal{C}|}$. Intuitively, the larger the coverage of q, the more important an aspect it captures from $\mathcal{C}$.

Given a set of K quasi-cliques $Q = \{q_1, q_2, \ldots, q_K\}$, we define the coverage of Q as follows.

$$Cov(Q) = \frac{|\bigcup_{q \in Q} q|}{|\mathcal{S}_\mathcal{C}|} \tag{1}$$

The goal is to derive a set Q that maximizes $Cov(Q)$. That way, we would obtain K quasi-cliques, which individually capture important cross-entity aspects, and collectively represent the content in $\mathcal{S}_\mathcal{C}$. This has the effect of discouraging overlap, thus favoring diversity, among the quasi-cliques's in Q.

At first glance, this is superficially similar to the *maximum coverage problem* [7]. However, one crucial distinction is that the covering sets (in our case the q's) are not given as input. In fact, they have to be derived from $\mathcal{S}_\mathcal{C}$. In Section 4, we discuss how to derive efficiently a set Q that maximizes coverage.

In Fig. 1, we illustrate $K = 2$ quasi-cliques, one involving the blue vertices concerning "pork", and the other involving the red vertices concerning "pizza".

Micro-Review Synthesis. A quasi-clique $q \in Q$ may contain an arbitrarily large number of sentences, which are not appropriate for use as a summary. The third step deals with deriving a "summary tip" r_q for the set q. r_q is a subset of sentences from q that represents the content of q, while still being concise ($\leq$ 200

characters)[4]. The final output is $R_{\mathcal{C}} = \{r_q\}_{q \in Q}$. Fig. 1 illustrates that a summary tip is synthesized from each identified quasi-clique, resulting in two summary tips: "Get the pork belly sandwich. Try the pork buns!", and "Best artichoke pizza. Must try the CRAB pizza!".

4 Quasi-Clique Finding

The objective is to discover K quasi-cliques that collectively provide maximum coverage over the input set of sentences $\mathcal{S}_{\mathcal{C}}$ associated with an entity collection $\mathcal{C}$.

4.1 Problem

Given a multi-entity graph $G_{\mathcal{C}}$ (Section 3), we seek dense subgraphs within $G_{\mathcal{C}}$. The densest possible subgraph is a *clique*, i.e., a completely connected subgraph. This may be too strict a requirement, as it may exclude slightly weaker connectivity that still supports a meaningful aspect. For example, two sentences: s_1="*Huge wait for a table.*", and s_2="*Crazy long lines!*" both concern the waiting time, but do not share common words (potentially no edge). However, there might be an indirect connection through another sentence s_3 = "*Is the line worth the wait... Always.*". To include s_1, s_2, and s_3 in one dense subgraph, we need to relax the density constraint. As we expect that different aspects may correspond to subgraphs of varying sizes, we employ a relative notion of density. In particular, we adopt the definition of *quasi-clique*, with a specified minimum relative density of α, as follows.

Definition 1 (Quasi-Clique) For density threshold $\alpha \in [0, 1]$, a multi-entity graph $G = (\mathcal{V}, \mathcal{E})$ is a quasi-clique if G is connected and $|\mathcal{E}| \geq \alpha\binom{|\mathcal{V}|}{2}$.

The above definition has not taken into account the multi-entity perspective of our problem. Such quasi-cliques may be overly skewed towards a single partition V_i with only spurious connections to other partitions.

We seek a quasi-clique that not only has coherence within a partition (signifying an important aspect to an entity), but also can bring together multiple partitions (signifying an important aspect of common concern across entities). Therefore, instead of a single density threshold, we employ two density thresholds: *intra-density* α within each partition and *inter-density* β across partitions. These thresholds allow specifying the degree of connectivity for each entity, as well as across entities in the collection.

Since we seek quasi-cliques that connect entities over some common aspect, every pair of partitions (i.e., entities) should meet the inter-density constraint, instead of letting a very dense pair of partitions compensate for a much less dense pair of partitions (which may have lower relevance to the aspect being discussed).

Definition 2 (Multi-Entity Quasi-Clique) For density thresholds $\alpha \in [0, 1]$ and $\beta \in [0, 1]$, a multi-entity graph $G = (\mathcal{V}, \mathcal{E})$, with $n \geq 2$ partitions, is a multi-entity quasi-clique if G is connected and:

- for every partition $V_i \in \mathcal{V}$, we have $|E_i| \geq \alpha\binom{|V_i|}{2}$, and

[4] We discuss this in Section 5.

– for every pair of partitions $V_i, V_j \in \mathcal{V}$ (where $i < j$), we have $|E_{ij}| \geq \beta |V_i||V_j|$.

Based on this definition, different multi-entity quasi-cliques extracted from the same $G_{\mathcal{C}}$ may involve different numbers of entities (between 2 to $|\mathcal{C}|$).

This notion of multi-entity quasi-clique is also distinct from *multipartite* quasi-clique [9]. The latter only has cross-partition density requirements. In some cases, only sequential partitions, e.g., V_i and V_{i+1}, are required to be connected.

Because it is not desirable to select an aspect that is subsumed by another selected aspect, we would consider only *maximal* multi-entity quasi-cliques.

Definition 3 (Maximal Multi-Entity Quasi-Clique) Given a multi-entity graph $G = (\mathcal{V}, \mathcal{E})$, a subgraph $G' \subseteq G$ is a *maximal* multi-entity quasi-clique if G' is a multi-entity quasi-clique and there does not exist any subgraph $G'' \subseteq G$ such that G'' is a multi-entity quasi-clique and $G' \subset G''$.

As each maximal multi-entity quasi-clique models an aspect, we seek a number of such quasi-cliques that can comprehensively represent the content within $\mathcal{S}_{\mathcal{C}}$. Earlier, we define the notion of coverage in Section 3. If we view a maximal multi-entity quasi-clique q as a set that "covers" the vertices in it, it follows that the coverage $Cov(Q_{\mathcal{C}})$ of the set $Q_{\mathcal{C}}$ containing K quasi-cliques is as defined in Equation 1. We can view this as a variant of *maximum coverage problem*, whereby the objective is to select a number of sets that cover as many vertices as possible.

Problem 1 (Maximum Coverage via Multi-Entity Quasi-Cliques) Given a multi-entity graph $G_{\mathcal{C}}$, density thresholds $\alpha, \beta \in [0, 1]$, and an integer K, find $Q_{\mathcal{C}}$ or the set of K maximal multi-entity quasi-cliques in $G_{\mathcal{C}}$ to maximize $Cov(Q_{\mathcal{C}})$.

Finding the optimal solution to Problem 1 above is computationally intractable. This can be shown by reducing the *Maximum Clique* problem to our problem.

Lemma 1 *Maximum Coverage via Multi-Entity Quasi-Cliques is NP-hard.*

Proof (Sketch) For *Maximum Clique*, given a graph G, the objective is to find the clique with the largest number of vertices. We transform $G = (V, E)$ into a multi-entity graph $G' = (\mathcal{V}, \mathcal{E})$, by including the original vertices V as one partition of $\mathcal{V}$. In addition, as the multi-entity definition requires at least two partitions, we create a second partition V' with only one dummy vertex that is connected to all vertices in V. The optimal solution to the *Maximum Coverage via Multi-Entity Quasi-Cliques* problem on G' (with $K = 1, \alpha = 1, \beta = 0$) will also be the optimal solution to the *Maximum Clique* problem on G after extracting out the dummy vertex from the resulting clique. Since the latter is known to be NP-hard [5,15], it follows that the former (the more general case) is also NP-hard.

4.2 Approach

Due to the computational intractability, it is more productive to seek a heuristic solution. We are inspired by the maximum coverage problem [7], with a well-accepted greedy algorithm, which incrementally adds the next set that brings in the largest number of newly covered elements into the solution. One crucial distinction is that, in our case, the sets are not given in advance. In fact, they are

the maximal multi-entity quasi-cliques to be discovered from the graph. Instead of enumerating all maximal quasi-cliques that may still require exponential time [2], we propose to incrementally find the next maximal multi-entity quasi-clique that adds as many newly covered vertices as possible into the solution.

Algorithm 1: *MMEQC*(G, α, β, K)

Algorithm `MMEQC`(G, α, β, K)
$Q = \emptyset$
for $k = 1, 2, \ldots, K$ **do**
  $q^* = findNextMMEQC(G, \alpha, \beta, Q)$
  $Q[k] := q^*$
end
return Q

Procedure `findNextMMEQC`(G, α, β, Q)
$q = \emptyset$
$q = Construct(G, \alpha, \beta, Q)$
$q = Local(G, q, \alpha, \beta, Q)$
return q

Procedure `Construct`(G, α, β, Q)
$q = \emptyset$
$\alpha^* = 1$
$\beta^* = 1$
$q^* = \{(x_1, x_2)\}$: initializing with an *inter-edge* $(x_1, x_2) \in G$ ($x_1 \notin Q$ and/or $x_2 \notin Q$), with the highest number of common uncovered neighbors of x_1 and x_2
while *TRUE* **do**
  $q := q^*$
  if $\mathcal{N}_{\alpha^*\beta^*}(q) \neq \emptyset$ **then**
    Select $x \in \mathcal{N}_{\alpha^*\beta^*}(q)$
  end
  else if $\mathcal{N}_{\alpha\beta}(q) \neq \emptyset$ **then**
    Select $x \in \mathcal{N}_{\alpha\beta}(q)$
  end
  else
    break
  end
  $q^* := q \cup \{x\}$
  Set α^* to be the lowest *intra density* in q^*
  Set β^* to be the lowest *inter density* in q^*
end
return q

Procedure `Local`(G, q, α, β, Q)
$C = \{t \mid t \in q \wedge t \in Q\}$: set of *covered* tip sentences in q
$U = \{u \mid u \notin q \wedge u \notin Q \wedge \exists v \in q : (u, v) \in G\}$: set of *uncovered* tip sentences in q
for *each* $w \in C$ **do**
  $q' := q \setminus \{w\}$
  for *each* $v \in U$ **do**
    if $q' \cup \{v\}$ *satisfies* α, β *requirements* **then**
      $q' := q' \cup \{v\}$
    end
  end
  if $q'.size() \geq q.size()$ **then**
    $q := q'$
    ensure that q *is maximal*
  end
end
return q

Algorithm 1 illustrates this approach, which we call *MMEQC*, the acronym of Maximal Multi-Entity Quasi-Clique. It takes as input a multi-entity graph G, the density thresholds α and β, the number of quasi-cliques to be discovered K. The output set of cliques Q is initially empty. The loop runs K times, each time identifying the next maximal multi-entity quasi-clique to be included in Q by calling *findNextMMEQC*. We say that a vertex in G is *covered* if it is found in Q.

findNextMMEQC finds the next maximal multi-entity quasi-clique. One related work inspiring our approach is [1], which applies the framework of *Greedy Randomized Adaptive Search Procedure (GRASP)* to the specific problem of finding the largest maximal quasi-clique in a graph. This framework employs a two-step approach: *Construct* that constructs an initial solution using a greedy approach, followed by *Local* that conducts local search to find a better solution. However, our requirement is different from [1]. We are not finding the largest maximal quasi-clique, but rather the objective is to find as many new vertices to cover as possible. We also have to deal with the dual constraints α, β due to our multi-entity scenario. These differences require several innovations affecting *Construct* and *Local*, which we will elaborate shortly.

Construct. We now elaborate on *Construct*, whose pseudocode is shown in as a procedure in Algorithm 1. First, we initialize q with an inter-edge containing at least one vertex that is not yet covered. The basic idea is to grow q with as high intra-density and inter-density as possible (initially $\alpha^* = 1$ and $\beta^* = 1$) by selecting a vertex x from $\mathcal{N}_{\alpha^*\beta^*}(q)$, the set of vertices adjacent to q whose addition would still satisfy the intra-density α^* and the inter-density β^*. Otherwise, we select x from $\mathcal{N}_{\alpha\beta}(q)$ that only seeks to satisfy the lower minimum thresholds α and β, after which we then update the current α^* and β^* to the actual densities in q. If there is no such vertex, q is already a maximal multi-entity quasi-clique meeting the specified α and β density thresholds, and the algorithm returns q.

One key component of this procedure is the selection of the next vertex x in line 8 or line 11. Because the objective is to be able to grow q to bring in as many newly covered vertices into the solution as possible, the intuition is to pick an x such that its addition to q would still allow q to keep growing. We associate q with a quantity called *potential*, which measures how much denser q is than the minimum required density. Because there are two types of density, we define *intra-potential* and *inter-potential* separately. The intra-potential for a set q with n entities is computed as Equation 2, where $V_i(q)$ is the subset of vertices within q that belong to partition V_i, and $E_i(q)$ is the set of intra-edges among vertices in $V_i(q)$. In turn, the inter-potential is computed as Equation 3, where $E_{ij}(q)$ is the set of inter-edges between vertices in $V_i(q)$ and $V_j(q)$. The overall potential is defined as the sum of the two types of potential, as shown in Equation 4. q has higher potential if its current density is still sufficiently high enough to allow adding more vertices without lowering its density below the α and β requirements.

$$\phi_{intra}(q) = \sum_{i=1}^{n} \left(|E_i(q)| - \alpha \binom{|V_i(q)|}{2} \right) \tag{2}$$

$$\phi_{inter}(q) = \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \left(|E_{ij}(q)| - \beta |V_i(q)||V_j(q)| \right) \tag{3}$$

$$\phi(q) = \phi_{intra}(q) + \phi_{inter}(q) \tag{4}$$

Because potential indicates future opportunities for growth, we would select a vertex x from the set of candidates $\mathcal{N}_{\alpha\beta}(q)$ (or $\mathcal{N}_{\alpha^*\beta^*}(q)$), whose addition into q will maximize the gain (or minimize the drop) in potential, especially with respect to the vertices that are not yet covered. Therefore, we select x that maximizes the potential difference $\Delta_{q,x}$ as shown in Equation 5.

$$\Delta_{q,x} = \sum_{y \in \mathcal{N}_{\alpha\beta}(q) \setminus \{x\},\ y \text{ is uncovered}} \phi(q \cup \{x, y\}) - \phi(q \cup \{y\}) \tag{5}$$

Local Search. After *Construct* produces an initial solution, we conduct a local search, to attempt swapping each covered vertex that is already included in the previous solution Q with one or more uncovered vertices. Because of the changes in the vertices, we need to ensure that the resulting q would still be a maximal multi-entity quasi-clique by growing it to its maximal again.

5 Micro-Review Synthesis

We expect the set $Q_{\mathcal{C}}$ of K maximal multi-entity quasi-cliques obtained through the process described in Section 4 to capture K salient aspects of a collection of entities $\mathcal{C}$. However, for presentation purpose, a quasi-clique is not appropriate due to the potentially large number of sentences (vertices). Therefore, we seek a representation for human consumption in the form of one "summary tip" (under 200 characters) r_q for each quasi-clique $q \in Q_{\mathcal{C}}$ discovered. The output summary is thus a collection of K summary tips $R_{\mathcal{C}} = \{r_q\}_{q \in Q_{\mathcal{C}}}$.

We observe that each quasi-clique q tends to contain sentences that pertain to a coherent aspect, e.g., pork dishes. The respective sentences then may capture still more specific information, such as different pork dishes, e.g., pork buns, pork belly, pulled pork. To capture these pertinent sub-aspects, we seek to select a few representative vertices from each q to make up r_q. To do so, we really need to address two issues: how many sentences should be selected, and which ones. The more sentences we select, the more detailed and representative a summary tip becomes. Meanwhile, it also becomes longer and requires higher cognitive load to read. We therefore need to manage the trade-off between the ability of a summary tip to represent the vertices in q and its length.

We find an appropriate formulation in terms of *Facility Location Problem* (FLP) [8]. Given a set of facilities and a set of customers, FLP seeks to find a subset of the facilities to open in order to serve all the customers with the lowest cost possible. There is a cost associated with opening a facility, as well as a cost for serving a customer from the closest facility. Therefore, FLP seeks to find a balance between the extremes of opening too many facilities (high opening costs) versus opening too few facilities (high service costs). In our context, customers are sentences in q, whereas facilities are candidate sentences for selection into r_q. Therefore, we seek a specific formulation of FLP in our context to balance opening costs (the overall length r_q) versus service costs (the ability of r_q to represent q).

We model the opening cost of a facility, i.e., a candidate sentence, to be a function of the length of the sentence. This is shown in Equation 6, where $\mathit{length}(s_i)$

is the length of a sentence s_i in characters [5], and λ is a coefficient to regulate the effect of opening cost. In practice, we find that $\lambda = 0.1$ works well in experiments.

$$open(s_i) = \lambda \cdot length(s_i) \tag{6}$$

In turn, we model the service cost between a facility s_i and a potential consumer s_j as a function of the cosine similarity between their corresponding sentences, as shown in Equation 7. The greater the similarity, the lower is the service cost.

$$service(s_i, s_j) = 1 - cosine(s_i, s_j) \tag{7}$$

Problem 2 (Micro-Review Synthesis) Given a maximal multi-entity quasi-clique q and a length limit of γ, select a subset of vertices in q to be the set of facilities to open r_q, so as to minimize the total cost:

$$\sum_{s_i \in r_q} open(s_i) + \sum_{s_j \in q} (\min_{s_i \in r_q} service(s_i, s_j))$$

subject to the constraint $\sum_{s_i \in r_q} length(s_i) \leq \gamma$.

Approach. FLP is known to be NP-hard [8]. There are different approaches to solve the problem [33]. For non-metric distances, there exists a known approximation algorithm obtained by casting FLP into *Minimum Weight Set Cover* (MWSC) as follows. Consider a tip sentence s_i as a facility, and let T_{s_i} be the set of tip sentences (i.e., customers) that are served by s_i (we regard $s_i \in T_{s_i}$ as it could serve itself). We define a set corresponding to s_i that "covers" the elements in T_{s_i}, with weight $open(s_i) + \sum_{s_j \in T_{s_i}} services(s_i, s_j)$. The solution to MWSC, i.e., the sub-collection of sets that cover all the elements with the minimum total of weight, also provides the solution for the corresponding instance of FLP.

The greedy algorithm for set cover has a performance guarantee by a factor of approximately $\ln n$. It may seem at first glance that we need to select from the enumeration of all the possible sets from the tips in q, which is intractable. However, [13] shows that, for each s_i, it is sufficient to consider the pairs $(s_i, T_{s_i}^k)$, for $k = 1, ..., |q|$, where $T_{s_i}^k$ denotes the first k tip sentences sorted by the serving cost. The process stops when all the tip sentences are covered. We then have a set of selected facilities r_q, and their corresponding partitions of customers.

Empirically, we observe that many times the greedy outcome naturally falls within the length limit γ. Otherwise, we process the greedy outcome further by omitting the facility that would result in the smallest increase of cost until the length falls within γ. This is more advantageous than prematurely terminating the greedy when γ is breached, as there may be another better facility to be omitted among those selected by greedy than the one that would have been avoided by such a premature termination.

[5] We are working with micro-reviews. A summary tip mimics a micro-review. By definition, micro-reviews are limited by character count. For instance, Foursquare defines the limit to be 200 characters. Therefore, we measure sentence length in terms of characters.

Table 2 Foursquare dataset for 102 restaurants

	Min	Max	Mean	StdDev	Total
#Tips	51	479	143.0	74.2	14583
#Tip Sentences	79	807	243.8	127.6	24872

Table 3 Datasets Based on Foursquare Entities

	#Collections	#Entities in a Collection			
		Min	Max	Mean	StdDev
ZIP Codes	16	2	21	5.9	4.7
Grids	56	2	12	5.8	2.9
Categories	39	2	22	4.7	3.6

6 Experiments

We describe experiments that evaluate the quality of summaries. A good summary $R_{\mathcal{C}}$ for a multi-entity collection $\mathcal{C}$ should be *representative* of the underlying content, *diverse* in capturing content relevant across entities, and *coherent*.

6.1 Experimental Setup

Dataset. We use the Foursquare data collected by [25]. This data contains the set of tips of 109 restaurants in New York as of March 2012. We retain 102 restaurants that have at least 50 tips for our experiments, so as to ensure that every restaurant has sufficient content for summarization. Table 2 shows the statistics of the number of tips and sentences, in terms of min, max, mean, standard deviation across restaurants, as well as the sum total. On average, each restaurant has 143 tips. As a tip may have multiple sentences, the average restaurant has 243.8 sentences. The respective maxima are 479 tips and 807 sentences.

In this work, we are dealing not with individual entities, but rather with collections of entities. There are various realistic scenarios whereby users may be interested to know more about a group of restaurants. For instance, when they are in a particular locality, they may want to know about restaurants in the neighbourhood. Alternatively, they may want to know about restaurants serving a particular type of food. We base on these scenarios to create three datasets, whereby each dataset consists of a number of different collections of restaurants.

1. *ZIP Codes*: The first dataset treats each ZIP code as a locality or neighborhood of interest. As shown in Table 3, there are 16 ZIP codes with at least two entities in the dataset. There are between 2 to 21 entities in each collection, with an average of 5.9 entities per ZIP code-based neighborhood.
2. *Grids*: As *ZIP Codes* may correspond to neighborhoods of varying sizes (the largest has 21 entities), we also consider another dataset based on uniform-sized localities that tend to cover smaller areas. To derive these neighborhoods, we first divide the area bounded by the restaurants into 60 x 20 equal-sized grids of approximately 0.25 mile by 0.25 mile each. Then, we use sliding windows of 2 x 2 grids to constitute the neighborhoods. Thus each neighborhood is approximately 0.5 mile by 0.5 mile, and any two adjacent neighborhoods overlap in half their areas. We retain only neighborhoods with at least two entities,

and ensure that there are no duplicate neighborhoods (exactly the same set of entities). Table 3 shows that there are 56 such neighborhoods with between 2 to 12 entities, with an average of 5.8 entities per grid-based neighborhood.
3. *Categories*: Each restaurant in the corpora is also assigned to one or more categories. These categories may correspond to cuisine type (e.g., Japanese, Cuban, Thai), or the type of venue (e.g., bar, cafe). There are 39 categories with between 2 to 22 entities, with an average of 4.7 entities per category.

Comparative Methods. We evaluate the following comparative methods in terms of multi-entity summarization. The first three are *graph-based* approaches. They produce K dense subgraphs, which are then transformed into summaries by the micro-review synthesis process described in Section 5. This comparison could help us understand the efficacies of several varying definitions of dense subgraphs. The next three methods are *text summarization* techniques that work directly with tips, and do not depend on finding dense subgraphs. This comparison helps us to understand the effectiveness of basing our summary on dense subgraphs.

1. *MMEQC*: Our proposed method *MMEQC* stands for Maximal Multi-Entity Quasi-Clique (see Definition 3). It is a composite of finding quasi-cliques described in Section 4, followed by micro-review synthesis described in Section 5. Our key distinction is modelling quasi-cliques with multi-entity constraints via the intra-density and inter-density requirements.
2. *MQC*: We consider a simpler version of our method, by removing the multi-entity constraints, and base the approach on regular Maximal Quasi Cliques (MQC) (see Definition 1) with a definition of density that is agnostic to entities. This is to test the effects due to multi-entity constraints.
3. *RAMC*: Related to finding dense subgraph, an existing work Redundancy-Aware Maximal Cliques (RAMC) [42] tries to produce a concise and complete summary of a set of maximal cliques. The output of RAMC is dependent on the *visibility* parameter, τ, which reflects the percentage of coverage that each maximal clique will be covered by the selected summary. We use the author implementation[6] to find the top K maximal cliques, following the procedure described in [42] with deterministic setting and global filtering.
4. *MEAD*: We run extractive summarization MEAD [32] as follows. The tips from the collection make up a 'document'. Because the ordering of tips in a document is not meaningful, the *position* feature is turned off. MEAD selects K tips from the document as summary. We use the author implementation[7].
5. *TextRank*: Another extractive method, but based on ranking, is TextRank [24]. The objects to be ranked are the input tips. TextRank forms a graph with tips as vertices connected by similarity-based edges, then runs a random walk algorithm to find the K vertices with the highest stationary probabilities.
6. *Opinosis*: As a representative of abstractive summarization, we compare to Opinosis [12], using the author implementation[8] with standard settings. Opinosis first forms a graph with words as vertices from the content in a collection, then selects the K highest-scoring paths (sentences) as summary.

[6] `https://github.com/cntswj/clique-summary`
[7] http://www.summarization.com/mead/
[8] http://kavita-ganesan.com/opinosis-summarizer-library

Graph Construction. As described in Section 4, for an entity collection $\mathcal{C}$, we construct a graph $G_{\mathcal{C}}$ that has sentences as vertices. For an edge to exist, there should be sufficient similarity between two sentences. We therefore impose a minimum similarity threshold, which is determined as follows. As ground truth, we randomly pick 2000 pairs of sentences from our corpora, and manually assign each pair with a binary label (match vs. non-match). To find the optimal similarity threshold to approximate this ground truth, we "classify" each pair with similarity equal or greater than the threshold as a match, and non-match otherwise. We compare these with the ground-truth labels, and compute *recall*, *precision*, and their harmonic mean *F-measure* for different thresholds. Fig. 2 tracks these metrics. As expected, as the threshold increases, recall decreases while precision increases. We do not show the trends beyond cosine threshold 0.1 because essentially the same trends continue. F-measure finds an optimal balance at 0.02. Subsequently, we use this threshold to build $G_{\mathcal{C}}$ for each collection $\mathcal{C}$. For any comparative method that involves similarity measure, we use *cosine similarity* consistently.

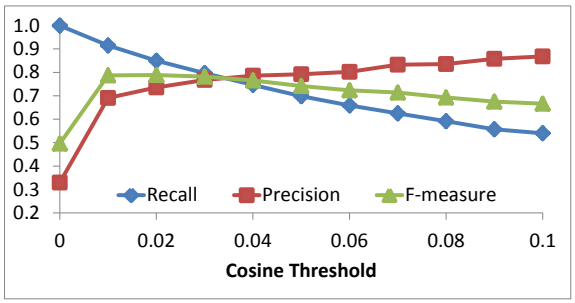

Fig. 2 Graph Construction: Cosine Similarity Threshold

6.2 Evaluating Representativeness, Diversity, and Coherence

We now compare the summary $R_{\mathcal{C}}$ generated for a collection of entities $\mathcal{C}$ by *MMEQC* to those generated by the baselines listed in Section 6.1.

Metrics. As motivated previously, we would like the output summaries to represent as much information as possible from the collection. Because we are addressing multi-entity summarization, we would like the summary to contain tips that are as diverse as possible, in terms of capturing content relevant across the entities being represented. We would also like each summary tip to be as coherent as possible. Towards these objectives, we introduce three metrics, as well as an overall aggregate measure, as follows.

– The first metric is *representativeness*, i.e., how well a summary $R_{\mathcal{C}}$ can capture the content of the input corpora of sentences $\mathcal{S}_{\mathcal{C}}$. Earlier we saw that cosine similarity of 0.02 is good at signifying shared meaning. Therefore, we say that a

summary tip $r \in R_{\mathcal{C}}$ "represents" a sentence $s \in \mathcal{S}_{\mathcal{C}}$, if the similarity between r and s is above this threshold. In other words, each summary tip r is associated with a subset $S_r \subseteq \mathcal{S}_{\mathcal{C}}$. We refer to S_r as the set of sentences that are represented by r. The representativeness of a summary $R_{\mathcal{C}}$ is thus defined as the fraction $\frac{|\bigcup_{r \in R_{\mathcal{C}}} S_r|}{|\mathcal{S}_{\mathcal{C}}|}$. The higher the representativeness, the more information is captured from $\mathcal{S}_{\mathcal{C}}$.

- The second metric is *diversity*. For each $r \in R_c$, we would like S_r to be diverse with respect to the entities. To this end, we use the normalized entropy $H(S_r) = -\sum_{i=1}^{n} \frac{p_i \log p_i}{\log n}$, where p_i is the fraction of tip sentences in S_r belonging to entity e_i, and n is the number of entities in the collection $\mathcal{C}$. The higher the entropy $H(S_r)$, the more even the distribution of representation among entities within S_r. The diversity of a summary $R_{\mathcal{C}}$ is thus defined as the average normalized entropy across the summary tips, i.e., $\frac{1}{K} \sum_{r \in R_{\mathcal{C}}} H(S_r)$.
- The third metric is *coherence*. We measure coherence in terms of whether the summary tip r itself contains related sentences, as well as whether the represented set of sentences S_r contains related sentences. We therefore take the average of the density within r, and the density within S_r. For the former, we measure the density of the subgraph in $G_{\mathcal{C}}$ induced by the vertices in r. Note that it is possible for a summary tip r to contain sentences from a single entity, as long as they are representative of S_r. For the latter, as S_r necessarily contains sentences from multiple entities, we measure its density as the mean of the intra-link density and inter-link density of the subgraph induced by S_r. For a summary $R_{\mathcal{C}}$, we take the average coherence of its constituent summary tips.
- The three factors of *representativeness*, *diversity*, and *coherence* are all in the range of 0 to 1. Because we consider all the three factors equally important, we look for a way to aggregate these factors in a balanced way. Average is one common measure that considers their contributions equally. Therefore, for an aggregate measure, we compute an *Overall* measure as the average of the representativeness, diversity, and coherence. This *Overall* measure offers a summative view of performance.

Some of the graph-based methods require some parameters to be tuned. We conduct a grid-search of parameter settings and pick the best one for each method that maximizes its *Overall* score. For *RAMC*, following [42], we vary the visibility parameter τ from 0.5 to 1.0, and discover that the best setting is 0.7 for *ZIP Codes*, 0.8 for *Grids*, and 0.5 for *Categories*. For *MQC*, we vary α from 0.5 to 1.0, and discover the best setting to be 0.9 on all datasets. For our method *MMEQC*, we consider different pairs of parameter values (α, β), for $\alpha \in [0.5, 1]$ and $\beta \in [0.5, 1]$, and arrive at the following settings: (0.9, 0.9) for *ZIP Codes* and *Grids*, and (1, 0.9) for *Categories*. We use these parameter settings in the following discussion.

Results. Table 4 shows the representativeness, diversity, coherence, and overall scores of various methods on the *ZIP Codes* dataset for $K = 10$. We will vary K shortly. As shown in Table 4, our method *MMEQC* and the simplified *MQC* have the highest representativeness. The simplified *MQC* is slightly better, because it does not face the inter-density constraint. On the other hand, in terms of diversity, our method *MMEQC* has the highest diversity among all methods at 0.57 for *ZIP Codes*, whereas *MQC* now has the lowest, implying that *MQC* tends to discover summary tips that are overly skewed towards a single entity. In terms of coherence,

Table 4 *ZIP Codes*: Representativeness, Diversity, Coherence and Overall scores for $K = 10$

	Representativeness	Diversity	Coherence	*Overall*
MMEQC	0.45	0.57	0.82	***0.61***
MQC	0.48	0.37	0.79	*0.55*
RAMC	0.44	0.40	0.83	*0.56*
MEAD	0.43	0.47	0.40	*0.43*
TextRank	0.42	0.42	0.35	*0.40*
Opinosis	0.34	0.46	0.80	*0.53*

Table 5 *Grids*: Representativeness, Diversity, Coherence and Overall scores for $K = 10$

	Representativeness	Diversity	Coherence	*Overall*
MMEQC	0.46	0.53	0.80	***0.60***
MQC	0.48	0.40	0.78	*0.55*
RAMC	0.44	0.38	0.82	*0.55*
MEAD	0.44	0.48	0.38	*0.43*
TextRank	0.43	0.42	0.37	*0.41*
Opinosis	0.34	0.46	0.79	*0.53*

Table 6 *Categories*: Representativeness, Diversity, Coherence and Overall scores for $K = 10$

	Representativeness	Diversity	Coherence	*Overall*
MMEQC	0.51	0.67	0.85	***0.68***
MQC	0.53	0.58	0.82	*0.64*
RAMC	0.48	0.56	0.86	*0.63*
MEAD	0.49	0.63	0.42	*0.52*
TextRank	0.50	0.62	0.43	*0.52*
Opinosis	0.39	0.59	0.83	*0.60*

RAMC is the highest, as it is based on completely connected cliques. However, *MMEQC*'s coherence is very near to *RAMC*'s. Considering the *Overall* scores (italicized), *MMEQC* outperforms the other methods. Based on paired samples t-test, *MMEQC*'s outperformance over the other methods is statistically significant at 0.05 level. This supports the motivation of factoring multi-entity constraints into *MMEQC*, which finds summary tips of greater diversity.

The results for *Grids* and *Categories* are listed in Table 5, and Table 6 respectively. The trends are essentially the same: the results and the observations echo those for *ZIP Codes*.

In Fig. 3, we plot the *Overall* score for different values of $K \in [1, 10]$. We observe that *MMEQC* consistently outperforms the other baselines (except for $K = 1$ in *Grids*), and the differences become more pronounced for larger K. For very small K, *MMEQC* and the baselines tend to discover similar aspects, which might be the most dominant aspects. However, for larger values of K, *MMEQC* continually discovers new aspects, while the baselines may still cover aspects redundantly. Thus, the differences between *MMEQC* and the baselines are statistically significant (at 0.05 level) for $K \geq 4$ for *ZIP Codes*, $K \geq 3$ for *Grids*, and $K \geq 2$ for *Categories*.

6.3 Evaluation based on Relevance Ranking

In the previous section, evaluation is based on a binary notion of relevance, i.e., a summary tip either represents or does not represent an input tip. However, there

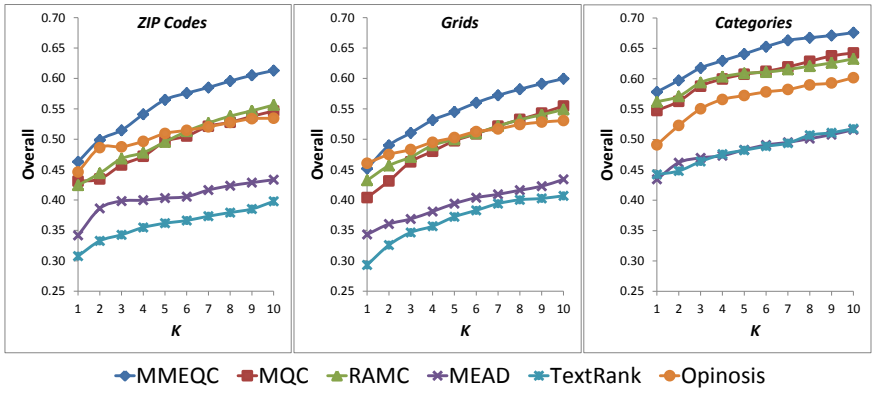

Fig. 3 The Overall Scores: Vary the Number of Summary Tips K

may be different degrees of relevance. In this section, we conduct a second set of evaluations based on relevance ranking. Specifically, we model it as an information retrieval task. A query is a salient keyword that a user may be interested in with respect to the collection of entities. For each method, we treat each summary tip as a 'document'. Given a query, we rank all the documents (individual summary tips) from all the comparative methods. A better method is expected to produce a summary containing tips that rank highly across various queries.

First, we discuss what make good queries. One way is to look into the salient words in the corpora. One popular notion for a word's salience within a document is *tf-idf* [23]. *idf* customarily refers to inverse document frequency. Because we would like to arrive at words salient to each entity, we adapt it to our scenario by computing *tf-irf*, where *irf* refers to inverse restaurant frequency, i.e., the inverse of the number of restaurants that contain a word. *tf* in this case is the normalized term frequency of a word in an entity's tips. Thus, words with high *tf-irf* are important to an entity. Because we are looking for words that are salient to a *collection* of entities c, we average the *tf-irf* values of words across entities in c. We then keep the top M words, occurring in two entities or more, with the highest averaged *tf-irf* values as the M queries. Such keywords are reflective of what are considered most important among the entities to be summarized.

We now discuss how to measure the performance of a method. For each query, we rank the summary tips from the six comparative methods by relevance (cosine similarity). The methods are then ranked from 1 (highest) to 6 (lowest) according to their respective best-performing summary tip. As the evaluation metric, we compute the *percentile rank* of a method as the fraction of methods with the same relevance score or lower as the method of interest. The highest percentile rank possible is 1. We average the percentile rank of a method across the M queries.

Fig. 4 shows the percentile ranks of the comparative methods with $K = 10$, for varying number of queries $M \in [10, 100]$. Evidently, the proposed method *MMEQC* tends to have the highest percentile ranks. The differences between *MMEQC* and others are statistically significant (0.05 level) in all cases. This is followed by our simplified model *MQC*, and then the baseline *RAMC*. Interestingly, these three

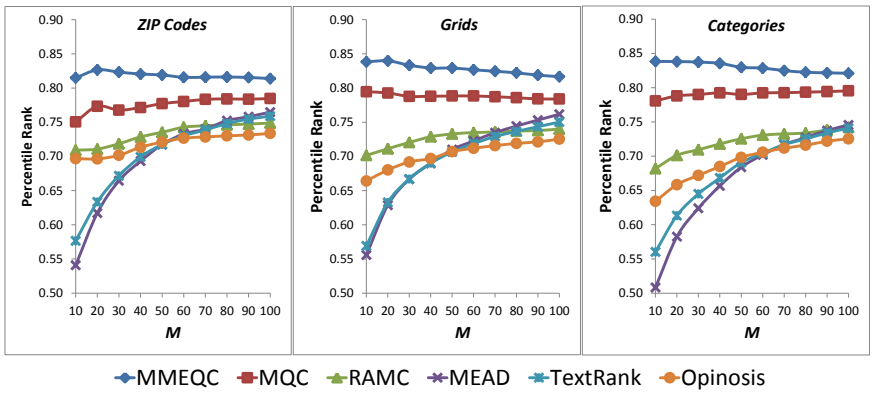

Fig. 4 Relevance Ranking with $K = 10$: Vary the Number of Queries M

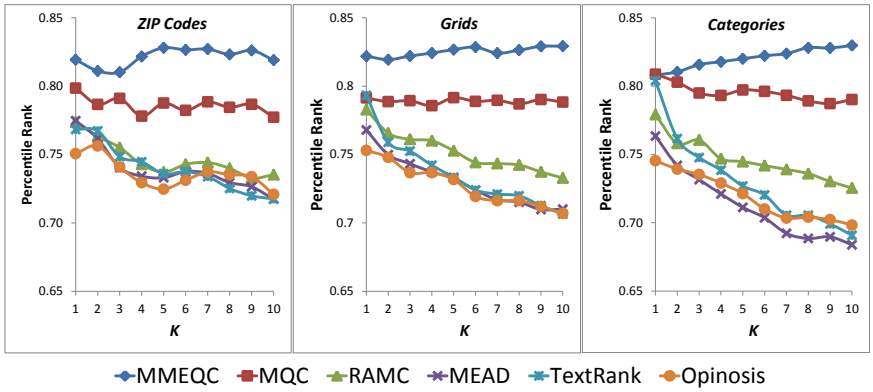

Fig. 5 Relevance Ranking with $M = 50$: Vary the Number of Summary Tips K

approaches based on dense subgraphs outperform the other baselines based on text summarization, such as *MEAD*, *TextRank*, and *Opinosis*.

This trend still stands, when we fix $M = 50$, and vary the number of summary tips K instead. Fig. 5 shows that *MMEQC* is still very competitive. The differences are more pronounced with increasing K, as the performance of various baselines begin to fall. The differences between *MMEQC* and others are statistically significant (0.05 level) in all cases for $K \geq 4$, and in a number of cases for $K < 4$. As we increase K, *MMEQC* still manages to generate summary tips that are relevant across entities. Whereas, for the other baselines, the later summary tips may be skewed towards a single entity, resulting in lower relevance overall.

6.4 Evaluating Readability

Readability is difficult to quantify through automatic means. Therefore, we rely on a simple user study, involving ten human judges who are not involved in this work. We show the judges the summaries produced by the six comparative methods for $K = 10$. Each human judge is asked to give a rating from 1 to 5, with 1 (lowest) signifying an unreadable piece of text and 5 (highest) signifying a highly readable, descriptive and easily understandable summary. Because of the heavy cognitive load of the user study, we conduct this only on *ZIP Codes*, for the nine collections of entities with at least five entities each. The judges are blind to which methods produce which summaries, and the ordering of the summaries is randomized.

Table 7 shows the average readability scores by the human judges as well as the standard deviations. The first four listed in Table 7, including our method *MMEQC*, have average readability scores above 3.5, which we consider to be highly readable. The differences between the top-four techniques are small, and as the standard deviation indicates they are within error. Looking at the standard deviations, we observe that the top ranking methods also have the highest standard deviation, indicating a variance of opinions between the judges for the quality of the methods. In contrast, the standard deviation for *MMEQC* is one of the lowest in the group, implying an agreement between the judges about the high quality of our method. The last two methods in Table 7 have scores below 3, noticeably lower than the previous four.

Delving into the results, we postulate that the differences in readability may be explained in part by the nature of the summarization approach itself, perhaps more so than the specific efficacy of the respective algorithms.

Two of the approaches, i.e., *TextRank* and *MEAD*, are *extractive* methods, which select from existing well-formed tips. As expected, their readability scores are high, because these tips have been put together by a single author.

MMEQC, *MQC*, and *RAMC* share the same micro-review synthesis phase. In this phase, a summary tip is assembled from sentences coming from different tips, which may have been written by different authors. There is some risk that this *constructive* approach may affect readability. Interestingly, the summaries from *MMEQC* and *MQC* are still highly readable, comparable to the extractive summaries above. *RAMC* may be lower because it is based on completely connected cliques, which may be too restrictive to yield well-rounded summaries.

The lowest readability score is for the fully *abstractive* summarization method *Opinosis*. This can be explained by the high level of difficulty of producing a natural language sentence, which is a disadvantage in terms of readability.

Table 7 User Study on Readability

		Readability (Mean ± StDev)
Extractive	TextRank	3.8 ± 0.6
	MEAD	3.6 ± 0.7
Constructive (Synthesis)	MMEQC	3.6 ± 0.4
	MQC	3.6 ± 0.4
	RAMC	2.9 ± 0.7
Abstractive	Opinosis	2.2 ± 0.5

We measure the agreement among the human judges by computing the Pearson's correlation coefficient between each pair of human judges in terms of their average readability scores for various methods. Understandably, there is some subjectivity that may result in variance among judges. However, in general, there is positive agreement among judges. In the range of [-1, 1] with -1 indicating total disagreement and 1 total agreement, the average correlation is 0.3.

A deeper look reveals that a majority group of 7 judges have higher agreement, with a higher average correlation of 0.5. The other 3 judges disagree with the majority that results in lower overall correlation. The minority judges tend to prefer shorter summaries, and as a result they penalize *TextRank* and *MEAD* which have relatively longer summaries. To some extent, this explains the higher standard deviations for *TextRank* and *MEAD* in Table 7.

7 Computational Efficiency

The main focus of this paper is on the effectiveness in discovering summaries that represent the entities within a given set, which has been discussed extensively in the previous section. In this section, we turn to a brief discussion on computational efficiency. In particular, we are interested in two questions. First, as our algorithm aims to construct a good summary by discovering interesting structures in the data, we benchmark our algorithm in terms of quality and efficiency against a black-box optimization approach that tries to directly optimize the quality of the summary, and we investigate the tradeoff between quality and efficiency. Second, we study how our algorithm performs, in terms of both effectiveness and efficiency, as we increase the size of the entity collections, and the number of sentences.

7.1 Benchmarking against a Black-Box Optimization Algorithm

In Section 6, we evaluated the quality of the summaries produced by our approach with the *Overall* measure, which is the mean of representativeness, diversity, and coherence. The intuition is that the structures discovered by our algorithm correspond to summaries with high overall score, and that the algorithm is able to discover them efficiently. Alternatively, we could consider an algorithm that directly optimizes the *Overall* measure, using unlimited amount of time. We now benchmark our algorithm against a black-box optimization algorithm that optimizes directly the *Overall* measure. We adopt the Simulated Annealing [17] optimization scheme, which is commonly used in practice, and we study the efficiency-effectiveness tradeoff.

Simulated Annealing. The desired output are K summary tips, where each tip consists of several sentences that collectively fall within 200 characters. The search space encompasses all such summary tips that could be formed by sentences within the corpus. Simulated annealing proceeds in iterations. It begins with K random summary tips, each initially containing a sentence. In each iteration, we create a neighboring solution by adding, swapping, or removing a sentence from one of the summary tips. At any point, we ensure that a summary tip always has at least one sentence, and has at most 200 characters. If the neighboring solution is

better, it replaces the current solution. Otherwise, it may still be accepted according to the following acceptance probability: $exp(-(E(R_{new}) - E(R))/T)$, where R and R_{new} are the current and the neighboring solutions respectively, and $E(\cdot)$ is the energy function, defined as the inverse of the *Overall* score of the summary. T is the current "temperature". Initially, T is high to allow greater exploration and to escape local optima. Over time, T reduces according to a cooling rate. Once the energy stops reducing, Simulated Annealing essentially has converged.

Benchmarking. We now compare the effectiveness (*Overall* measure) and the running time of *MMEQC* vs. Simulated Annealing (*SA*) for $K = 10$ summary tips on the three datasets used in Section 6, namely: *ZIP Codes*, *Grids*, and *Categories*. Table 8 summarizes the *Overall* scores for all three datasets. First, we discuss the performance of SA when given the same amount of time that our method takes to complete. We refer to this as *SA-EqualTime*. Evidently, *SA-EqualTime* performs significantly lower than *MMEQC* across the datasets, probably because it has not yet converged. The version of *SA* that is run to convergence, *SA-Converged*, tends to improve in terms of representativeness and diversity, but lags in coherence, resulting in an *Overall* measure that is higher than *MMEQC*'s. However, as we will soon see, this comes at a cost in running time.

We now discuss the running times, which are listed in Table 9. First, for *Zip Codes*, we look at the average running time across the entity collections in the dataset. *MMEQC* completes in about 45s (under a minute), while *SA-Converged* takes about 3504s (about an hour). We also show the median, minimum and maximum running times for *SA-Converged*, as well as the corresponding times for *MMEQC*. These statistics represent approximately two orders of magnitude increase in running time required by *SA-Converged*. Similar results can be seen for the other two datasets as well. If we consider the longest times for each dataset, *SA-Converged* takes between 4 to 9 *hours*, whereas *MMEQC* requires only 3 to 10 *minutes* for the same cases.

This benchmarking suggests that *MMEQC* is relatively efficient in realizing the gain in effectiveness within a much shorter running time than *SA-Converged*. When given unfettered running time, *SA-Converged* could achieve a higher *Overall* measure, but it is infeasible in practice due to the two order-of-magnitude increase in running time.

For reference, to see if *MMEQC* would outperform a simpler, and more computationally efficient baseline, we include a comparison to an algorithm based on "two-level" *KMeans*. First, at the level of entities, we divide the entities into clusters based on their bag-of-words representations. For parity, we generate 10 clusters using the *KMeans* algorithm to compare with *MMEQC* with 10 summary tips. Second, at the level of sentences, in order to generate a summary tip for each cluster, we will further divide the tip sentences in a cluster of entities into K groups of sentences, again using the *KMeans* algorithm. Because each summary tip is restricted to 200 characters, we set the number of sentence groups to $K = 200/L_s$, where L_s is the average length in character of the sentences in the cluster. From each group of sentences, we select the medoid, and combine these medoids into a summary tip. We repeatedly pick the medoid in the order of descending sizes of sentence groups, while still meeting the length restriction. The resulting 10 summary tips, one for each cluster of entities, make up the summary.

As shown in Table 8, this approach, referred to as *KMeans*, has the worst *Overall* performance among the algorithms. Because one or more than one entity

Table 8 Comparison with Simulated Annealing: *Overall* Scores

	ZIP Codes	Grids	Categories
MMEQC	0.61	0.60	0.68
SA-EqualTime	0.46	0.48	0.55
SA-Converged	0.75	0.75	0.80
KMeans	0.34	0.35	0.42

Table 9 Comparison with Simulated Annealing: Running Time in Seconds

		Average	Median	Min	Max
Zip Codes	MMEQC	44.6	4.0	0.03	346.8
	SA-Converged	3504.4	1269.3	42.73	18379.5
	Kmeans	1.5	0.5	0.08	15.0
Grids	MMEQC	33.9	4.9	0.05	345.2
	SA-Converged	2999.0	1640.3	56.56	14825.7
	Kmeans	0.8	0.6	0.11	2.7
Categories	MMEQC	44.5	3.6	0.12	625.0
	SA-Converged	3927.5	1370.7	51.49	33274.8
	Kmeans	0.9	0.4	0.07	13.6

can only be summarized using one summary tip limited to 200 characters, the summary could not cover many aspects, resulting in low *Coverage*. For the same reason, a summary tip tends to cover diverse aspects of that entity, resulting in low *Coherence*. While in some instances *KMeans* may have reasonable *Diversity*, in aggregate the *Overall* scores are significantly lower than *MMEQC*. Thus the gain in efficiency achieved by *KMeans* in Table 9 is at the expense of much lower *Overall* scores.

7.2 Scalability

We now explore how the proposed method behaves with respect to different sizes of entity sets. In order to consider sets of increasing sizes in a natural way, we build on the concept of grids. Previously, the *Grids* dataset comprises of small grids of equal sizes. In this experiment, we begin with a 6×6 grid in the center of the map. As shown in Table 10, this corresponds to a set of 2 entities involving 466 sentences (graph vertices). We then systematically enlarge the grid, each time expanding by one unit in every direction, eventually reaching a 20×20 grid corresponding to an area of 5 miles by 5 miles, encompassing 62 entities and 11,885 sentences. Each subsequent grid is a superset of the preceding grid, and the sizes naturally increase. We use the same $\alpha = 0.9$, $\beta = 0.9$ settings as the original *Grids* dataset.

First, we discuss the effectiveness of our algorithm as a function of the input size. Table 10 shows that for larger grid sizes, representativeness tends to decrease, which is expected as the same number of summary tips ($K = 10$) would need to cover increasingly larger sets of entities. Meanwhile, diversity increases as there are more entities involved. Coherence generally stays the same, with a slight decrease to accommodate more varied entities. The *Overall* score is relatively stable across grid sizes, suggesting that the algorithms are still effective for larger problem sizes.

We now look at efficiency. Fig. 6 shows how the running time of *MMEQC* varies with the number of the input sentences. The number of sentences, which translates to the number of vertices in the input graph, is a more reflective measure

Table 10 Increasing Grid Sizes: Effectiveness

GridSize	#Sentences	#Entities	Representativeness	Diversity	Coherence	*Overall*
6 × 6	466	2	0.64	0.58	0.83	*0.68*
8 × 8	710	3	0.57	0.36	0.85	*0.60*
10 × 10	854	4	0.50	0.41	0.81	*0.57*
12 × 12	2435	10	0.49	0.42	0.82	*0.58*
14 × 14	4667	22	0.43	0.66	0.72	*0.60*
16 × 16	8424	41	0.40	0.68	0.77	*0.62*
18 × 18	10567	54	0.43	0.72	0.77	*0.64*
20 × 20	11885	62	0.41	0.72	0.78	*0.64*

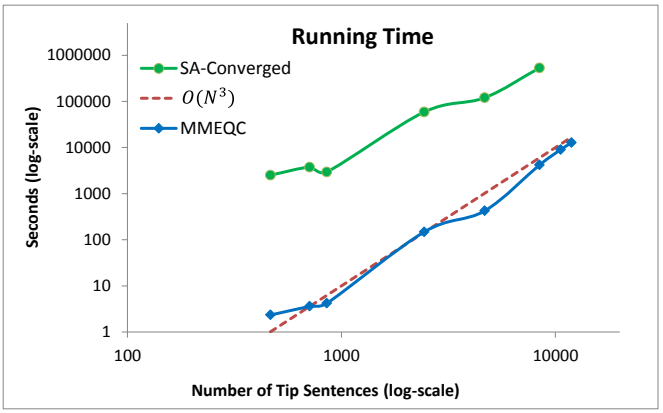

Fig. 6 Increasing Grid Sizes: Running Times in seconds

of the problem size than the number of entities (some entities have few, while other entities have many sentences). For the largest set involving 11K sentences, the running time is under four hours. For the typical inputs we considered in this paper, the average running time is well under a minute (see also Table 9). Moreover, as shown in Fig. 6 there is an approximately linear trend in the log-log scale, which suggests that our algorithm scales polynomially with respect to the size of the input N. For reference, we also include $\mathcal{O}(N^3)$ trend line in the figure. We also include the running time of *SA-Converged*[9] on this dataset. Fig. 6 shows that *MMEQC* is consistently much faster than *SA-Converged* across different input sizes.

8 Conclusion

In this work, we develop an approach for multi-entity summarization, in the context of synthesizing micro-reviews for a collection of entities from the content associated with the underlying entities. We show that obtaining a summary for multiple entities requires careful identification of aspects, modeled as maximal

[9] For *SA-Converged*, we include data points that completed within 7 days.

multi-entity quasi-cliques, drawing common threads across the entities. Experiments on Foursquare data show that our summaries, in the form of micro-reviews, are more representative, diverse, and readable than the baselines.

There are also some limitations to the approach. Because our intention is to form a summary for a set of entities collectively, we assume that the entities have some common aspects. In the case when all entities in a set are completely different, by enforcing inter-density, our technique may result in few summary tips. Another limitation is the quality of the output summaries inherently depends on the quality of the input graph. If edges are accurate and sufficient, the summaries would be of high quality. If the graph is too noisy or sparse, it may affect the output.

As future work, we are also interested in further exploring other forms of multi-entity summarization including keyphrase extraction or word clouds.

References

1. J. Abello, M. G. Resende, and S. Sudarsky. Massive quasi-clique detection. In *Latin American Symposium on Theoretical Informatics*, pages 598–612. Springer, 2002.
2. E. Akkoyunlu. The enumeration of maximal cliques of large graphs. *SIAM Journal on Computing*, 2(1):1–6, 1973.
3. D. M. Blei, A. Y. Ng, and M. I. Jordan. Latent dirichlet allocation. *Journal of Machine Learning Research*, 3(Jan):993–1022, 2003.
4. P. Bogdanov, B. Baumer, P. Basu, A. Bar-Noy, and A. K. Singh. As strong as the weakest link: Mining diverse cliques in weighted graphs. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 525–540. Springer, 2013.
5. M. Brunato, H. H. Hoos, and R. Battiti. On effectively finding maximal quasi-cliques in graphs. In *International Conference on Learning and Intelligent Optimization*, pages 41–55. Springer, 2007.
6. W.-H. Chong, B. T. Dai, and E.-P. Lim. Did you expect your users to say this?: Distilling unexpected micro-reviews for venue owners. In *Proceedings of the 26th ACM Conference on Hypertext & Social Media*, pages 13–22. ACM, 2015.
7. R. Cohen and L. Katzir. The generalized maximum coverage problem. *Information Processing Letters*, 108(1):15–22, 2008.
8. G. Cornuéjols, G. L. Nemhauser, and L. A. Wolsey. The uncapacitated facility location problem. Technical report, Defense Technical Information Center (DTIC) Document, 1983.
9. M. Dawande, P. Keskinocak, J. M. Swaminathan, and S. Tayur. On bipartite and multipartite clique problems. *Journal of Algorithms*, 41(2):388–403, 2001.
10. G. Ference, M. Ye, and W.-C. Lee. Location recommendation for out-of-town users in location-based social networks. In *Proceedings of the 22nd ACM International Conference on Information & Knowledge Management*, pages 721–726. ACM, 2013.
11. K. Filippova. Multi-sentence compression: Finding shortest paths in word graphs. In *Proceedings of the 23rd International Conference on Computational Linguistics*, pages 322–330. Association for Computational Linguistics, 2010.
12. K. Ganesan, C. Zhai, and J. Han. Opinosis: a graph-based approach to abstractive summarization of highly redundant opinions. In *Proceedings of the 23rd International Conference on Computational Linguistics*, pages 340–348. Association for Computational Linguistics, 2010.
13. D. S. Hochbaum. Heuristics for the fixed cost median problem. *Mathematical Programming*, 22(1):148–162, 1982.
14. D. Jiang and J. Pei. Mining frequent cross-graph quasi-cliques. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 2(4):16, 2009.
15. R. M. Karp. Reducibility among combinatorial problems. In *Complexity of Computer Computations*, pages 85–103. Springer, 1972.
16. H. D. Kim and C. Zhai. Generating comparative summaries of contradictory opinions in text. In *Proceedings of the 18th ACM Conference on Information and Knowledge Management*, pages 385–394. ACM, 2009.
17. S. Kirkpatrick, C. D. Gelatt, M. P. Vecchi, et al. Optimization by simulated annealing. *Science*, 220(4598):671–680, 1983.

18. T. Lappas, M. Crovella, and E. Terzi. Selecting a characteristic set of reviews. In *Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 832–840. ACM, 2012.
19. T. Lappas and D. Gunopulos. Efficient confident search in large review corpora. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 195–210. Springer, 2010.
20. J. Lindqvist, J. Cranshaw, J. Wiese, J. Hong, and J. Zimmerman. I'm the mayor of my house: Examining why people use foursquare-a social-driven location sharing application. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 2409–2418. ACM, 2011.
21. G. Liu and L. Wong. Effective pruning techniques for mining quasi-cliques. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 33–49. Springer, 2008.
22. Y. Lu, P. Tsaparas, A. Ntoulas, and L. Polanyi. Exploiting social context for review quality prediction. In *Proceedings of the 19th International Conference on World Wide Web*, pages 691–700. ACM, 2010.
23. C. D. Manning, P. Raghavan, H. Schütze, et al. *Introduction to Information Retrieval*, volume 1. Cambridge University Press Cambridge, 2008.
24. R. Mihalcea and P. Tarau. Textrank: Bringing order into texts. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 404–411. Association for Computational Linguistics, 2004.
25. T.-S. Nguyen, H. W. Lauw, and P. Tsaparas. Review selection using micro-reviews. *IEEE Transactions on Knowledge and Data Engineering*, 27(4):1098–1111, 2015.
26. T.-S. Nguyen, H. W. Lauw, and P. Tsaparas. Review synthesis for micro-review summarization. In *Proceedings of the Eighth ACM International Conference on Web Search and Data Mining*, pages 169–178. ACM, 2015.
27. A. Noulas, S. Scellato, C. Mascolo, and M. Pontil. An empirical study of geographic user activity patterns in foursquare. *International Conference on Weblogs and Social Media*, 11:70–573, 2011.
28. F. M. Pajouh, Z. Miao, and B. Balasundaram. A branch-and-bound approach for maximum quasi-cliques. *Annals of Operations Research*, 216(1):145–161, 2014.
29. J. Pattillo, A. Veremyev, S. Butenko, and V. Boginski. On the maximum quasi-clique problem. *Discrete Applied Mathematics*, 161(1):244–257, 2013.
30. M. J. Paul, C. Zhai, and R. Girju. Summarizing contrastive viewpoints in opinionated text. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 66–76. Association for Computational Linguistics, 2010.
31. T. Pontes, M. Vasconcelos, J. Almeida, P. Kumaraguru, and V. Almeida. We know where you live: Privacy characterization of foursquare behavior. In *Proceedings of the 2012 ACM Conference on Ubiquitous Computing*, pages 898–905. ACM, 2012.
32. D. R. Radev, H. Jing, M. Styś, and D. Tam. Centroid-based summarization of multiple documents. *Information Processing Management*, 40(6):919–938, Nov. 2004.
33. D. B. Shmoys, É. Tardos, and K. Aardal. Approximation algorithms for facility location problems. In *Proceedings of the Twenty-Ninth Annual ACM Symposium on Theory of Computing*, pages 265–274. ACM, 1997.
34. R. Sipos and T. Joachims. Generating comparative summaries from reviews. In *Proceedings of the 22nd ACM International Conference on Conference on Information & Knowledge Management*, pages 1853–1856. ACM, 2013.
35. H. Sun, A. Morales, and X. Yan. Synthetic review spamming and defense. In *Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1088–1096. ACM, 2013.
36. I. Titov and R. McDonald. Modeling online reviews with multi-grain topic models. In *Proceedings of the 17th International Conference on World Wide Web*, pages 111–120. ACM, 2008.
37. P. Tsaparas, A. Ntoulas, and E. Terzi. Selecting a comprehensive set of reviews. In *Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 168–176. ACM, 2011.
38. C. Tsourakakis, F. Bonchi, A. Gionis, F. Gullo, and M. Tsiarli. Denser than the densest subgraph: Extracting optimal quasi-cliques with quality guarantees. In *Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 104–112. ACM, 2013.

39. T. Uno. An efficient algorithm for solving pseudo clique enumeration problem. *Algorithmica*, 56(1):3–16, 2010.
40. M. Vasconcelos, J. M. Almeida, and M. A. Gonçalves. Predicting the popularity of micro-reviews: A foursquare case study. *Information Sciences*, 325:355–374, 2015.
41. X. Wan, J. Yang, and J. Xiao. Towards an iterative reinforcement approach for simultaneous document summarization and keyword extraction. In *Association for Computational Linguistics*, volume 7, pages 552–559, 2007.
42. J. Wang, J. Cheng, and A. W.-C. Fu. Redundancy-aware maximal cliques. In *Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 122–130. ACM, 2013.
43. S. R. Yerva, F. A. Grosan, A. O. Tandrau, and K. Aberer. Tripeneer: User-based travel plan recommendation application. In *7th International AAAI Conference on Weblogs and Social Media*, number EPFL-CONF-185877, 2013.
44. Z. Yu, Y. Feng, H. Xu, and X. Zhou. Recommending travel packages based on mobile crowdsourced data. *IEEE Communications Magazine*, 52(8):56–62, 2014.
45. Z. Zeng, J. Wang, L. Zhou, and G. Karypis. Coherent closed quasi-clique discovery from large dense graph databases. In *Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 797–802. ACM, 2006.
46. C. Zhai, A. Velivelli, and B. Yu. A cross-collection mixture model for comparative text mining. In *Proceedings of the tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 743–748. ACM, 2004.