5-2018

# Entity summarization of reviews and micro-reviews

Thanh Son NGUYEN
*Singapore Management University*, tsnguyen.2013@phdis.smu.edu.sg

# ENTITY SUMMARIZATION OF REVIEWS AND MICRO-REVIEWS

NGUYEN THANH SON

SINGAPORE MANAGEMENT UNIVERSITY

2018

# Entity Summarization of Reviews and Micro-Reviews

by

Nguyen Thanh Son

Submitted to School of Information Systems in partial fulfillment of the requirements for the Degree of Doctor of Philosophy in Information Systems

**Dissertation Committee:**

Singapore Management University
2018

Entity Summarization of Reviews and Micro-Reviews

by

Nguyen Thanh Son

# Abstract

Along with the regular review content, there is a new type of user-generated content arising from the prevalence of mobile devices and social media, that is *micro-review*. 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. Both reviews and micro-reviews are useful for users to get to know the entity of interest, thus facilitating users in making their decision of purchasing or dining. However, the abundant number of both reviews and micro-reviews makes it increasingly difficult to go through them and extract the useful information. In this dissertation, we propose to summarize reviews and micro-reviews to ease users in understanding entity (or a set of entities).

We focus on two different scenarios when generating a summary: summarization for a single entity (Part I) and summarization for a set of entities (Part II). In Part I, we propose a novel mining problem, which brings together the two disparate sources of review content. The objective is to summarize the micro-reviews of an entity. The summaries are generated based on reviews' content as reviews are often coherent, well-written and comprehensive.

In Chapter 3, we use coverage of micro-reviews as an objective for *selecting* a set of reviews that covers efficiently the salient aspects of an entity. Our approach consists of a two-step process: matching review sentences to micro-reviews, and selecting a small set of reviews that covers as many micro-reviews as possible, with few sentences. We formulate this objective as a combinato-

rial optimization problem, and show how to derive an optimal solution using Integer Linear Programming. We also propose an efficient heuristic algorithm that approximates the optimal solution.

In Chapter 4, 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.

In Part II, instead of summarizing for a single entity, we focus on summarizing for a set of entities. In Chapter 5, we address the problem of summarizing the micro-reviews of multiple entities in a collection, by 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 (i.e., aspects) of 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 micro-review 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.

In Chapter 6, we consider the problem of ranking entities in terms of both relevance and quality based on reviews and reviewer expertise. The intuition is that reviews written by "experts" (for a specific query) should be considered more important in determining the rankings. We realize this intuition through a novel probabilistic model that incorporates various signals such as popularity, relevance, as well as reviewer expertise as manifested by how knowledgeable and authoritative each reviewer is.

# Contents

# List of Figures

# List of Tables

# Acknowledgements

First and foremost, I would like to express my special appreciation and gratefulness to my supervisor Professor Hady W. Lauw, for his continuous guidance, support and encouragement throughout my PhD study. His advice on both research as well as my career have been invaluable.

I would also like to thank Professor Panayiotis Tsaparas for his advice on the first three research works of this dissertation, especially for the first work.

My sincere thanks also goes to Professor Jamie Callan for his supervision when I visited Carnegie Mellon University in 2016. His guidance plays an important role in forming the fourth research work of this dissertation.

I also would like to thank my committee members, Professor Zheng Baihua, Professor David Lo, and Professor Sun Aixin, for their useful comments and suggestions during my dissertation revision and defense.

I thank my labmates, especially the ones in my research group for the stimulating discussions, suggestions and supports for the last five years.

I would like to thank Singapore Management University for the scholarship and financial supports for my study. I also would like to thank Living Analytics Research Centre for supporting my visit at Carnegie Mellon University.

Last but not least, I would like to send a special thanks to my family: my parents, my brother, my sister, my beloved wife, and especially to my unborn baby, for their consistent love and encouragement.

*This dissertation is dedicated to my family*

# Chapter 1

# Introduction

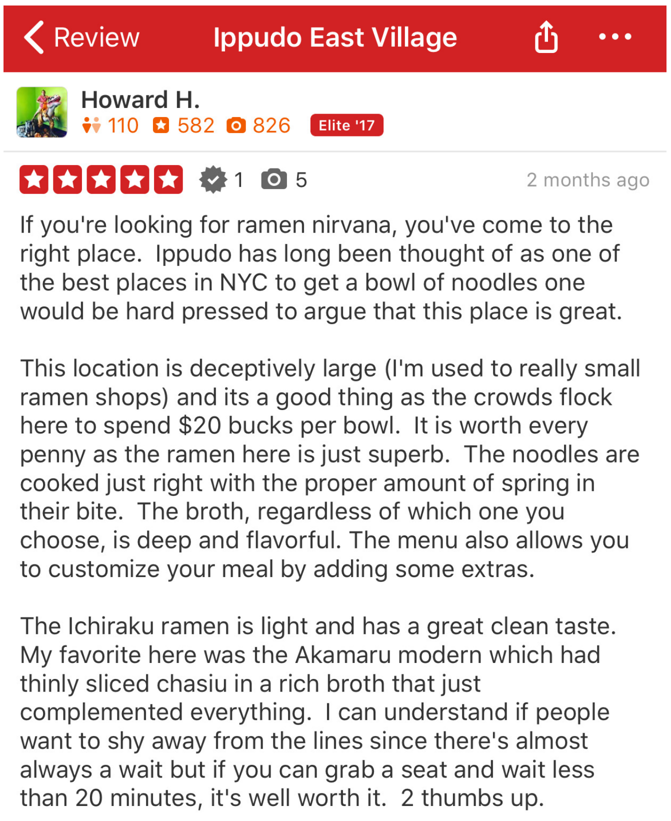
## 1.1 Summarizing Reviews and Micro-Reviews

Today, we can find ample review content in various Web sources. For instance, Amazon.com hosts product reviews, as part of an online shopping experience to assist their customers in determining which product is most suitable for their need. Yelp.com is a popular site for restaurant reviews, assisting diners to plan which restaurant to visit. Reviews are immensely useful in aiding decision-making, because they allow the readers to anticipate what their experience would potentially be based on the prior experiences of others, without having to make a trip to the store or the restaurant. Figure 1.1 shows, as an example, a review for *Ippudo East Village*[1] (or *Ippudo*) restaurant in New York. By reading the review, we can see that the user was satisfied and strongly recommended this restaurant to the people who are looking for a good ramen place. He also elaborated his opinions about the noodles, the broth, as well as his favorite selection of Akamaru Modern, and did not forget to mention the waiting time with the comment that *"it's well worth it"*.

While useful, the deluge of online reviews also causes some issues. Readers are inundated by the numerous reviews, and it is not clear which reviews are worthy of a reader's attention. For example, as of December 1$^{st}$, 2017, there

[1] `https://www.yelp.com/biz/ippudo-east-village-new-york`

Figure 1.1: A review for *Ippudo East Village* restaurant in New York



If you're looking for ramen nirvana, you've come to the right place. Ippudo has long been thought of as one of the best places in NYC to get a bowl of noodles one would be hard pressed to argue that this place is great.

This location is deceptively large (I'm used to really small ramen shops) and its a good thing as the crowds flock here to spend $20 bucks per bowl. It is worth every penny as the ramen here is just superb. The noodles are cooked just right with the proper amount of spring in their bite. The broth, regardless of which one you choose, is deep and flavorful. The menu also allows you to customize your meal by adding some extras.

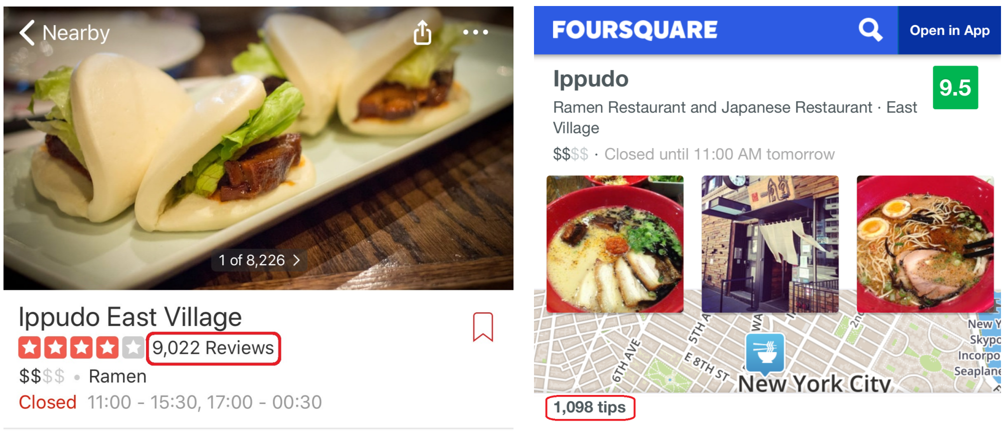
The Ichiraku ramen is light and has a great clean taste. My favorite here was the Akamaru modern which had thinly sliced chasiu in a rich broth that just complemented everything. I can understand if people want to shy away from the lines since there's almost always a wait but if you can grab a seat and wait less than 20 minutes, it's well worth it. 2 thumbs up.

are more than nine thousand reviews for *Ipuddo East Village* restaurant on Yelp.com (Figure 1.2a). This is worsened by the length and verbosity of many reviews, whose content may not be wholly relevant to the product or service being reviewed. Reviewers[2] often diverge, meandering around personal details that do not offer any insight about the place being reviewed. Furthermore, it is getting increasingly more difficult to determine the authenticity of a review, whether it has been written by a genuine customer sharing her experience, or by a spammer seeking to mislead[3]. Identifying and selecting high quality reviews to show to the users is a hard task, and it has been the focus of substantial amount of research [57, 80, 95, 175, 87, 143, 86].

**Micro-Reviews.** With the growth of social networking and micro-blogging services, we observe the emergence of a new type of online review content. This new type of content, which we term *micro-reviews*, can be found in

[2]We use "user" and "reviewer" interchangeably

[3]`http://www.businessweek.com/magazine/a-lie-detector-test-for-online-reviewers-09292011.html`



Ippudo East Village
9,022 Reviews
$$$$ • Ramen
Closed 11:00 - 15:30, 17:00 - 00:30

Ippudo
9.5
Ramen Restaurant and Japanese Restaurant · East Village
$$$$ · Closed until 11:00 AM tomorrow
1,098 tips

(a) *Ippudo* on Yelp.com (b) *Ippudo* on Foursquare.com

Figure 1.2: Number of Reviews and Tips for *Ippudo* Restaurant on Yelp.com and Foursquare.com, Respectively.

micro-blogging services that allow users to "check-in", indicating their current location or activity. For example, at Foursquare, users check in at local venues, such as restaurants, bars, coffee shops. After checking in, a user may choose to leave a tip about their experience, effectively a *micro-review* of the place. Different from regular reviews, micro-reviews are shorter by design, e.g., Foursquare imposes a limit of 200 characters to each tip[4]. In addition to Foursquare, there are also alternative sources for micro-reviews in several domains. For instance, Jiepang (in Chinese) and VK (in Russian) feature similar services, and Flicktweets to post micro-reviews on movies.

In the case of restaurants, tips are frequently recommendations (e.g., what to order), opinions (what is great or not), or actual "tips". For example, the following are some Foursquare tips for a popular burger joint in New York:

> *"Order the Trifecta - a shack stack, cheese fries and a milk shake and then see if you don't get winded on your way home. Head there early to avoid the line."* (a recommendation)

> *"SHROOM BURGER!!! Its the ONLY good veggie burger in the city!"* (an opinion)

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

> *"If you only want ice cream there is a short line. The A line is for food and shakes and is long. The B line has little or no waiting for ice cream & floats."* (an actual tip).

Those are usually for different aspects of the location being reviewed. Let us take for an example *Ippudo*[5] restaurant in New York. On Foursquare, users have left more than a thousand of tips[6] (Figure 1.2b) about various aspects, about signature dishes:

> *"Get the pork buns (it's true; better than Momofoku) and Akamaru Modern. You will leave a very happy person."*

waiting time:

> *"Go in the afternoon to avoid having to wait. Sat afternoon = 15 min wait."*

or reservation:

> *"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.

**Comparison of Reviews and Micro-Reviews.** Micro-reviews serve as an alternative source of content to reviews for readers interested in finding information about a place. Table 1.1 shows the comparison of the properties of micro-reviews and reviews. Micro-reviews have several advantages. *First*, due to the length restriction, micro-reviews are *concise and distilled*, identifying the most salient or pertinent points about the place according to the author. *Second*, because some micro-reviews are written on site, right when the user has checked in, they are *spontaneous*, expressing the author's immediate and

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

[6]Information checked on December 1st, 2017

Table 1.1: Micro-Reviews vs. Reviews

| **Micro-Reviews** | **Reviews** |
|---|---|
| On-site Check-ins | Off-site |
| Brusque and Curt | Well-written, Narrative/Descriptive Flow |
| Concise and Distilled | Elaborate and Comprehensive |
| Reactive and Spontaneous | Reflective and Contemplative |

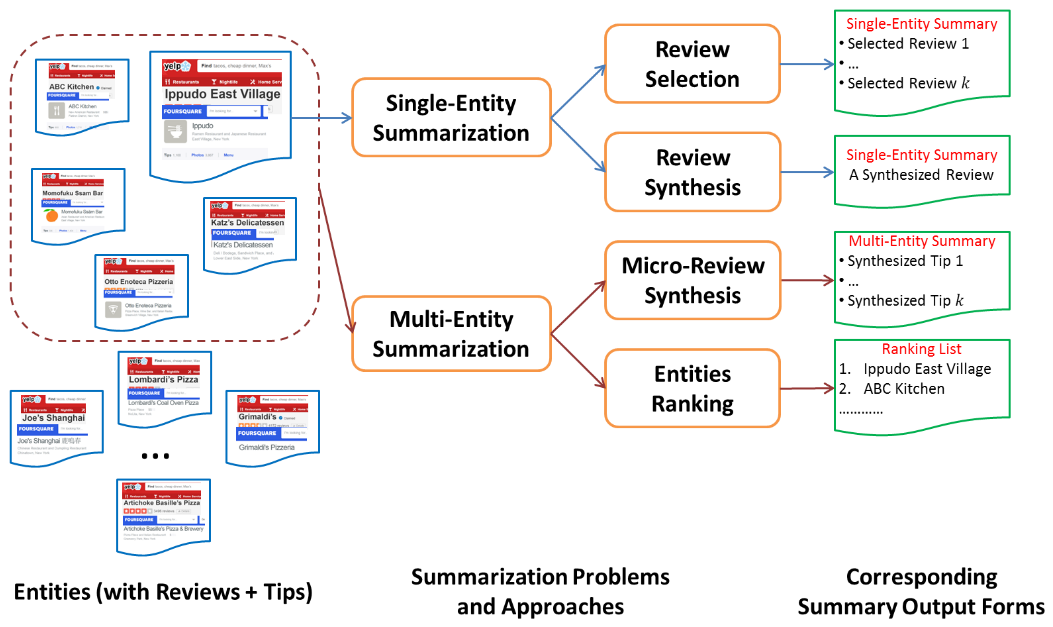
unadulterated reaction to her experience. *Third*, because most authors check in by mobile apps, these authors are likely at the place when leaving the tips, which makes the tips more likely to be *authentic*. Micro-blogging sites also have the ability, if necessary, to filter out tips without an accompanying check-in, thus, boosting the authenticity of the tips.

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

Reviews and micro-reviews are useful for users to get to know the entity of interest, thus facilitating users in making their decision of purchasing or dining. However, the abundant number of both reviews and micro-reviews cause users some inconvenience. For example, with the number of reviews and tips as shown in Figure 1.2, users probably cannot read all of them and they do not know which reviews or tips they should read either. Therefore, it raises the need of having summaries for reviews and/or tips.

There are different types of summarization problem depending on the needs of users. Imagine the scenario when a user wants to know something about an area that has many entities (venues). For example, when she is in an unfamiliar area and she wants to know what offers (e.g., types of food) are available in the

Figure 1.3: Different Summarization Scenarios with the Corresponding Problems and Approaches, as well as the Summary Output Forms



area. It raises the problem of *multi-entity summarization*. Another scenario is when a user needs to know something about a specific entity. That is the problem of *single-entity summarization*. Therefore, in this dissertation, we look into both single-entity summarization and multi-entity summarization.

We group our research works into two parts. In the first part, single-entity summarization, we focus on summarizing for a single entity. The second part, multi-entity summarization, focuses on creating a summary for a set of entities. A summary can be in the form of a subset of representative reviews/tips, or a synthesized review/tip, or a ranked list of entities. Figure 1.3 illustrates the two different summarization scenarios with the corresponding problems and approaches, as well as the summary output forms. In the next two sections, we present the overviews of our research on single-entity summarization and multi-entity summarization.

## 1.2 Single-Entity Summarization

For single-entity summarization, the objective is to summarize the micro-reviews of an entity. The summaries are generated based on reviews' content as reviews are often coherent, well-written pieces of text, produced by an author who seeks to comprehensively describe his/her experience with the entity. For generating the summaries, we choose to either select a small number of reviews (review selection, Chapter 3) or synthesize a review (review synthesis, Chapter 4).

**Review Selection.** A summary could be in the form of a subset of selected reviews that most represent the entity. In Chapter 3, we present the research on *review selection based on coverage of micro-reviews.* The problem of review selection has been studied in the past [87, 143, 86]. In the prior works, this is mainly modeled as a *coverage problem*, where the selected reviews are required to cover the different aspects of the entity (e.g., product attributes), and the polarity of opinions about the entity (positive and negative). To extract the aspects covered by a review, and the sentiment polarity, off-the-shelf tools for supervised techniques are usually applied. Such approaches, although generally successful, cannot generalize to arbitrary domains. Unsupervised techniques, e.g., topic modeling, have also been applied (e.g., [97]), however they suffer from the broadness of the topic definition.

We view tips as a crowdsourced way to obtain the aspects of an entity that the users care about, as well as the sentiment of the users. By covering the tips, we effectively identify the review content that is important, and the aspects of the entity upon which the reviews need to expand and elaborate. In the formulation, which we outline in Chapter 3, we make sure that the selected reviews are compact, that is, the content does not diverge from what is important about the reviewed entity. We view this as an important constraint, especially for viewing on mobile devices, where screens are small, and time is short.

To perform the selection, we first need to determine when a review covers a tip. Given the matching between reviews and tips, we then select a small number of reviews that cover as many tips as possible. We also introduce the notion of selection *efficiency*, which assures the selected set to not contain too many sentences that do not cover any tip. A review is said to cover a tip if they have high *syntactic similarity* (based on *cosine similarity*), high *semantic similarity* (based on *topic models*), and high *sentiment similarity* (based on a *maximum entropy* classifier). We formulate the review selection problem as a *maximum coverage with efficiency* problem. To find the optimal solutions, we show how to adapt a known Integer Linear Programming (ILP) formulation [150] for the *maximum coverage* problem to take into account the efficiency constraints. Since the problems are NP-hard, it is not tractable to find the optimal solution for very large problem sizes. We design a greedy algorithm, which is empirically shown to be virtually identical to the optimal solutions in coverage and efficiency, and yet is much faster computationally.

**Review Synthesis.** One issue with the review selection approach is that when selecting a review, we have to use the whole review, which may contain irrelevant content. Moreover, selected reviews may cover the same aspect(s) of the entity, e.g., about a famous dish of a restaurant. We therefore propose to *synthesize* a new "review", by taking the "best" parts of some reviews, and putting them together into a text summary. In Chapter 4, we present the problem of *review synthesis as concise representation of micro-reviews*: 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 4.1, 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 4.1.3). 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 4.2). In Section 4.3, 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.

The problem of review selection is fundamentally different from review synthesis. Review selection imposes the restriction of selecting multiple *full* reviews from the existing corpus, while review synthesis 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", review synthesis does 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 Section 4.3, we compare synthesized review with the selected review, and more generally to all existing reviews in the dataset as well.

## 1.3 Multi-Entity Summarization

Sometimes, having a summary for a single entity is not enough as users may not even know which offer (e.g., type of food) is available around a particular area or which restaurant they should pay attention to, given their own preferences. It raises the need of having a summary that is not for a single entity, but for multiple entities. In Part II, we present research on multi-entity summarization. In Chapter 5, we summarize the micro-reviews belonging to the entities in a collection by synthesizing micro-reviews which describe the most popular-common aspects shared among the input entities. In Chapter 6, we summarize the reviews belonging to the entities in a collection with regards to the query of interest. The summary is in the form of a ranked list of the input entities.

**Micro-Review Synthesis for Multi-Entity Summarization.** Location-based social networks (LBSNs) are increasingly popular as a travel tool to get a glimpse of what is available in a new unfamiliar locality [49, 164, 167]. 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).

In Chapter 5, we present the problem of *micro-review synthesis based on multi-entity quasi-cliques.* We propose to generate or synthesize micro-reviews for a collection of entities, by summarizing the micro-reviews of the underlying entities within the collection. 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.

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 5.1. 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 5.2, 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 5.3, 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.

**Query-Dependent Ranking of Entities.** There are many occasions when one seeks to identify a manageable number of options to potentially meet a specific need. For instance, one may travel as a tourist to a new city, and seek dining options that meet specific requirement or dietary restriction. We may find ourselves in new neighborhood, and wish to satisfy our craving for our favorite dish. We could try to browse the reviews of the candidate restaurants around the area as reviews contain a lot of useful information, and allow us to imagine what our experience would be like before taking a plunge. However, there are just too many potential candidates, and some of these candidates have a huge number of reviews. For instance, as of the time of writing, Yelp lists 4005 restaurants in Las Vegas. The most-reviewed restaurant in our dataset has 5558 reviews. Moreover, even if we could have a “perfect” summarizer that could generate a “perfect” summary for each restaurant, the amount of content that we still need to go through make it infeasible. Evidently, we need a simpler way to navigate this huge corpus of information.

Assuming that the user need can be expressed in terms of a query, we can formulate this into a form of retrieval problem, whereby the objective is to retrieve the top-ranked entities (e.g., products, restaurants) that are most likely to meet the need expressed by the query. In Chapter 6, we consider the problem of deriving this ranking based on the corpus of reviews. One advantage of relying on reviews is that once we return the ranked list of entities, the querier can peruse the reviews to further examine the suitability of the entities.

A naive treatment is to associate each entity with a document describing it, and to retrieve the most relevant document to the query. For instance, since an entity can be described by its reviews, one may create a “document” by concatenating all its reviews. One could then apply standard retrieval method to retrieve the most relevant “documents”.

However, such an approach treats reviews merely as text, and ignores the intrinsic nature of reviews. In addition to text, reviews are also associated with other important information, such as ratings as well as the reviewers. Because it ignores the intrinsic nature of reviews, the naive approach has several shortcomings. For one, by concatenating the reviews, one may lose important signals. For instance, an entity with more reviews may signal popularity. For another, the reviews represent diverse opinions, and they may not all agree with one another. Moreover, the person behind the review is as important as (if not more than) the content.

In this work, we propose two models for review-based ranking of entities. Both our proposed models are based on probabilistic language modeling techniques that have been shown useful for various information retrieval tasks, including *expertise retrieval* [7]. Each model ranks the entities based on the probability that an entity is apt for the query, but the two models differ in how this is done. The first model is purely based on reviews that seek to capture signals, such as how popular an entity is, how relevant a review is to the query, as well as the degree to which a review lends its support to the entity of interest. In addition to those used in the first model, the second model incorporate the expertise of reviewers when deriving the rankings. Specifically, during the estimation, each review is now weighted by how likely its author is an "expert" with regards to the query.

Experiments on Yelp reviews showcase how incorporating review features, as well as reviewer expertise, helps to improve the quality of the rankings significantly over the traditional approaches that primarily consider relevance alone.

## 1.4 Contributions

- Part I:
    - Although the content of micro-blogging sites has been studied extensively, micro-reviews is a source of content that has been largely overlooked in the literature. In the works in Part I, we study micro-reviews, and we show how they can be used for the problem of

review selection (Chapter 3) and review synthesis (Chapter 4). To the best of our knowledge we are the first to mine micro-reviews such as Foursquare tips and combine them with full-text reviews such as Yelp reviews.

- In Chapter 3, our work introduces a novel formulation of review selection, where the goal is to maximize coverage while ensuring efficiency, leading to novel coverage problems. The coverage problems we consider are of broader interest, and they could find applications to different domains. We consider approximation and heuristic algorithms, and study them experimentally, demonstrating quantitatively and qualitatively the benefits of our approach. We also propose an Integer Linear Programming (ILP) formulation, and provide an optimal algorithm. This allows us to quantify the approximation quality of the greedy heuristics. We investigate the number of reviews needed to obtain perfect coverage through an alternative formulation inspired by set cover. This work has been published as a conference paper in the 22nd ACM International Conference on Information & Knowledge Management (CIKM) 2013 [115] and as a journal paper in the IEEE Transactions on Knowledge and Data Engineering (TKDE) 2015 [116].
- In Chapter 4, 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. 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. 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. This work has been published as a conference paper in the Eighth ACM International Conference on Web Search and Data Mining (WSDM) 2015 [117].

- Part II:
  - For the work in Chapter 5, 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. 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. 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, 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. This work has been published as a journal paper in Data Mining and Knowledge Discovery (DAMI) 2017 [118].
  - In Chapter 6, we introduce the problem of ranking entities for a

given query. To retrieve a ranking, in addition to the traditional use of reviews merely as text contents, we propose to take into account the expertise of users who wrote the reviews. We propose a novel probabilistic model that incorporates various signals extracted from reviews (e.g., popularity, relevance) and reviewer expertise. We estimate the expertise of a user by considering her *authoritativeness* in general and how *knowledgeable* she is about the query.

# Chapter 2

# Literature Review

This dissertation focuses on summarizing entities using reviews and micro-reviews. The work is related to *text summarization*, as we work with the textual content of reviews and micro-reviews. Our research is also related to *mining reviews and micro-reviews.* We now discuss the literature review for text summarization and mining reviews and micro-reviews.

## 2.1 Text Summarization

The objective of text summarization is to reduce one or more text documents into a compressed text. There are several main methodologies. One is *extractive*, which selects text snippets (e.g., sentences, paragraphs) from the documents to be summarized. Another methodology is *abstractive*, which generates new contents (e.g., sentences) from an abstract representation. Figure 2.1 shows our categorization of the previous works in *text summarization* and Table 2.1 shows the related works for the different variations. We categorize document summarization into two main categories: *update summarization* and *static summarization.*

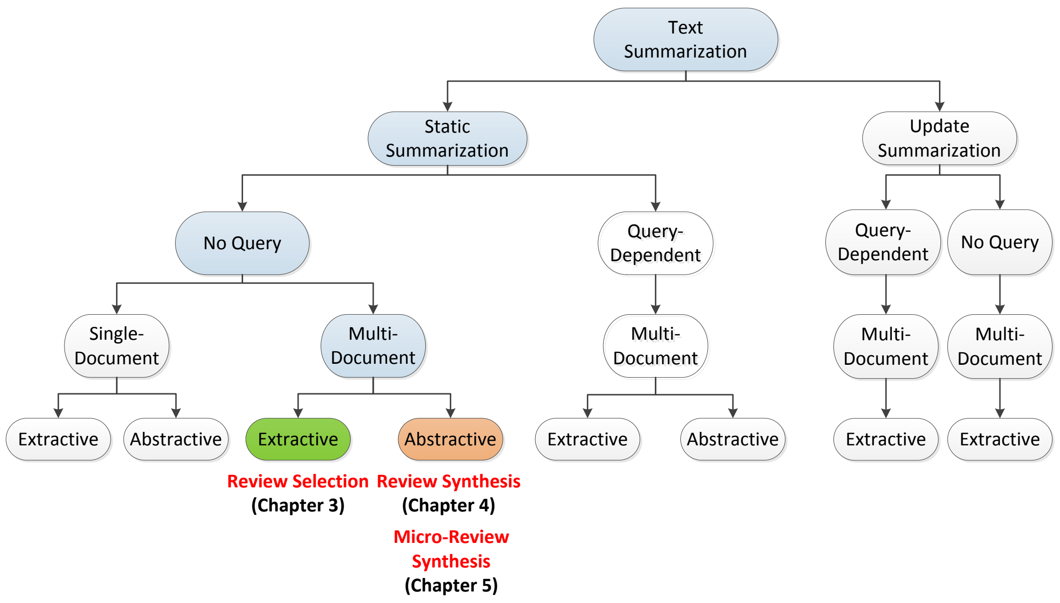


Figure 2.1: Text Summarization Taxonomy with Our Corresponding Research

### 2.1.1 Update Summarization

In update summarization, the set of input documents to be summarized is continuously added new documents and the goal is to summarize the new ones with the assumption that the readers already read the earlier documents [43, 65, 154]. A summary might be generated to summarize general information of a documents set (i.e., no query) or specific information with regards to a query of interest (i.e., query-dependent). One of the main challenges in update summarization is the capability of extracting new information, rather than getting the information already appeared in the earlier documents. Most of the research on update summarization adopt *extractive* methodology.

For query-dependent update summarization, Wenjie et al. [159] propose a graph-based sentence ranking algorithm, namely *PNR*$^2$ (Ranking with Positive and Negative Reinforcement). The algorithm is based on the intuition that sentences belonging to the same set of documents (i.e., earlier or new set) will have *positive* influence, whereas, sentences belonging to the different sets will have *negative* influence. This helps the ranking process to favor new information rather than the old information already appeared in the earlier documents. Another graph-based approach is proposed by He et al. [65], in which, they build a *similarity graph* from the input sentences. On one hand, *iterative feedback mechanism based evolutionary manifold-ranking* is applied on the similarity graph to rank the sentences based on the *evolutionary*, *important*, and *query-relevant* information. On the other hand, the authors propose to combine the spectral clustering with the evolutionary manifold-ranking to improve the coverage of the summary content. Wang and Li [154] propose an approach based on incremental hierarchical clustering (IHC) to obtain the sentence hierarchy with the capability of obtaining the most representative sentence at each tree node. Sentence clusters will be re-organized as soon as there is new documents added. The IHC method used is *COBWEB*, which is proposed by Fisher [52].

Although most of the research on update summarization focus on query-dependent, Delort and Alfonseca [43] propose a nonparametric Bayesian approach, called *DualSum*, for query-independent update summarization. In *DualSum*, documents are modeled as a bag of words which are assumingly sampled from a mixture of latent topics. *DualSum* is designed to capture the differences between the early and the new documents by learning *joint topic* (i.e., common information) and *update topic* (i.e., new information). This approach is a variation of Latent Dirichlet Allocation (LDA) [16].

### 2.1.2 Static Summarization

Our research is more on static summarization, in which the input documents are unchanged. Henceforth, the term *summarization* refers to *static summarization.* Depends on the problem setting, a summary could be used to represent general information or query-dependent information of a single document or multiple documents.

**Query-Dependent.** Carbonell and Goldstein [23] propose an approach that aims to reduce the redundancy while maintaining query relevance when

Table 2.1: Text Summarization Variations and the Corresponding Related Works

| | *Query Relevance* | *Type of Input* | *Extractive* | *Abstractive* |
|---|---|---|---|---|
| Static Summarization | Query-Dependent | Multi-Doc | Carbonell et al. [23]<br>Varadarajan et al. [147]<br>Lin et al. [92]<br>Ji et al. [74]<br>Glavaš et al. [59]<br>Canhasi et al. [20]<br>Yang et al. [161] | Ji et al. [74]<br>Wang et al. [158] |
| | No Query | Single Doc | Mihalcea et al. [109]<br>Mihalcea et al. [110]<br>Shen et al. [134]<br>Wan et al. [153]<br>Litvak et al. [93]<br>Mendoza et al. [107] | Rush et al. [132] |
| | | Multi-Doc | Goldstein et al. [60]<br>Erkan et al. [46]<br>Radev et al. [130]<br>Mihalcea et al. [110]<br>Wan et al. [151]<br>McDonald [105]<br>Arora et al. [5]<br>Wan et al. [152]<br>Wang et al. [155]<br>Haghighi et al. [64]<br>Wang et al. [156]<br>Celikyilmaz et al. [26]<br>Lin et al. [92]<br>Sipos et al. [136]<br>Carenini et al. [24]<br>Christensen et al. [35]<br>Ferreira et al. [50]<br>Hong et al. [68]<br>Kumar et al. [84]<br>Cao et al. [22]<br>Peyrard et al. [127]<br>Tohalino et al. [142] | Barzilay et al. [9]<br>McKeown et al. [106]<br>Filippova [51]<br>Ganesan et al. [54]<br>Ganesan et al. [55]<br>Carenini et al. [24]<br>Bing et al. [13] |
| Update Summarization | Query-Dependent | Multi-Doc | Hickl et al. [66]<br>Wenjie et al. [159]<br>Wang et al. [154]<br>He et al. [65]<br>Li et al. [90] | |
| | No Query | Multi-Doc | Delort et al. [43] | |

producing a summary, namely *Maximal Marginal Relevance* (MMR). The idea is to select the sentences that are not only relevant to the query, but also least similar to the already selected ones. This is to maximize the diversity of the generated summary while guarantee query-relevance. Continuing this idea, Lin and Bilmes [92] propose a class of submodular functions meant for extractive document summarization. To preserve the monotonicity of objective functions, instead of negatively penalizing redundancy, they propose to positively reward diversity. Yang et al. [161] propose a hierarchical Bayesian topic model to improve the process of determining sentences similarity. The model is built upon the hierarchical Latent Dirichlet Allocation model [15] that incorporates the concepts of n-grams into hierarchical latent topics to capture the topics with the contextual information (i.e., contextual topics).

Varadarajan and Hristidis [147] propose to represent a document as a weighted document graph. Given a query of interest, the summary is the minimum spanning tree on the corresponding graph that contains all the keywords of the query. Canhasi and Kononenko [20] formalize the problem of query-dependent document summarization as the weighted archetypal analysis problem. The main idea is to cluster sentences into weighted archetypes and sentences are selected based on the archetype membership weights. Focusing on event-oriented retrieval and summarization, Glavaš and Šnajder [59] propose to represent documents by *event graphs*, in which each vertex is an event mention, and edges denote temporal relations between them. A summary is then derived by selecting sentences ranked by their aggregated event mentions' scores.

Ji et al. [74] propose to apply Information Extraction (IE) to solve the query-dependent multi-entity summarization problem. Abstractive summaries can be generated based on the IE results. The authors show how to improve over extractive summarizer by integrate IE results for re-ranking sentences and removing redundancy. Wang et al. [158] applies a sentence compression based framework (i.e., abstractive) to find summaries. They propose three types of compression approaches, i.e., rule-based, sequence-based and tree-based. The best performance achieved when incorporating measures of query relevance, content importance, redundancy and language quality.

**Query-Independent (No Query).**

*Extractive Summarization.* As mentioned, one methodology for document(s) summarization is *extractive* , in which the key requirement is the capability of selecting sentences for constructing the summary. One of the popular approaches for solving this problem is graph-based, in which a document (or a set of documents) is represented as a graph with vertices are text snippets to be ranked (e.g., words, sentences) and edges are formed based on the relevance and the relationship between the snippets [109, 153, 93]. Graph-based ranking algorithms are then applied on the graph to rank the text snippets [109, 110, 151].

Erkan and Radev [46] introduce a stochastic graph-based method, namely LexRank, to rank sentences based on the concept of eigenvector centrality in a graph representation of sentences. Mihalcea and Tarau [109] propose a ranking model, namely TextRank, that conducts random walks on a graph of text snippets, and selects the snippets with the highest stationary probabilities. Mihalcea and Tarau [110] show that graph-based approaches could be applied for different languages. To solve the multi-document summarization problem, they perform single-document summarization on the summaries generated for each individual document. In order to avoid duplication, links between sentences whose similarity exceeds a threshold are removed. Similar idea is proposed by Wan et al. [153], but simultaneously considers both the rankings of words and sentences. In addition to applying graph-based ranking algorithm, Litvak and Last [93] introduce a supervised approach for the problem of keyword extraction which is formulated as classification problem with labels of YES or NO depends on whether the words appear in the ground-truth summary. Christensen et al. [35] propose to jointly optimize coherence and salience when generating a summary for multi-document summarization. The core representation is a discourse graph approximating the discourse relations across sentences.

Wan and Yang [152] propose to integrate the cluster information of the sentences in a document set with link analysis techniques, such as Markov Random Walk Model or HITS, to compute the salient scores of the sentences. Wang et al. [155] propose a summarization framework based on sentence-level semantic analysis and symmetric non-negative matrix factorization. The basic idea is to group sentences into clusters by applying symmetric matrix factorization on the sentence-sentence similarity matrix. The summary is formed by selecting the most informative sentences from each cluster. Ferreira et al. [50] also propose to score sentences based on sentence clusters information with the use of statistic similarities and linguistic treatment.

Topic model based approaches are also widely considered. Arora and Ravindran [5] propose to generate the summary by selecting the sentences that represent the different events of the input documents. The events are extracted by applying Latent Dirichlet Allocation (LDA) [16]. Haghighi and Vanderwende [64] explore different generative probabilistic models for multi-document summarization, including a simple word frequency based model, Kullback-Lieber divergence based, LDA based, and hierarchical LDA-style [62] based model. Wang et al. [156] propose a Bayesian sentence-based topic model to capture the topics embedded in the documents that helps guiding the sentence selection when generating summaries. Celikyilmaz and Hakkani-Tur [26] propose a two-step hybrid model that (step 1) extracts hierarchical topics on sentence level building on hierarchical LDA [62] for calculating sentences' scores which will be used in step 2 to train a regression model for scoring new sentences.

Radev et al. [130] introduce a centroid-based summarizer, namely MEAD, that 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. Carenini et al. [24] propose two different approaches including an adapted version of MEAD [130] and a natural language generation based approach. Shen et al. [134] treat the document summarization task as a sequence labeling problem and apply a Conditional Random Fields [85] based framework to label the sentences in the sequence with 1 (i.e., in the summary) and 0 (i.e., not in the summary). McDonald [105] formulates multi-document summarization as a inference problem and shows that a dynamic programming algorithm provides near optimal results (generated by Integer Linear Programming) with preferable scaling properties.

Mendoza et al. [107] formulate the single-document extractive summarization as a binary optimization problem where the quality of a solution is evaluated based on the weighting of individual statistical features of each sentence, such as position, sentence length, and the coherence of the sentences in the summary. They propose a memetic algorithm to direct the exploration towards the most promising regions of the search space that optimize the linear combination of the features. Hong and Nenkova [68] introduce a supervised model combining different features from unsupervised approaches and features including position, part-of-speech, polarity, etc, for predicting summary keywords. Kumar et al. [84] propose a scoring sentence approach using fuzzy reasoning on cross-document relation information to produce high quality multi-document news summaries.

Cao et al. [22] develop a sentences ranking system based on Recursive Neural Networks that transform the ranking task into a hierarchical regression process. Peyrard and Kohler [127] propose to first learn a scoring function based on supervised learning to assign ROUGE score for each sentence. The summary is then generated by selecting sentences that maximize the ROUGE score of the summary. The sentence selection is formulated as a budgeted subset selection optimization problem. Tohalino and Amancio [142] propose to generate summaries by modeling input documents as a multi-layer network where each node represents a sentence and edges are formed based on the cosine similarity between the corresponding sentences. They then measure the relevance (centrality) of each node by applying network measurements (e.g., degree, shortest paths, and PageRank) and select the best (highest or lowest scored) nodes as the summary. The redundancy is avoided by considering the similarity between a candidate sentence with the selected sentences. For the work in Chapter 3, considering micro-reviews as text documents, the problem can be considered as an instance of extractive multi-document summarization. The key difference of our work is that the summary is generated by selecting existing reviews, which is an independent corpus, not from the corpus to be summarized (i.e., micro-reviews).

*Abstractive Summarization.* Another methodology for document(s) summarization is *abstractive*, exemplified by the works from Ganesan et al. [54] and Filippova [51], which generate 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. Barzilay et al. [9] propose to generate new sentences from common phrases identified from similar sentences. To generate fluent sentences from phrases, they apply grammatical constraints, paragraphing and different techniques to map predicate argument structures. In [55], the summary is a list of short phrases

Rush et al. [132] propose an approach that combines a neural network language model with a contextual input encoder to generate each word of a summary given the input sentence. The encoder learns a latent soft alignment over the input text to help create the summary. The encoder and the generation model are jointly trained. Bing et al. [13] propose to construct new sentences by exploring fine-grained syntactic units, i.e., noun/verb phrases. They extract concepts and facts (i.e., phrases) from the input documents. They formulate the phrases selection and merging using Integer Linear Programming that maximizes the salience of phrases and satisfies the sentence construction constraints.

Our work in Chapter 4 focuses on review synthesis in which we synthesize a new review, instead of using an existing one. The key difference from the works on abstractive summarization is that the source to synthesize the new review is independent from the one to be summarized (i.e., tips). To validate our approach, we will compare against exemplars of both extractive (i.e., MEAD [130]) and abstractive (i.e., Opinosis [54]) methods in the corresponding experiments (see Section 4.3). In Chapter 5, we work with multiple text corpora. Some related works are based on topic modeling. Zhai et al. [171] model topics with a common component across the corpora, and a unique component within each corpus. Titov and McDonald [141] model 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 [124]. The objective is to align sentences across documents, either based on similarity [136] or contradiction [79].

## 2.2 Mining Reviews and Micro-Reviews

Figure 2.2 shows the taxonomy for the task of mining reviews and micro-reviews and Table 2.2 shows the related works grouped into the corresponding variations of the task.

**Review Assessment.** There are works on assessing reviews, such as spam detection [2], predicting the helpfulness or the quality of a review [133], or estimating the impact of reviews on selling products [166]. Costa et al. [39] work on the problem of detecting spam tips on Apontador (a popular Brazilian location-based social network system) using Random Forest classifier

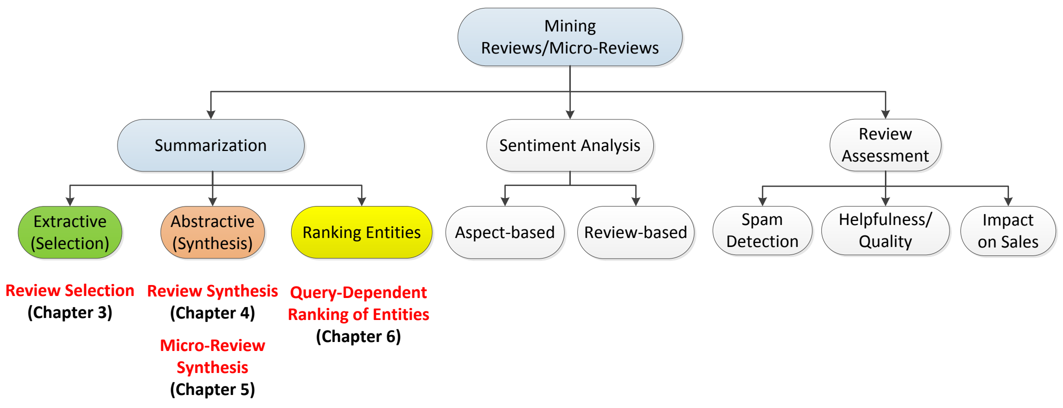


Figure 2.2: Mining Reviews/Micro-Reviews Taxonomy with Our Corresponding Research

[18]. Aggarwal et al. [2] present research on detecting spam tips and tip spammers (i.e., users who post spam tips) on Foursquare data. They identify four categories of spamming behavior: Advertising, Self-promotion, Abusive and Malicious. Machine learning techniques are applied on manually labeled data to train automatic detection tools. In contrast, Sun et al. [139] introduce a technique to synthesize spam reviews by replacing sentences of a truthful review that could fail feature-based detection algorithms. At the same time, they also propose a general framework to defend such synthesized reviews.

For the task of reviews helpfulness prediction, Kim et al. [80] propose to apply Support Vector Machine to assess how helpful a review is with various types of features including structural features, lexical features (e.g., unigram, bigram), syntactic features, etc. Liu et al. [95] show that the helpfulness of a review depends on three factors: the reviewer's expertise, the writing style and the timeliness of the review. They present a nonlinear regression model based on the factors for predicting reviews' helpfulness. Lu et al. [96] propose to incorporate social context information derived from authors' identities and social networks when assessing reviews by adding regularization constraints to the text-based predictor. Cao et al. [21] empirically examine the impact of different features, including basic, stylistic, and semantic of characteristics of online reviews on reviews' helpfulness. They show that semantic characteristics are more influential than the others. Salehan and Kim [133] analyze reviews' helpfulness based on mining sentiment of different components of a review (e.g., title, main content).

Ghose and Ipeirotis [57] propose two ranking mechanisms to not only rank reviews by the helpfulness but also rank reviews by the impact on product sales. They learn the effect of the subjectivity of reviews' sentences on product sales and helpfulness. Continuing working on this problem setting, Ghose et al. [58] further analyze the effects of more reviews' features including *readability*, *spelling errors* and reviewer's self-disclosed identity displayed next to a review. Archak et al. [4] demonstrate how textual data can be used to learn prediction models for future sales changes. Yu et al. [166] propose a model for production sales prediction based on sentiment information, called Autoregressive Sentiment Aware (ARSA) model. Experiments on ARSA show that sentiments play an important role in predicting future sales performance. The authors also propose a model for predicting the quality of a review and show that review quality has impact on product sales prediction as well.

**Sentiment Analysis.**

*Review-based Sentiment Analysis.* At a coarse-grain level, review-based sentiment analysis aims to identify the sentiment polarity of the whole review. Turney [145] classify reviews into *thumbs up* (positive) and *thumbs down* (negative) based on the average semantic orientation of the phrases in the review that contain adjectives or adverbs. The semantic orientation of a phrase is calculated as the mutual information with positive or negative words.

Supervised learning approaches are widely used for review-based sentiment analysis [163, 137]. Dave et al. [41] present a sentiment classifier using features from information retrieval. Chaovalit and Zhou [28] investigate movie review sentiment analysis using machine learning and semantic orientation approaches. They find that movie review mining is more challenging than other type of reviews mining as movie reviews have more ironic words. Ye et al.

Table 2.2: Mining Reviews/Micro-Reviews Variations and the Corresponding Related Works

| | | | |
|---|---|---|---|
| Review Assessment | Spam Detection | Costa et al. [39]<br>Aggarwal et al. [2]<br>Sun et al. [139] | |
| | Helpfulness/ Quality | Kim et al. [80]<br>Chevalier et al. [32]<br>Ghose et al. [57]<br>Liu et al. [95] | Lu et al. [96]<br>Ghose et al. [58]<br>Cao et al. [21]<br>Salehan et al. [133] |
| | Impact on Sales | Archak et al. [4]<br>Yu et al. [166] | |
| Sentiment Analysis | Aspect-based | Hu et al. [69]<br>Hu et al. [70]<br>Zhuang et al. [176]<br>Popescu et al. [129]<br>Ding et al. [44]<br>Ding et al. [45]<br>Meng et al. [108]<br>Kim et al. [81] | Sun et al. [140]<br>Zhang et al. [174]<br>Lu et al. [97]<br>Somprasertsri et al. [138]<br>Bjørkelund et al. [14]<br>Chen et al. [29]<br>Marrese-Taylor et al. [103] |
| | Review-based | Turney [145]<br>Dave et al. [41]<br>Hu et al. [71]<br>Chaovalit et al. [28]<br>Kim et al. [81]<br>Ye et al. [163] | Leung et al. [89]<br>Kouloumpis et al. [83]<br>Smeureanu et al. [137]<br>Moraes et al. [113]<br>Moraes et al. [112] |
| Review Summarization | Extractive (Selection) | Lappas et al. [87]<br>Tsaparas et al. [143]<br>Lappas et al. [86]<br>Carenini et al. [24] | Yu et al. [165]<br>Chen et al. [30]<br>Chong et al. [33] |
| | Abstractive (Synthesis) | Ganesan et al. [54]<br>Sun et al. [139] | |
| | Ranking Entities | Jindal et al. [76]<br>Sun et al. [140]<br>Xu et al. [160]<br>Zhang et al. [173]<br>Zhang et al. [172]<br>Bjørkelund et al. [14]<br>Li et al. [91]<br>Sipos et al. [136] | Chong et al. [34]<br>Ganesan et al. [53]<br>Makris et al. [98]<br>Gourgaris et al. [61]<br>Peleja et al. [126]<br>Chutmongkolporn et al. [36]<br>Bashir et al. [11] |

[163] compare three supervised machine learning algorithms of Naïve Bayes, SVM and the character based N-gram model for review sentiment analysis on travel blogs.

Moraes et al. [112] present an empirical study comparing different approaches of supervised and unsupervised learning on Foursquare tips. Kouloumpis et al. [83] propose a supervised learning approach for predicting sentiment for Twitter messages that utilizes linguistic features, e.g., n-gram features, lexicon features, part-of-speech features and micro-blogging features.

*Aspect-based Sentiment Analysis.* For a finer-grain, the goal of aspect-based sentiment analysis is to analyze the sentiment orientation of individual aspects mentioned in a review, rather than the sentiment of the review as a whole. Hu and Liu [69, 70] summarize reviews in terms of the statistical distributions of sentiments for various features or aspect of a product. Extending the work from Hu and Liu [69, 70], Marrese-Taylor et al. [103] propose a deterministic semantic orientation approach for aspect-based opinion mining of tourism reviews. Producing the same output format as in [70], Zhuang et al. [176] propose a multi-knowledge based approach for extracting features and sentiments of movie reviews. Lu et al. [97] propose to generate "rated aspect summaries" for short comments by identifying major aspect clusters of the short comments, predicting ratings for each aspect and extracting representative phrases to explain the aspect rating.

Popescu and Etzioni [129] introduce an unsupervised information extraction system, namely OPINE, which is built on top of KnowItAll [47]. Given a product and the corresponding set of reviews, OPINE outputs a set of product features with a list of strength-based ranked opinions for each feature. Ding et al. [44, 45] propose a holistic lexicon-based approach exploiting external evidences and linguistic conventions of natural language expressions that allow the system to handle context-dependent opinion words. Kim et al. [81] propose a method extracting features and opinion based on POS tagging and generate summaries using association rules.

Zhang et al. [174] propose to extract features and opinion phrases based on shallow dependency parsing. Somprasertsri and Lalitrojwong [138] propose an approach for mining product features and opinions based on syntactic information and semantic information. Specifically, they apply dependency relations and ontological knowledge with probabilistic based model. Chen et al. [29] propose to adopt Condition Random Fields (CRF) model for the task of feature-level opinion mining and compare different approaches: rule-based, association rule mining based, L-HMMs based and CRF based. Instead of measuring sentiment orientations, Meng and Wang [108] propose to generate a summary of aspect-based using the original sentiment words for the features.

Aspect-based sentiment analysis is also used for the problem of products comparison. Sun et al. [140] introduce a products comparison system which incorporates subjective information mined from reviews and products technical details descriptions. In addition to mining reviews opinions, Bjørkelund et al. [14] illustrate on how to visualize sentiment analysis results using Google Maps that facilitates users in comparing hotels. These are orthogonal directions to our goal of producing a flowing text as a summary.

**Summarization.**

*Extractive Summarization (Selection).* Recently, there is a line of work that deals with the selection of a "good" set of reviews. In [87], the objective is to select a set of reviews that cover all attributes (for a given set of attributes). Lappas and Gunopulos [87] propose to formulate it as a search problem. Tsaparas et al. [143] refine the objective to also include both the positive and negative aspects of each attribute and formulate the review selection problem as a maximum coverage problem. Lappas et al. [86] further seek to preserve the underlying distribution of positive and negative comments in the reviews. They formalize the task as a combinatorial optimization problem and propose heuristic algorithms since the problem is NP-hard and also

NP-hard to approximate. Yu et al. [165] define the objective as to cover more diversified opinion clusters. Chen et al. [30] propose a framework for application review mining, namely AR-Miner, to select a set of most informative reviews via an intuitive visualization approach. Slightly different from selecting the top reviews for users, Chong et al [33] propose to discover unexpected information in micro-reviews that primarily to serve venue or product owners.

Our work on *review selection based on coverage of micro-reviews* (Chapter 3) is along the same lines, but is distinct in two ways. *First*, in terms of formulation, we seek to represent micro-reviews, rather than attributes. *Second*, in terms of approach, we introduce the efficiency requirement to the coverage formulation. To compare against approaches that focus on coverage but not efficiency, we compare against a max coverage algorithm as a baseline in Section 3.4. There also exists a variant of max coverage called budgeted max coverage [78] where the constraint is a total cost that cannot be exceeded. Our coverage formulation is different in how both constraints of cost and count apply.

Another related coverage formulation is the red-blue set cover problem [25, 125], whereby the input is a collection of sets, and each set may contain a mix of red and blue elements. The objective in *reviews selection* is to select a sub-collection of sets that covers all blue elements, but covers as few red elements as possible. If we interpret a red element as an irrelevant sentence in a review, the objective is then to get a set cover while minimizing the *number* of irrelevant sentences. In our work of *reviews selection* (Chapter 3), efficiency is used as a threshold constraint, rather than a minimization objective. Moreover, our efficiency definitions are not expressed in terms of a count, and instead are expressed in terms of fractions, which results in a significantly different formulation.

Related to the notion of finding a "good" set of reviews is the problem of determining the quality of each individual review [96]. Sites such as Amazon or Yelp allow users to rate each review by its helpfulness or usefulness. Most review ranking works rely on a supervised regression or classification approach, using the helpfulness votes as the target class [57, 80, 95]. One possible formulation to produce a set of reviews is to first rank all the reviews based on individual merits, and then selecting the top $K$. The weakness of this formulation is that it ignores the potential similarities among the top reviews. It may well be that the top few reviews all represent the same information. For comparison, we introduce a baseline called *Useful* in Section 3.4, which ranks reviews by its usefulness votes, and selects the top $K$.

*Abstractive Summarization (Synthesis).* Regarding to the work in Chapter 4, while we synthesize a "review" to create a summary of micro-reviews, there is a previous effort to create a synthetic review [139] to simulate a *fake review* or synthesize sentences for summarizing opinion sentences [54]. *Review ranking* seeks to rank reviews based on some notion of "quality" [96]. *Review selection* [87] 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 [143, 86].

While reviews have been studied extensively, relatively little attention has been paid to micro-reviews. One related work focuses on very short comments on eBay left by buyers about sellers [97], but the problem there was to extract aspects from the comments. There are also works [73, 83] on analyzing opinions in micro-blogging services such as Twitter. However, because Twitter is a general micro-blogging platform, these opinions are usually about more general concepts (e.g., brands, hashtags) rather than specific entities (e.g., products, restaurants). Unlike Foursquare tips, tweets are not attached to any entity, and it is difficult to separate "reviews" from other types of content in Twitter. Other works studying micro-reviews in Foursquare address different purposes. Chong et al. [33] identify unexpected micro-reviews for a single venue. Vasconcelos et al. [148] predict which micro-reviews would be popular (high number of likes).

Most of the previous works on Foursquare or other check-in services does not view them as a source of micro-reviews, but rather as location-based social networks (LBSN), 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 mining user profiles [149], movement patterns [119], privacy [128], or POI recommendation [168, 56, 31, 162].

In focusing on the reviewing feature of LBSNs, the work in Chapter 5 is related to the broader area of review mining. Previous works address various problems, such as ranking reviews by quality [96], selecting a small subset of representative reviews [87, 143, 86], or synthesizing a review [139]. The crucial distinction is their focus on reviews for a *single entity*, as opposed to our *multi-entity* scenario.

*Ranking Entities.* For the work in Chapter 6, we generate a different form of summary, i.e., ranking list. In ranking entities based on some entity information that may be derived from reviews, ours is related to several works. The key distinction is our use of the concept of reviewer expertise as a key feature in the model.

There are works focus on comparing entities based on comparative sentences [76, 140, 160]. Zhang et al. [173] used both comparative sentences and aspect-based sentiment analysis to construct a product graph and apply a PageRank-based algorithm on the graph for finding the ranking of products for a given feature. Zhang et al. [172] propose to rank products based on the importance of their reviews estimated by the reviews' credibility and posting date.

There also exist graph-based methods. Peleja et al. [126] propose a three-step procedure to estimate the reputation of linked entities by considering the connections between them and the sentiment words used to describe the entities. Chutmongkolporn et al. [36] apply HITS-based algorithm on a bipartite graph of reviewers and (entity, aspect) pairs to generate rankings. These are different from our problem setting because the entities in our problem are not linked and we do not require generating all the aspects beforehand.

Bashir et al. [11] adopt learning-to-rank to train a ranking model using genetic programming using features such as keywords matching and polarities. This requires supervised labels for training. Bashir [10] proposes a ranking model combining the query keyword match with the context subjectivity computed based on Senti-WordNet 3.0 [6].

Ganesan and Zhai [53] focus on the multi-aspect nature of queries. They propose two extensions to standard text retrieval models. Each entity is represented by a document derived from concatenating all of its reviews. The first extension deals with the respective ranking for each aspect. The second extension deals with expanding the query by adding opinion synonym words. Extending the idea from [53], Makris and Panagopoulos [98] propose to further use additional resources such as WordNet [111] and Bing Liu Opinion Lexicon [69]. Gourgaris et al. [61] explore user personalization. Our work in Chapter 6 is distinct in that we do not concatenate the reviews. Instead, we identify the quality of each review, either intrinsically or attributed to the reviewer. To investigate the effect of this distinction, in the experiments in Chapter 6 (Section 6.2), we compare to [53] in single-aspect scenario.

# Part I

# Single-Entity Summarization

# Chapter 3

# Review Selection Based on Coverage of Micro-Reviews

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

## 3.1 Overview

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

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

### 3.1.1 Matching Reviews and Tips

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

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

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

We want to match a sentence $s$ and a tip $t$ if they convey a similar meaning, and therefore one can be seen as covering the content of the other. We consider the following three criteria for making the matching decision. The first criterion considers the sentence and the tip as bags of words. If they share a substantial subset of textual content then we assume that they convey a similar meaning. In this case we say that they have high *syntactic similarity*. The second criterion considers the concept that is discussed. A sentence and a tip may discuss the same concept (e.g., a menu dish), but use different words (e.g., soup vs. broth). In this case we say that they have high *semantic similarity.* Finally, reviews as well as tips, express the opinions of their respective authors. Hence, in addition to sharing similar keywords and concepts, we would also like a matching sentence-tip pair to share the same sentiment (positive or negative). In this case we say that they have high *sentiment similarity.* In Section 3.3, we elaborate further on each of the above three types of similarity, and how they can be defined and measured. We also describe how to combine them into a single matching function $\mathcal{F}$.

### 3.1.2 Selection Coverage

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

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

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

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

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

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

### 3.1.3 Selection Efficiency

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

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

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

- **Minimum Efficiency:** In this case, the efficiency of a set of reviews $\mathcal{S}$ is defined as the minimum efficiency of any review in the set. Formally:

$$\mathrm{Eff}_{\min}(\mathcal{S}) = \min_{R \in \mathcal{S}} \mathrm{Eff}(R)$$

- **Average Efficiency:** In this case, the efficiency of a set $\mathcal{S}$ is defined as the average efficiency of the reviews in the set. Formally:

$$\mathrm{Eff}_{\mathrm{avg}}(\mathcal{S}) = \frac{\sum_{R \in \mathcal{S}} \mathrm{Eff}(R)}{|\mathcal{S}|}$$

- **Bag Efficiency:** In this case, we view a collection of reviews $\mathcal{S}$ as a single review $R_{\mathcal{S}}$ consisting of the union of the sentences of the reviews. We then define the efficiency of the collection as the efficiency of $R_{\mathcal{S}}$. Formally, we have $R_{\mathcal{S}} = \cup_{R \in \mathcal{S}} R$, and $\mathrm{Eff}_{\mathrm{bag}}(\mathcal{S}) = \mathrm{Eff}(R_{\mathcal{S}})$.

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

### 3.1.4 Problem Statement

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

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

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

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

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

We note that for the applications we consider (e.g., checking reviews on mobile devices), space and time resources are limited, so we need to select a small number of reviews to show to the user. We consider the EffSetCover formulation in order to quantify the minimal number of reviews necessary to cover all tips, and understand the tradeoff between coverage and efficiency. We compare the two alternative formulations experimentally in Section 3.4.3.

## 3.2 Algorithms

Ideally, we would like to solve the EffMaxCoverage problem optimally. In Section 3.2.1, we will show that it can be expressed as Integer Linear Programming (ILP) problem, for which there are known algorithms to derive the solutions. However, the ILP formulation, while optimal, may not be tractable for cases where the number of reviews is very large. This is because EffMaxCoverage is NP-hard. This result follows from the fact that in the special case where $\alpha = 0$ (no efficiency constraint) the EffMaxCoverage is the same as the MaxCoverage problem, which is known to be NP-hard. Therefore, we also need to look for approximation, or heuristic algorithms, which we discuss in Section 3.2.2.

### 3.2.1 Finding the Optimal Solution

Our problem definition differs depending on the choice of the efficiency function. In the case that we use the $\text{Eff}_{\text{min}}$ function requiring that $\text{Eff}_{\text{min}}(\mathcal{S}) \geq \alpha$ implies that each of the selected reviews must have individual efficiency of at least $\alpha$. Therefore, this is equivalent to the MaxCoverage problem, where the universe of available reviews is restricted to the subset of reviews that have efficiency at least $\alpha$. There is a known ILP formulation [150] for the MaxCoverage that we can use for obtaining an optimal solution.

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

We express the problem as a set of constraints:

$$\text{maximize} \sum_{j=1}^{m} y_j \tag{3.1}$$

$$\text{subject to} \sum_{i=1}^{n} x_i \leq K \tag{3.2}$$

$$\sum_{i:t_j \in \mathcal{T}_{R_i}} x_i \geq y_j \quad \forall t_j \in \mathcal{T} \tag{3.3}$$

$$x_i = \{0, 1\} \tag{3.4}$$

$$y_j = \{0, 1\} \tag{3.5}$$

Equation 3.1 is an objective function that maximizes the number of tips covered. Constraint 3.2 ensures that the number of selected reviews does not exceed $K$. Constraint 3.3 ensures that if $y_j = 1$ (i.e., tip $t_j$ is covered), then at least one review that covers $t_j$ must be selected. Constraints 3.4 and 3.5 require that every variable can take on the value of either 0 or 1.

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

- *Average Efficiency.* Constraint 3.6 below ensures that the average precision of the selected reviews will be at least equal to threshold $\alpha$.

$$\frac{\sum_{i=1}^{n} \frac{|R_i^r|}{|R_i|} x_i}{\sum_{i=1}^{n} x_i} \geq \alpha \Leftrightarrow \qquad \sum_{i=1}^{n} \left( \frac{|R_i^r|}{|R_i|} - \alpha \right) x_i \geq 0 \tag{3.6}$$

- *Bag Efficiency.* Constraint 3.7 ensures that the bag precision of the selected reviews will be greater than or equal to threshold $\alpha$.

$$\frac{\sum_{i=1}^{n} |R_i^r| x_i}{\sum_{i=1}^{n} |R_i| x_i} \geq \alpha \qquad \Leftrightarrow \sum_{i=1}^{n} \left( |R_i^r| - \alpha |R_i| \right) x_i \geq 0 \tag{3.7}$$

We use *ILP-EffMaxCover*, indexed by the efficiency function that we consider, to refer to the ILP formulations. With the constraints in place, the formulations can be solved by existing solvers. In our experiments, we employ the publicly available lp_solve library[1].

### 3.2.2 Greedy Selection

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

It is well known that due to the submodularity property of the coverage function, the greedy algorithm that always selects the review whose addition maximizes the coverage produces a solution with approximation ratio $(1 - \frac{1}{e})$ for the MaxCoverage problem, where $e$ is the base of the natural logarithm

[1]`http://lpsolve.sourceforge.net/5.5/`

[114]. That is, the coverage of the greedy algorithm is at least a $(1 - \frac{1}{e})$ fraction of the coverage of the optimal algorithm. Therefore, we obtain the following lemma.

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

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

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

**Algorithm 1** *Greedy-EffMaxCover* algorithm.

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

The algorithm, shown in Algorithm 1, proceeds in iterations each time adding one review to the collection $\mathcal{S}$. At each iteration, for each review $R$ we compute two quantities. The first is the *gain* $\text{gain}(R)$, which is the increase in coverage that we obtain by adding this review to the existing collection $\mathcal{S}$. The second quantity is the cost $\text{cost}(R)$ of the review $R$, which is proportional to the *inefficiency* $1 - \text{Eff}(R)$ of the review, that is, the fraction of sentences of $R$ that are not matched to any tip. We select the review $R^*$ that has the highest gain-to-cost ratio, and guarantees that the efficiency of the resulting collection is at least $\alpha$, where $\alpha$ is a parameter provided in the input. The intuition is that reviews with high gain-to-cost ratio cover many additional tips, while introducing little irrelevant content, and thus they should be added to the collection.

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

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

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

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

### 3.2.3 The EffSetCover Problem

Similar to the EffMaxCoverage problem, the EffSetCover is also NP-hard, since the special case where $\alpha = 0$ is equivalent to the SetCover problem which is known to be NP-hard [150]. For the SetCover problem we can use the ILP formulation below to obtain an optimal solution. We can add the efficiency constraints 3.6 and 3.7 to obtain a solution to the EffSetCover problem.

$$\text{minimize} \sum_{i=1}^{n} x_i \tag{3.8}$$

$$\text{subject to} \sum_{i:t_j \in \mathcal{T}_{R_i}}^{n} x_i \geq 1 \qquad \forall t_j \in \mathcal{T} \tag{3.9}$$

$$x_i = \{0, 1\} \tag{3.10}$$

## 3.3 Matching Reviews and Tips

As mentioned in Section 3.1, for matching reviews and tips, we consider 3 types of similarity. In this section, we define each type in detail, as well as how they can be measured. We also describe how to combine them into the matching function $\mathcal{F}$.

**Syntactic Similarity (SynSim).** A review sentence and a tip are syntactically similar if they share important keywords. A well-established model for keyword similarity is the vector space model [102]. Each review sentence $s$, and each tip $t$, are associated with vectors $\mathbf{s}$ and $\mathbf{t}$ respectively. The dimensionality of the vectors is the size of the vocabulary. Each vector entry signifies the importance of the corresponding word. The degree of similarity between the sentence and the tip is then measured as the cosine similarity [102]. Therefore, we have:

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

To compute the importance weights for the words we form a corpus of documents, where each document represents an entity (e.g., restaurant) and it consists of all the tips about this entity. We use the standard *tf-idf* [102] scheme for determining the importance of a word.

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

While there are several potential topic models, here we describe an approach based on the well-known Latent Dirichlet Allocation (LDA) [16]. For illustration, in Table 3.1, we show an example of topics discovered from the Foursquare tips of the restaurant *Shake Shack*[2], located at Madison Square Park in New York. Due to space limitation, we show five out of 20 topics learned from the restaurant's tips. The topics reflect: (1) the main menu of burgers, shakes, and cheese fries; (2) the waiting time; (3) frozen custard; (4) the mushroom burger vegetarian option; and (5) the location at Madison Square Park. This small example serves to demonstrate that the topics do reflect the pertinent concepts in each restaurant.

LDA associates each tip $t$ with a probability distribution $\theta_t$ over the topics, which captures which topics are most important for a tip. Given the topics, and the corresponding language model for each topic as it is learnt from the tips, we can estimate the topic distribution $\theta_s$ for each review sentence $s$,

[2]`https://foursquare.com/v/shake-shack/40e74880f964a520150a1fe3`

Table 3.1: Example Topics for *Shake Shack, Madison Square Park*

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

which captures how well a sentence $s$ reflects the topics being discussed in the corpus of tips. To measure the semantic similarity between a review sentence and a tip, we measure the similarity of the topic distributions $\theta_s$ and $\theta_t$. A commonly used distance measure between two probability distributions is the Jensen-Shannon Divergence (JSD) [102]. Intuitively, a sentence and a tip are semantically similar if their topic distributions can describe each other well. The lower the divergence, the greater is the similarity. Therefore, we have:

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

**Sentiment Similarity (SentSim).** A matching pair of review sentence and tip should also represent the same sentiment. Sentiment extraction from text is an active area of research [122]. Here, we cast the problem as a classification problem, where the goal is to predict the sentiment (positive or negative) of a sentence or a tip. We thus have two classes $c^+$ and $c^-$. We use a maximum entropy classifier (MEM) [100], which has been demonstrated to work well for sentiment classification in text [122], using N-gram features (both letter N-grams and word N-grams). To illustrate these features, in Table 3.2, we show several features found to be important (high feature weight) for each class. Letter N-grams are prefixed by a dash. For the positive class, we have words such as "so" and "best", as well as the superlative suffix (letter N-gram) "-est". For the negative class, we have negations, such as "not", or words with negative connotation, such as "over" or "but".

Given a document $d$ (a sentence or a tip), the MEM classifier outputs con-

Table 3.2: Example Important Features for Sentiment Classification

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

ditional probabilities $P(c^+|d)$ and $P(c^-|d)$ for the positive and negative classes, where $P(c^+|d) + P(c^-|d) = 1$. Given the classifier output for a document $d$, we transform the probability $P(c^+|d) \in [0, 1]$ into polarity$(d) = 2P(c^+|d) - 1$, in the range of -1 (extremely negative) to 1 (extremely positive). For $P(c^+|d)$ close to 1/2, the polarity is close to zero, which agrees with our intuition that in these cases the document has neutral polarity. We define the sentiment similarity between a sentence $s$ and a tip $t$ as the product of their polarities: it approaches 1 when the sentence and the tip's polarities are similar; it approaches -1 when their polarities are opposite; it approaches 0 when the tip or the sentence polarity is neutral. Therefore, we have:

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

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

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

## 3.4 Experiments

The objective of the experiments is to showcase the effectiveness of the proposed approach in finding a set of reviews that cover as many tips as possible, in an efficient manner. First, we will describe the real-life dataset used in the experiment. This is followed by an evaluation of the matching process described in Section 3.3. We then investigate how the coverage algorithms proposed in Section 3.2 behave under different parameter settings, as well as how they compare against the baselines.

### 3.4.1 Dataset

The experiments require data coming from two different sources (reviews and micro-reviews), concerning the same set of entities. We pick the domain of restaurants, because it is a popular domain where there are active platforms for reviews as well as for micro-reviews. For reviews, we crawl Yelp.com to obtain the reviews of the top 110 restaurants in New York City with the highest number of reviews as of March 2012. For micro-reviews, we crawl the popular check-in site Foursquare.com to obtain the tips of the same 110 restaurants. However, some of the restaurants in Foursquare.com have too few tips, which may not adequately reflect the restaurant's information. Therefore, we retain only the 102 restaurants with at least 50 tips each. For these 102 restaurants, we have a total of 96,612 reviews, with a minimum of 584, and a maximum of 3460 per restaurant. We also have a total of 14,740 tips, with a minimum of 51, and a maximum of 498 per restaurant. Note that we get the *full* set of reviews and tips of each restaurant at the time of extraction, and that these are the realistic sizes of the real-world data. It is also important to note that every restaurant is a distinct instance of the coverage problem.

### 3.4.2 Matching

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

To build the matching classifier, we generate the three real-valued features described in Section 3.3. For semantic similarity, we train LDA [16] topic models using the MALLET toolbox [104]. Because topic modeling is probabilistic, we average the semantic similarity over ten runs. To determine the sentiment polarity of each sentence and tip, we train a sentiment classifier using the Stanford Classifier toolkit [100] with textual features (word and letter n-grams).

To train the matching classifier, we sample 20 entities, and for each entity we sample 50 sentence-tip pairs sharing at least one common word. We assume no match otherwise. For these 1,000 pairs, we get three judges to label whether the pairs match in meaning, and take the majority label as the ground truth. Finally, we use the real-valued features and the majority labels to train the matching classifier using the MEM classifier from [100]. Based on the feature weights learned by the classifier, we find that among the three features, semantic similarity is the most important, followed by syntactic, and lastly sentiment.

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

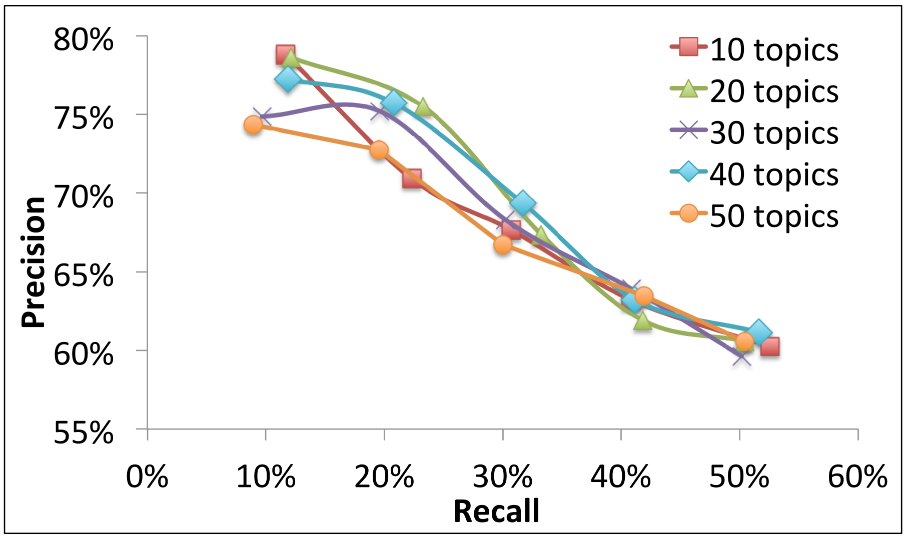


Figure 3.1: Matching: Precision-Recall Curve

Table 3.3: Performance of Matching Classifier

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

the underlying tips.

**Number of topics.** We study the performance of matching classifier as we vary the number of topics used for the semantic similarity. In Figure 3.1, we plot the precision-recall curve for $\eta = 0.65$ (discussed below). It appears that the effect of the number of topics on precision and recall is not significant. The performance for 20–40 topics is better than for 10 (which may underfit), or for 50 (which may overfit). The results for 20 topics are slightly better, especially in terms of precision, which is our main concern. Subsequently, we show the results for 20 topics.

**Threshold $\eta$.** We experiment with different values for the threshold $\eta$ on the probability of matching $P(s, t)$. Table 3.3 shows the precision and recall of the matching classifier at different values of $\eta$. If we were to skip the matching classification, and simply take all the pairs with at least one common word as matching, we get a precision of only 43%, which means more than half of all matching pairs would be incorrect. As we increase the threshold $\eta$, the precision improves significantly. If we would like at least three-quarters of matching pairs to be correct, we need to put the threshold at 0.65 of higher. At this threshold, the recall is low at 23%, but this can be compensated by the fact that a tip may be covered by many different sentences.

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

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

To get an intuitive sense of the matching quality, we show some examples of matching pairs for the burger joint *Shake Shack* in Table 3.4. The first pair discuss how good the fries are. The second pair discuss the long waiting times, while the third pair discuss the mushroom burger. These examples showcase how the features, i.e., syntactic, semantic, and sentiment similarity, help to identify relevant matching pairs.

### 3.4.3 Coverage and Efficiency

Given the matching between review sentences and tips, we now investigate the effectiveness of our algorithms in terms of coverage and efficiency. First, we will compare the EffMaxCoverage formulation with the EffSetCover formulation. Then, we will compare the proposed greedy algorithm against the optimal solution, and then against the baselines. Finally, we will show a case study to provide an intuitive sense of the kind of results that our algorithm produces compared to those of the baselines.

Table 3.4: Example Matching Pairs for *Shake Shack, Madison Square Park*

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

## EffMaxCoverage vs. EffSetCover

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

The first issue that we need to address is that an optimal solution does not always exist. Table 3.5 shows the percentage of the 102 entities, for which *ILP-EffMaxCover* can discover an optimal solution for the three efficiency functions (average, bag, and minimum). We focus on $\alpha \geq 0.5$, which is the more interesting range, since, as we will show in the experiment results below, the baseline *MaxCover* (maximizing coverage without efficiency constraint) has an efficiency of 0.43. The table shows that at efficiency threshold $\alpha = 0.5$, there exists an optimal solution for 90% of the entities. Of the ten entities for which an optimal solution cannot be obtained, nine are due to the two-hour cut-off that we impose on the lp_solve package in order to avoid infinite running time for large problem sizes. For the last entity, there is no optimal

Table 3.5: ILP-EffMaxCover: Entities with Optimal Solutions

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

Table 3.6: ILP-EffSetCover: Entities with Optimal Solutions

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

solution that can satisfy the constraints. As expected, as we increase $\alpha$, the percentage of optimal solutions decreases monotonically, as there are fewer and fewer entities that can satisfy the increasingly stringent efficiency constraint.

Table 3.6 shows the percentage of entities, for which *ILP-EffSetCover* can discover an optimal solution. The ILP solver terminates under two hours in all cases, therefore the cases where there is no optimal solution are due to the non-existence of such a solution. Note that the percentages are much lower than those for *ILP-EffMaxCover*, especially for larger $\alpha$. This implies that requiring perfect coverage is too stringent, since in most cases such a solution cannot be found.

Table 3.7: ILP-EffSetCover: Optimal Number of Reviews

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

Table 3.7 shows the average number of reviews produced by the *ILP-EffSetCover* algorithm for the entities for which an optimal solution exists. From the table it is clear that in order to cover all the tips, we require a substantial number of reviews (in the order of twenties and thirties). This number becomes higher for larger values of $\alpha$ indicating that we cannot reduce the

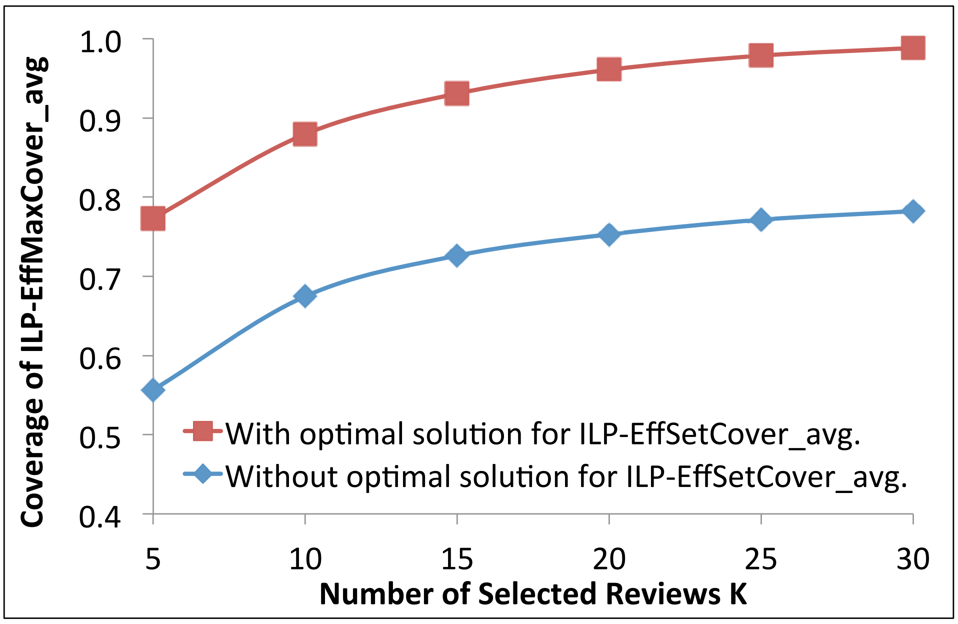


Figure 3.2: Coverage of *ILP-EffMaxCover*$_{\text{avg}}$ ($\alpha = 0.5$)

amount of content by imposing stricter efficiency constraints. For the application scenarios we consider, where a user wants to quickly make a decision, by reading little content on a screen with limited real estate, this number of reviews is too high.

We now investigate the level of coverage achieved by *ILP-EffMaxCover* as a fraction of the full coverage. As a representative, we use *ILP-EffMaxCover*$_{\text{avg}}$, but similar conclusions can be drawn for other efficiency functions. Figure 3.2 plots the coverage of *ILP-EffMaxCover*$_{\text{avg}}$ at $\alpha = 0.5$, for different values of $K$. We divide the entities into two groups. The first group (corresponding to the top red line in Figure 3.2) consists of 63 entities for which a perfect coverage of all tips exists. It shows that as $K$ increases the solution for *ILP-EffMaxCover*$_{\text{avg}}$ eventually converges to the perfect coverage (1.0) around $K = 30$. Interestingly, even for smaller values of $K$, which are of interest to our applications, the coverage is still very high, e.g., 0.8 coverage at $K = 5$. The second group (corresponding to the bottom blue line in Figure 3.2) consists of 29 entities for which no perfect coverage exists. Even in such cases, the *ILP-EffMaxCover* is able to obtain a satisfactory solution (coverage between 0.6 and 0.7).

In conclusion, we find that the EffMaxCoverage formulation is the appropriate formulation for the problem we consider. Obtaining a set of reviews with full coverage is often not feasible, and when feasible it results in a large number of reviews. Moreover, EffMaxCoverage, even for small values of $K$, produces a set of reviews with significant coverage of the tips.

### EffMaxCoverage: ILP vs. Greedy

While ILP produces an optimal solution, it suffers from high running time. On the other hand, the greedy algorithm is much more efficient, but it produces an approximate solution. We will now measure experimentally how closely the greedy coverage can approximate the optimal solution by ILP. In particular, we will measure two quantities. The first quantity, *coverage approximation ratio*, takes the ratio of the coverage by Greedy-EffMaxCover to that by *ILP-EffMaxCover*. The second quantity, *efficiency approximation ratio*, is a similar measurement on the efficiency. Both measurements are averaged across the 92 entities (i.e., the 90% of entities with optimal results at $\alpha = 0.5$ in Table 3.5). These ratios will reveal how closely the greedy algorithm can approximate the optimal solutions.

Table 3.8: Greedy vs. ILP: Approximation Ratios ($K = 5$)

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

Table 3.8 shows that both the coverage and efficiency approximation ratios are all very close to 1 for any $\alpha$. This means that in practical terms, the greedy version of *EffMaxCover* is virtually identical to the ILP version.

Table 3.9 compares the time taken by the greedy and ILP versions (on a regular PC with Intel Core i5 3.20GHz CPU and 4GB RAM). Greedy is

Table 3.9: Greedy vs. ILP: Running Time in Milliseconds ($K = 5$)

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

easily two orders of magnitude faster than ILP. Even this speedup is a severe underestimation, since it does not account for the cases where ILP cannot complete its run within the two-hour cut-off time. Such cases involve larger problem sizes. The entities that ILP can solve within the cut-off time have on average 606 reviews and 107 tips. The unsolvable entities have on average 1505 reviews and 246 tips. The greedy algorithm can still solve the latter very efficiently (within 25 ms).

In the next discussions, we will focus on the understanding and evaluation of the greedy algorithm.

## Greedy-EffMaxCover: Parameter Analysis

There are two ways in which *Greedy-EffMaxCover* controls the efficiency of the selected set of reviews. The first is by the threshold $\alpha$, which guarantees the efficiency of the set is at least $\alpha$. The second is by the parameter $\beta$ which controls the sensitivity of the selection process to the efficiency of the next review to be added to the set. In the following, we would first study the effect of $\alpha$, and then the effect of $\beta$, before selecting an appropriate setting for $\alpha$ and $\beta$ jointly.

**EffMaxCover: Varying $\alpha$.** To isolate the effect of $\alpha$, we fix $\beta = 0$, making the cost a constant, independent of the efficiency. We vary $\alpha$ from 0.5 to 1.0 for $K = 5$, and show the coverage and efficiency in Table 3.10. Table 3.10 shows that as $\alpha$ increases, as expected, efficiency monotonically increases. However, coverage monotonically decreases, since the constraint

Table 3.10: Greedy-EffMaxCover ($K = 5, \beta = 0$)

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

on $\alpha$ disqualifies some of the higher-coverage, but lower-efficiency reviews. Furthermore, as $\alpha$ increases, the percentage of entities for which a result that satisfies all the constraints can be found, also decreases. For large values of $\alpha$, i.e., $\alpha > 0.8$, the percentage of entities with a solution drops below 85%, which is too limiting. Therefore, subsequently, we will search for a suitable setting in the range $0.5 \leq \alpha \leq 0.8$.

Among the different ways of aggregating efficiency for *Greedy-EffMaxCover*, we observe $\text{Eff}_{\text{avg}}$ and $\text{Eff}_{\text{bag}}$ behave similarly, and slightly differently than $\text{Eff}_{\text{min}}$. The $\text{Eff}_{\text{min}}$ tends to have higher efficiency but lower coverage. This is due to the fact that every selected review has to meet the efficiency threshold, reducing the set of available candidate reviews. In contrast, the other two algorithms consider the efficiency of the set as a whole, and they may select reviews with high coverage that have efficiency below $\alpha$, if the reviews already in the set have high efficiency. If there is a strict requirement on efficiency, $\text{Eff}_{\text{min}}$ is probably the better choice. Otherwise, $\text{Eff}_{\text{avg}}$ and $\text{Eff}_{\text{bag}}$ are slightly better with their higher coverage. We will use $\text{Eff}_{\text{avg}}$ as a representative in the subsequent analysis, due to its slightly higher coverage.

**EffMaxCover: Varying $\beta$.** We now study the effect of $\beta$ on the performance of the algorithm, for different values of $\alpha$. Figure 3.3 shows how the coverage and efficiency change as $\beta$ increases from 0 to 1 for *Greedy-EffMaxCover* with $\text{Eff}_{\text{avg}}$. The curves for the other efficiency variants $\text{Eff}_{\text{bag}}$ and $\text{Eff}_{\text{min}}$ are similar and not shown here due to space limitation. We plot the curves for the

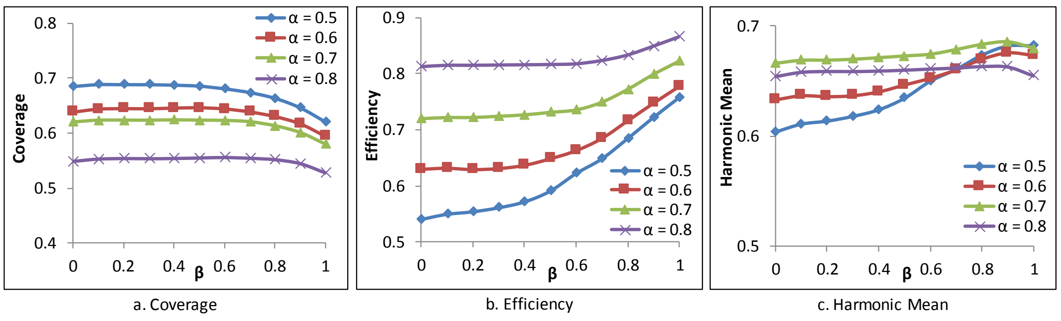


Figure 3.3: Comparison of Coverage & Efficiency for varying $\beta$ for $\alpha \in [0.5, 0.8]$ for *Greedy-EffMaxCover* with $\mathrm{Eff}_{\mathrm{avg}}$

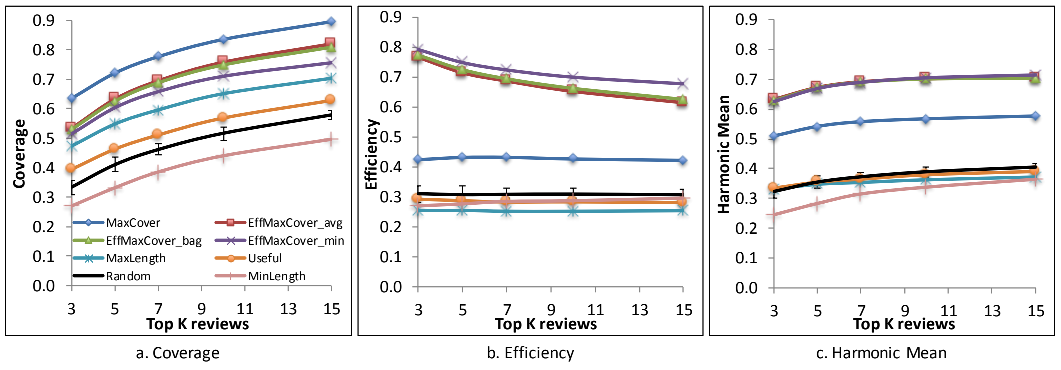


Figure 3.4: Comparison of Coverage & Efficiency for varying $K$ for $\alpha = 0.5$, $\beta = 0.9$

values of $\alpha$ between 0.5 to 0.8, for which there are at least 85% of entities that satisfy the constraints (see the earlier discussion on Table 3.10).

At $\beta = 0$, the cost is a constant, and we rely entirely on $\alpha$ to maintain efficiency. As we increase $\beta$, the greedy selection of reviews will increasingly be sensitive to the cost (loss in efficiency). Figure 3.3 shows that for all values of $\alpha$, as $\beta$ increases, the efficiency increases while the coverage decreases. Interestingly, the gain in efficiency outpaces the loss in coverage. For example, for $\alpha = 0.5$, from $\beta = 0$ to $\beta = 1$, efficiency increases from 0.54 to 0.76 (efficiency gain of 0.22), while the coverage reduces from 0.68 to 0.62 (coverage loss of 0.06). This shows $\beta$ is an effective way to gain efficiency with minimal loss in coverage.

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

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

Figure 3.3(c) plots the harmonic mean, which consistently reaches the peak at around $\beta = 0.9$ for all values of $\alpha$. For $\alpha = 0.5$ and $\beta = 0.9$, we can guarantee that the efficiency is at least 50%, and the harmonic mean is as high as possible. Subsequently, we will use this setting for *Greedy-EffMaxCover*.

### EffMaxCover vs. Baselines

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

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

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

**Varying $K$.** Figure 3.4(a) shows how coverage varies with $K$ for various methods. As expected, *MaxCover* has the highest coverage, followed closely by the *EffMaxCover* variants. *MaxLength* and *Useful* also do better than

Table 3.11: User Study Comparing EffMaxCover and MaxCover

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

*Random*, but worse than *EffMaxCover*. *MinLength* has the lowest coverage, as it has very few sentences to capture the tips.

Figure 3.4(b) shows that the efficiency of *EffMaxCover* algorithms is by far superior to all the baselines. This underlines the effectiveness of *EffMaxCover* in finding efficient reviews. The efficiency tends to decrease slightly with increasing $K$, which is expected as it gets increasingly more difficult to find high-coverage and high-efficiency reviews after each selection. Interestingly, the efficiency of *MaxLength* and *Useful* fall below that of *Random*, which could be due to the length of the reviews, resulting in having many sentences that may not represent any tip. *MinLength* is more efficient than *MaxLength*, but is also worse than *Random*. This suggests that being short alone is not sufficient if it does not also capture the tips well.

To emphasize the efficacy of *EffMaxCover* at achieving both coverage and efficiency, we plot the harmonic mean in Figure 3.4(c). It shows how the three *EffMaxCover* variants outperform the rest significantly, followed by *MaxCover*. *MaxLength* and *Useful* are no better than *Random*, whereas *MinLength* is the worst.

**Qualitative Analysis.** We conduct a qualitative analysis involving three judges who are not related to this paper. To each judge, we show the top 3 reviews selected by an algorithm for a sample of 20 restaurants, and ask the judge to choose which aspects are mentioned in the reviews from a manually hand-picked list of aspects. Because the objective is to investigate the trade-off between coverage and efficiency, we focus the comparison on two methods: *EffMaxCover*$_{\text{avg}}$ as a representative of the *EffMaxCover* variants, and *MaxCover*, as the closest competitor.

Table 3.11 shows that on average, the judges identify 5.1 aspects for *MaxCover*, and 3.7 aspects for *EffMaxCover*$_{\text{avg}}$. This lower coverage of aspects is expected, and consistent with the previous experiments. On the other hand, the reviews selected by *EffMaxCover*$_{\text{avg}}$ are much more compact, with an average of 26.8 sentences total in three reviews, as compared to the lengthy 121 sentences by *MaxCover*. This suggests a gain in efficiency. If we look at the density of information covered (the ratio of aspects covered per sentence), the third column of Table 3.11 shows that *EffMaxCover*$_{\text{avg}}$ has much higher density of 0.14 aspects per sentence, than 0.04 by *MaxCover*.

### 3.4.4 Case Study

To illustrate the different types of reviews selected by the various criteria, as a case study, we show an example of the top review selected by each algorithm for the venue *Shake Shack* in Figure 3.5, Figure 3.6 and Figure 3.7. This is a burger joint located in Madison Square Park in New York.

The top review selected by the greedy version of *EffMaxCover* (Figure 3.5a) (all three efficiency variants selected the same) is compact and informative, describing the main attributes of the place: the long wait at peak hours, the location at Madison Square Park, the dishes (burgers, fries), as well as the affordable prices. If one is to select only one review to show on a mobile device, this review will best make use of the small screen size and the short user attention span in conveying a lot of information with a small footprint.

*MaxCover*'s top review (Figure 3.5b) also covers these attributes, but with a significantly longer review. Parts of the review are not to the point. For instance, the mentions of "hearing voices inside my head" and "Disney princess" do not concern the restaurant directly. The same can be said for *MaxLength* (Figure 3.6a), which discusses the same aspects, but using many more words. These very long reviews may not display as well on a mobile device, and may require inconvenient pagination.

The other two reviews, while also compact, are not informative. *Useful*'s top review (Figure 3.7a) is written in a sarcastic tone, including irrelevant mentions (e.g., “I hate when the Mets lose.”). *MinLength*'s top review (Figure 3.7b) is too short and only covers the generics, without getting into helpful details such as dishes or waiting time.

## 3.5 Discussion

In this chapter, we present our research on summarizing micro-reviews, in which we introduce the use of micro-reviews for finding an efficient set of reviews, which is novel in the objective of micro-review coverage, as well as in the efficiency constraint. We describe an optimal algorithm based on Integer Linear Programming (ILP). Because the problem is NP-hard, we design a greedy algorithm, which is empirically shown to be virtually identical to the optimal solutions in coverage and efficiency, and yet is much faster computationally. Evaluation over a corpus of restaurants' reviews and micro-reviews shows that our approach outperforms the baselines in discovering review sets consisting of compact, yet informative reviews.

While in Union Square, I decided to deviate from my usual plan of lunch at Wildwood BBQ, and chose to venture off to the Shake Shack. I tried to go last saturday, but the line was at least a 45 minute wait so I decided to go at a non-peak hour so the line wouldn't be as long. The Shake Shack is in a gorgeous location, right in the heart of Madison Square Park in Gramercy. The Shake Shack...its litterally a shack, don't expect Tao. But the burgers are one of the best in the city and are most deffinetly on par with those of the Burger Joint. While J.G. Melon has the best fries between the Burger Joint, and the Shake Shack, the Shake Shack's fries weren't too shabby. The best part, lunch cost me $8.75 (one burger, an order of fries, and a coke). Not too bad for an upscale burger. What really makes it though is the location, eating one of the best burgers in the city while gazing at the beauty of Madison Square Park, is just phenominal. The Shake Shack is a great place for lunch, it won't cost you an arm and a leg, and you'll eat well.

(a) EffMaxCover

Psh, who travels to Manhattan for a burger at 11 ? Alright fine, I did exactly just that. Now, for a few years, I've been hearing murmurs of Shake Shack cranking out one of the best burgers in town. I told my doctor I was hearing voices inside my head, but he told me to just go to Shake Shack. I guess he's a fan too. So went I did. Accompanied by the abundance of squirrels, birds, and pigeons (I can't group them in the same category), I waited in line, feeling like a Disney princess. Since nobody else is crazy enough to get burgers 11 in the morning in the freezing cold, the line was pretty short. (I wasn't as crazy as the guy that downed two shakes in the time I was there though) "A single cheeseburger and a Shack burger pleas–oh and fries, cheese fries...please !" A wand-like device was handed to us. Shrugging, we picked our seats. Not every table comes with a heat lamp, but they do nothing anyways. Except for burning the top of heads. We waited a short while for our food to arrive– approximately ten minutes. A few minutes later, I found myself looking at a hot-dog-at-a-baseball-game-esque box. Our treasures laid within. Food: 10/10 Though the burgers were smaller than I had expected, I appreciated the balance of each ingredient. The ratios were perfect. The bun wasn't too small– there was just enough meat peeking out, so you don't feel jipped for paying more than 3 bucks for a single burger. Both burgers weren't pink in the middle–I had expected med rare. The seasoning was fantastic, enough so you don't feel like you're chewing on dead cow. Single cheeseburger- Interestingly enough, despite the fact that Shake Shack is a chain, not every burger is created equally. Meaning, my boyfriend's Shake burger was a tiddle bit rarer than mine. Mine was still juicy, but lost a bit of its tenderness. The lettuce and tomato was extremely fresh– I loved how they give you the lettuce by the leaf, and like the crazy shredded massacres that they plop down in other establishments. Cheese ? Deliciously melty. It makes me feel okay about being American. A solid burger, a solid burger. Shack burger-Deeeeeeeeeeeeelicious. Oh my goodness, I saw stars. The sauce tasted a bit naughty, in the fattening but tasty sort of way. Props for a nice thickness in the patties– nothing like those wimpy thin, thin patties. This is the stuff. Cheese fries- I love fries. Therefore, when you give me fries, smothered in fake, plastic-like cheese, I will love them even more. You can't really go wrong, people. Delightfully crunchy, pretty good for crinkle cut fries (they're usually soft and floppy). Though it's not the cheapest option, it sure tastes a lot better than any other fast food establishment I've been in. I'm more than happy to shell out the bucks, knowing that I'll be guaranteed these delicious things. Wowza. I'm coming for you Shake Shack. Hide 'yo burgers.

(b) MaxCover

Figure 3.5: Top Review for *Shake Shack, Madison Square Park*

Very solid fast food burger ***Short Review Came here for dinner Oct'11. Very good fast food burger. I feel it tops 'Five Guys' and 'In N Out'. What makes it stand out are great beef patties & well thought out use of cheese & sauce. Beef patties are relatively thick & juicy. Burger doesn't fall apart and isn't too messy. Price of the double patty 'Shack Burger' at $7 isn't cheap, but I feel the value is reasonable for what you get. Fries are decent but not exceptional. My coffee & vanilla shake was awesome. Price isn't cheap, and value is probably about average. Cost of a double patty 'Shack Burger', regular fries, and a 'Fair Shake' set me back around $16 after taxes, & before tip. Waiting for the burger here can take some time. It took me approximately 30 mins from waiting in line to getting my food. Ambiance is outdoors seating. Its fine as long as the weather's tolerable. ***Detailed Review I really enjoyed Shake Shack. I feel it takes on some of the qualities of better fast food burgers, improves on them, and adds a bit of it's own charm. What's also great is their commitment to excellent shakes, offering beer, and some promising desserts (which unfortunately I didn't get the chance to try). This is one of my favorite fast food burgers right now, certainly much better than both 5 Guys and 'In N Out'. It's also a lot more enjoyable than many of the gourmet burgers, often which are more expensive than Shake Shack. On to the food; 'Shack Burger' American cheese, lettuce, tomato, & shack sauce. (Double patty, approx $7) (Dish Rating, 75%) At first glance, this burger resembles somewhat of a hybrid between In N Out & 5 guys. Like 5 guys, you get two larger, thicker (relative to a fast food burger) beef patties. The patties are thicker & juicier than 5 guys. The meat at 5 guys is cooked throughout, but Shake Shack does it closer to medium. Inside of patty is a little pink. Meat is quite tasty with a distinct char. Size of patty isn't small either, maybe around 4oz each, or 8oz total. When biting into the burger, the beef patty is the central focus. The grind of the beef is fairly coarse, so the texture is more chewy although not difficult to chew. Biting into the juicy, chewy meat is delicious affair. Like 'In N Out', there's also a heavier presence of cheese and sauce that gives the burger some extra flavor (if that's your preference). However, it excels further in that the sauce and cheese aren't so over-powering. Unlike In N Out, the sauce isn't quite as thick or sweet. The cheese isn't quite as strong or gooey. There is a great balance between cheese, sauce, and meat. Mostly with the sauce and cheese complementing the meat in a more balanced manner. Other than the cheese and sauce, it's Shake Shack's burger doesn't seem to focus too much on other condiments. There's a slice of tomato and some regular lettuce. I feel that's totally fine, as there's already enough good things going on, that there need not be much more complexity Bun used, I felt did it's job but wasn't in itself taking up much attention. It held the burger in place, wasn't heavy, and wasn't too messy to eat. It's difficult to pinpoint an inherent weakness for Shake Shack. As a fast-food burger, it's close to as good as it gets. Only 'Little Big Burger' in Portland, OR has a slight advantage, and that's because the meat quality is better, the beef patty is thicker & even juicier. They also allow you to cook the beef patty 'rare' or 'medium rare'. Nonetheless, Shake Shack isn't far behind. For a somewhat larger burger franchise, I feel it's hard to top. Fries here are probably closer to decent. They're a crinkle cut. Thankfully low on grease, starch, and seasoning. They're served nice and crispy. There's no inherent weakness, but nothing outstanding about them either. On this count, I feel Shake Shack also falls behind somewhat. 'Fair Shake' Vanilla shake spun 100% certified Arabica fairly traded coffee ($5.50) (Dish Rating, 76%) As far as shakes go, I only had the chance to try their vanilla coffee shake. Here I felt Shake Shack were truly exceptional. The flavor of the vanilla and coffee quite intense, and the consistency of the shake to be rich but not too heavy. It's difficult for me to articulate beyond that what makes a good shake, but I feel that it was exceptional. There are few places that can make a better shake. Some people might feel that $7 for a Shake Shack double patty burger is too expensive. At this price point, it's beginning to encroach on the cost of many gourmet burgers. Nonetheless, it's also a whole lot better than several gourmet burgers. It's probably not a great deal, but the quality of the burger justifies it's cost in my opinion. If you're looking for value based on quantity of food, or the lowest price possible, you might be better off looking elsewhere.

(a) MaxLength

Figure 3.6: Top Review for *Shake Shack, Madison Square Park* (cont.)

I hate Shake Shack. I hate the wait. I hate the hype. I hate it's proximity to my office. I hate my absolute inability to resist whenever anyone suggests it. I hate when I get a craving, and no one else has it. Because I hate standing in line, friendless. I hate the Outback-esque vibrating wand they hand you. I hate mosquitoes. I hate humidity. I hate you. I hate non-potato bread. Why does it exist? I hate that the burgers taste better at CitiField. I hate when the Mets lose. I hate that they lose a lot. I hate my gut, which Shake Shack is at least somewhat responsible for. I hate that a single cheeseburger is too small, and a double necessary. I hate that it would probably take me less time to trek up to the UWS location and back than it would to wait here in Madison Square Park. But I love life when I'm eating delicious burgers and cheese fries in the middle of the park. You just have to ask yourself if the wait is worth the 3 minute face-stuffing. And you know what? It often is.

(a) USEFUL

Sweet corn frozen custard? Mmmm. Mint honeydew? MMMM. All hail the Shake Shack and summer!

(b) MINLENGTH

Figure 3.7: Top Review for *Shake Shack, Madison Square Park* (cont.)

# Chapter 4

# Review Synthesis as Concise Representation of Micro-Reviews

In this chapter, we consider the following summarization task. 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.

We formulate the problem based on the Minimum Description Length (MDL) framework. MDL, which was introduced by Rissanen [131], is a well-established principle for model selection [63]. 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, [27] 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 4.1.3). There are also works on metric UFLP [135, 72, 88], but the metric assumption does not apply to our case.

## 4.1 Problem Formulation

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

### 4.1.1 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.

### 4.1.2 Problem Definition

To identify this "optimal" summary, we turn to Minimum Description Length (MDL) [131], 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 [63]. 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 [40] 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.

$$\begin{aligned} model(S) &= \log n + (k-1)\log n + \sum_{r \in S} \text{bits}_W(r) \\ &= \sum_{r \in S} (\log n + \text{bits}_W(r)) \end{aligned} \tag{4.1}$$

**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 [101]. 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 [170]. The smoothed generation probability of a word is as follows.

$$P_r(w) = \frac{\text{tf}_{r,w} + \alpha}{|r| + \alpha|W|} \tag{4.2}$$

In this equation, $\text{tf}_{r,w}$ is the number of occurrences of the word $w$ in the snippet $r$, while $\alpha$ is the smoothing coefficient. Larger $\alpha$ 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.

$$\mathit{data}(\mathcal{T}|\mathcal{S}) = \sum_{r \in S} \sum_{t \in T_r} \text{bits}_r(t) \tag{4.3}$$

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 4.4.

$$\begin{aligned}\mathit{cost}(\mathcal{T},\mathcal{S}) &= \mathit{model}(\mathcal{S}) + \mathit{data}(\mathcal{T}|\mathcal{S}) \\ &= \sum_{r \in S} \left[ (\log n + \text{bits}_W(r)) + \sum_{t \in T_r} \text{bits}_r(t) \right] \end{aligned} \tag{4.4}$$

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 3** (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 $\mathit{cost}(\mathcal{T},\mathcal{S})$ is minimized.*

### 4.1.3 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 2.** *The* MIRS *problem is NP-hard.*

*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 [48]. 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). $\square$

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

**Lemma 3.** *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) [67]. 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 [135, 72], 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) [67]. We describe the algorithm in Section 4.2.1. This algorithm has a provable approximation ratio of $1 + \log n$, where $n$ is the number of tips, or customers. $\square$

## 4.2 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 $\mathcal{S} = (S, \{T_r\}_{r \in S})$.

### 4.2.1 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, [67] 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 $c_{rt}$.

**Algorithm 2** Greedy Synthesis

```
Initialize $S = \emptyset$; $T = \mathcal{T}$; $U = \mathcal{U}_{\mathcal{R}}$ while $T \neq \emptyset$ or $U \neq \emptyset$ do
    Find the pair $(r, T_r)$, where $r \in U$ and $T_r \subseteq T$, which minimizes $\frac{f_r + \sum_{t \in T_r} c_{rt}}{|T_r|}$
    Update $S = S \cup r$; $U = U \setminus r$; and $T = T \setminus T_r$
return $S$ and $\forall r \in S, T_r = \{t \in \mathcal{T} | r = \arg\min_{r' \in S} c_{r't}\}$
```

The pseudocode of the Greedy Synthesis algorithm is shown in Algorithm 2. 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

**Algorithm 3** Partitional Synthesis

```
$S_1 = \{r\}$, where $r = \arg\min_{r' \in \mathcal{U}_\mathcal{R}} f_{r'} + \sum_{t \in \mathcal{T}} c_{r't}$.    $C_1 = cost(S, \mathcal{T})$
for $k = 2, \ldots, |\mathcal{T}|$ do
    Let $S_k$ be $k$ random snippets drawn from $\mathcal{U}_\mathcal{R}$.
    repeat
        for $r \in S_k$ do
            $T_r = \{t \in \mathcal{T} \mid r = \arg\min_{r' \in S_k} c_{r't}\}$.
        for $T_r$ do
            $r^* = \arg\min_{r' \in \mathcal{R}} f_{r^*} + \sum_{t \in T_r} c_{r^*t}$.  replace $r$ with $r^*$ in $S_k$.
    until $C_k = cost(S_k, \{T_r\}_{r \in S_k})$ does not change
    if $C_k > C_{k-1}$ then
        break
return $S_{k-1}$ and $\{T_r\}_{r \in S_{k-1}}$
```

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 $S$.

### 4.2.2 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 3. 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., $O(k \times M \times |\mathcal{U}_{\mathcal{R}}| \times |\mathcal{T}|)$.

### 4.2.3 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 4. 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 successful, 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 4** Hierarchical Synthesis

```
$S_1 = \{r\}$, where $r = \arg\min_{r' \in \mathcal{U}_{\mathcal{R}}} f_{r'} + \sum_{t \in \mathcal{T}} c_{r't}$.   $C_1 = cost(S, \mathcal{T})$
for $k = 2, \ldots, |\mathcal{T}|$ do
    repeat
        Let $r$ be the next un-tried snippet in $S_{k-1}$ with highest $\frac{f_r + \sum_{t \in T_r} c_{rt}}{|T_r|}$.
        Let $T_r$ be $\{t \in \mathcal{T} | r = \arg\min_{r' \in S_{k-1}} c_{r't}\}$.
        Find new snippets $r_1$ and $r_2$ using Partitional Synthesis to split $T_r$ into 2 partitions.
        if successful split then
            $S_k = (S_{k-1} \setminus r) \cup \{r_1, r_2\}$  break
    until all snippets in $S_{k-1}$ have been tried
    if no split then
        break
return $S_{k-1}$ and $\{T_r\}_{r \in S_{k-1}}$
```

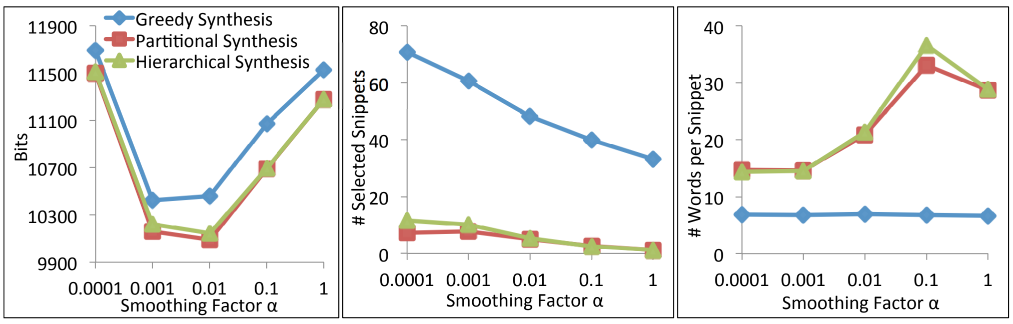
## 4.3 Experiments

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.

**Dataset.** We use the same dataset as in Chapter 3. The statistics of this dataset are shown in Table 4.1 with the new information of the number of snippets. 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 4.1: Statistics of 102 Restaurants in the Dataset



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

Figure 4.1: Comparison of Proposed Synthesis Algorithms

## 4.3.1 Comparison of Proposed Algorithms

We first compare the performance of the three proposed algorithms in Section 4.2, 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 $\alpha$.

Figure 4.1(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 4.1(b) shows the average number of snippets picked by each algorithm, while Figure 4.1(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.

We observe that for all algorithms, as $\alpha$ increases, initially the MDL cost decreases, and then increases again. On one hand, with a smaller $\alpha$, 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 $\alpha$, 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 $\alpha$ changes causes the U-shaped trend in Figure 4.1(a).

The best $\alpha$ 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.

### 4.3.2 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 [101]. 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 4.2 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 4.2 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 4.2 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 [115], 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 [115]). 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).

*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

| 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 4.2: Comparison with Reviews: Representativeness

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 4.3 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 4.3 shows that indeed both Partitional Synthesis and Hierarchical Synthesis produce summaries that are very close to the median length. Partitional

| 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 4.3: Comparison with Reviews: Compactness

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.

### 4.3.3 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[1] [54], which is an example of abstractive summarization, and MEAD[2] [130], which is an example of extractive summarization. For both, we use their default settings. As in [54], 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 pa-

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

rameter 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 4.3.2. Table 4.4 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 4.4: Comparison with Baselines: Representativeness (with respect to reviews)

Table 4.5 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 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 4.5: 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 4.6 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 4.6: 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.

| Method | Readability Score |
|---|---|
| PARTITIONAL SYNTHESIS | 4.23 |
| PARTITIONAL SYNTHESIS WITH TIPS | 3.99 |
| OPINOSIS | 2.07 |
| MEAD | 3.47 |

Table 4.7: Readability Scores

Table 4.7 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.

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 4.2, we show example summaries for the restaurant *Eataly*[3]. The summaries are generated by the summarization techniques that are not based on a limited set of aspects. Instead, each summary is made up by several components. In order to show clearly the separation between the components, we put colors to the different components[4]. For different approaches the notion of component is different. The summaries generated by Partitional Synthesis (Figure 4.2a) and Partitional Synthesis with Tips (Figure 4.2b) are based on the notion of partitions. Therefore, each partition, which is represented by a review snippet, is a component. In the summaries generate by Opinosis (Figure 4.2c) and Mead (Figure 4.2d), each sentence is a component as for these approaches, in each step, one sentence is synthesized or selected, respectively.

The summary by Partitional Synthesis (Figure 4.2a) covers the various

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

[4]Noted that non-consecutive text having the same color do not belong to the same component.

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 4.2b) is compact and dense, but is not as descriptive and narrative. The summary of OPINOSIS (Figure 4.2c) contains useful information and keywords, but it is not presented in a flowing, articulate manner. In the summary of MEAD (Figure 4.2d) 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!*.

## 4.4 Discussion

In this chapter, 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.

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 4.2: Summaries for *Eataly*

# Part II

# Multi-Entity Summarization

# Chapter 5

# Micro-Review Synthesis Based on Multi-Entity Quasi-Cliques

In this work, we propose to generate or synthesize micro-reviews for a collection of entities, by summarizing the micro-reviews of the underlying 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 5.1 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.

Table 5.1: 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)*) |

The multi-entity quasi-clique formulation (defined in Definition 2) is related to the problem of finding dense subgraphs in a large graph [144]. 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 [75], or mining important groups in a network [17], 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 [19], branch and bound [121], pruning [169, 94], mixed integer programming [123], etc. Our objective is not to enumerate all dense subgraphs in a graph [146], 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 5.2).

Specifically, redundancy-aware maximal cliques or RAMC by Wang et al. [157] 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 5.4.

## 5.1 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 $\Omega$ 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 collec-

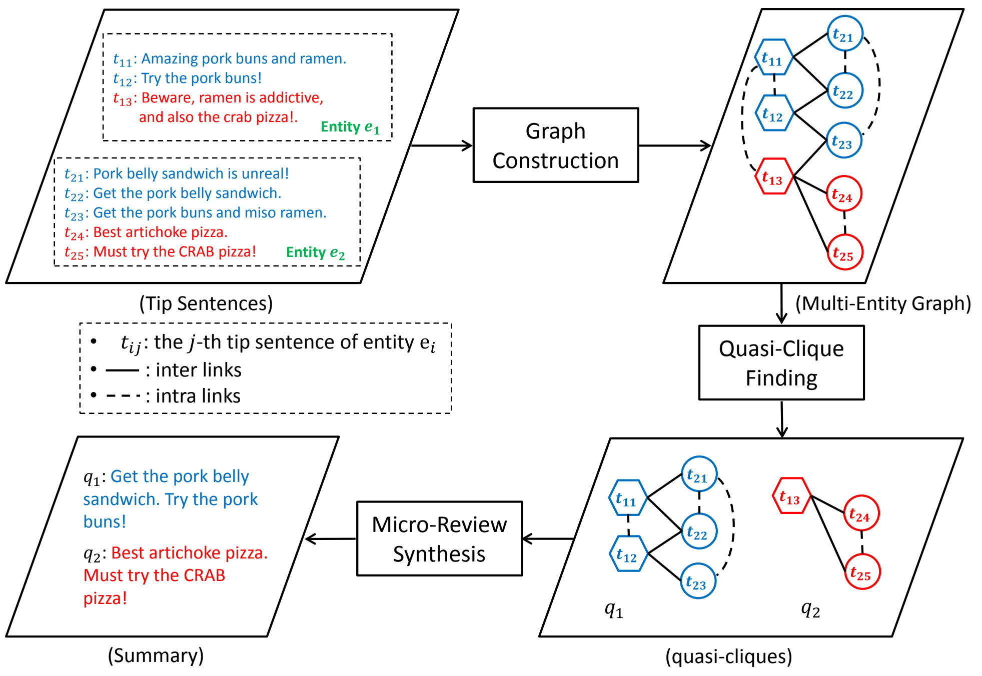

$t_{11}$: Amazing pork buns and ramen.
$t_{12}$: Try the pork buns!
$t_{13}$: Beware, ramen is addictive, and also the crab pizza!.
Entity $e_1$
$t_{21}$: Pork belly sandwich is unreal!
$t_{22}$: Get the pork belly sandwich.
$t_{23}$: Get the pork buns and miso ramen.
$t_{24}$: Best artichoke pizza.
$t_{25}$: Must try the CRAB pizza!
Entity $e_2$
(Tip Sentences)
$t_{ij}$: the $j$-th tip sentence of entity $e_i$
: inter links
: intra links
Graph Construction
(Multi-Entity Graph)
Quasi-Clique Finding
(quasi-cliques)
$q_1$
$q_2$
Micro-Review Synthesis
$q_1$: Get the pork belly sandwich. Try the pork buns!
$q_2$: Best artichoke pizza. Must try the CRAB pizza!
(Summary)

Figure 5.1: Multi-Entity Summarization Framework

tion $\mathcal{C}$ 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 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.** Figure 5.1 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 5.2.1). 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 $e_i$. Figure 5.1 illustrates this using an example of two entities. $t_{ij}$ refers to the $j^{th}$ sentence of entity $e_i$. In Figure 5.1, 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 Figure 5.1, 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 Figure 5.1, 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 5.4).

Aside from cosine similarity, there may be other approaches to determine similarity between sentences, such as based on semantics or topic modeling [16]. The framework in Figure 5.1 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 $\alpha$ and $\beta$. 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 5.2.

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.

$$Cov(Q) = \frac{|\bigcup_{q \in Q} \; q|}{|\mathcal{S}_\mathcal{C}|} \tag{5.1}$$

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* [37]. 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 5.2, we discuss how to derive efficiently a set $Q$ that maximizes coverage.

In Figure 5.1, 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)[1]. The final output is $R_{\mathcal{C}} = \{r_q\}_{q \in Q}$. Figure 5.1 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!”.

## 5.2 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 $\mathcal{C}$.

### 5.2.1 Problem

Given a multi-entity graph $G_{\mathcal{C}}$ (Section 5.1), 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

[1]We discuss this in Section 5.3.

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 $\alpha$, 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* $\alpha$ within each partition and *inter-density* $\beta$ 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*
- *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 [42]. 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 5.1. 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 5.1. 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 4** (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 4 above is computationally intractable. This can be shown by reducing the *Maximum Clique* problem to our problem.

**Lemma 4.** *Maximum Coverage via Multi-Entity Quasi-Cliques is NP-hard.*

*Sketch Proof.* 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 [19, 77], it follows that the former (the more general case) is also NP-hard. $\square$

### 5.2.2 Approach

Due to the computational intractability, it is more productive to seek a heuristic solution. We are inspired by the maximum coverage problem [37], 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 [3], 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 5 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 $\alpha$ and $\beta$, 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

**Algorithm 5** *MMEQC*$(G, \alpha, \beta, K)$

```
Algorithm MMEQC($G, \alpha, \beta, K$)
    $Q = \emptyset$   for $k = 1, 2, \ldots, K$ do
        $q^* = findNextMMEQC(G, \alpha, \beta, Q)$   $Q[k] := q^*$
    end
    return $Q$
Procedure findNextMMEQC($G, \alpha, \beta, Q$)
    $q = \emptyset$   $q = Construct(G, \alpha, \beta, Q)$  $q = Local(G, q, \alpha, \beta, Q)$   return $q$
Procedure Construct($G, \alpha, \beta, Q$)
    $q = \emptyset$   $\alpha^* = 1$   $\beta^* = 1$   $q^* = \{(x_1, x_2)\}$ : 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 $\alpha^*$ to be the lowest intra density in $q^*$   Set $\beta^*$ to be
        the lowest inter density in $q^*$
    end
    return $q$
Procedure Local($G, q, \alpha, \beta, Q$)
    $C = \{t \mid t \in q \wedge t \in Q\}$ : 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 $\alpha, \beta$ 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 $q$
```

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 $\alpha$, $\beta$ 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 5. 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 $\alpha^*$ and the inter-density $\beta^*$. Otherwise, we select $x$ from $\mathcal{N}_{\alpha\beta}(q)$ that only seeks to satisfy the lower minimum thresholds $\alpha$ and $\beta$, after which we then update the current $\alpha^*$ and $\beta^*$ to the actual densities in $q$. If there is no such vertex, $q$ is already a maximal multi-entity quasi-clique meeting the specified $\alpha$ and $\beta$ 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 5.2, 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 5.3, 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 5.4. $q$ has higher potential if its current density is still sufficiently high enough to allow adding more vertices without lowering its density below the $\alpha$ and $\beta$ requirements.

$$\phi_{intra}(q) = \sum_{i=1}^{n} \left( |E_i(q)| - \alpha \binom{|V_i(q)|}{2} \right) \tag{5.2}$$

$$\phi_{inter}(q) = \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \left( |E_{ij}(q)| - \beta |V_i(q)||V_j(q)| \right) \tag{5.3}$$

$$\phi(q) = \phi_{intra}(q) + \phi_{inter}(q) \tag{5.4}$$

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 5.5.

$$\Delta_{q,x} = \sum_{y \in \mathcal{N}_{\alpha\beta}(q) \setminus \{x\},\ y \text{ is uncovered}} \phi(q \cup \{x, y\}) - \phi(q \cup \{y\}) \tag{5.5}$$

**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.

## 5.3 Micro-Review Synthesis

We expect the set $Q_{\mathcal{C}}$ of $K$ maximal multi-entity quasi-cliques obtained through the process described in Section 5.2 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) [38]. 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 5.6, where *length*$(s_i)$ is the length of a sentence $s_i$ in characters[2], and $\lambda$ is a coefficient

[2]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

to regulate the effect of opening cost. In practice, we find that $\lambda = 0.1$ works well in experiments.

$$open(s_i) = \lambda \cdot length(s_i) \tag{5.6}$$

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 5.7. The greater the similarity, the lower is the service cost.

$$service(s_i, s_j) = 1 - cosine(s_i, s_j) \tag{5.7}$$

**Problem 5** (Micro-Review Synthesis)**.** *Given a maximal multi-entity quasi-clique $q$ and a length limit of $\gamma$, 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 [38]. There are different approaches to solve the problem [135]. 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

limit to be 200 characters. Therefore, we measure sentence length in terms of characters.

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, [67] shows that, for each $s_i$, it is sufficient to consider the pairs $(s_i, T^k_{s_i})$, for $k = 1, ..., |q|$, where $T^k_{s_i}$ 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 $\gamma$. 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 $\gamma$. This is more advantageous than prematurely terminating the greedy when $\gamma$ 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.

## 5.4 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.*

### 5.4.1 Experimental Setup

**Dataset.** We use the Foursquare data collected by [116]. 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 5.2 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,

Table 5.2: 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 5.3: 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 |

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 5.3, 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 5.3 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 5.3. 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 5.2, followed by micro-review synthesis described in Section 5.3. 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) [157] tries to produce a concise and complete summary of a set of maximal cliques. The output of RAMC is dependent on the *visibility* parameter, $\tau$, which reflects the percentage of coverage that each maximal clique will be covered by the selected summary. We use the author implementation[3] to find the top $K$ maximal cliques, following the procedure described in [157] with deterministic setting and global filtering.

4. *MEAD*: We run extractive summarization MEAD [130] 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[4].

5. *TextRank*: Another extractive method, but based on ranking, is TextRank [109]. 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 [54], using the author implementation[5] with standard settings. Opinosis first forms a graph with words as vertices from the con-

[3]`https://github.com/cntswj/clique-summary`
[4]http://www.summarization.com/mead/
[5]http://kavita-ganesan.com/opinosis-summarizer-library

tent in a collection, then selects the $K$ highest-scoring paths (sentences) as summary.

**Graph Construction.** As described in Section 5.2, 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. Figure 5.2 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.

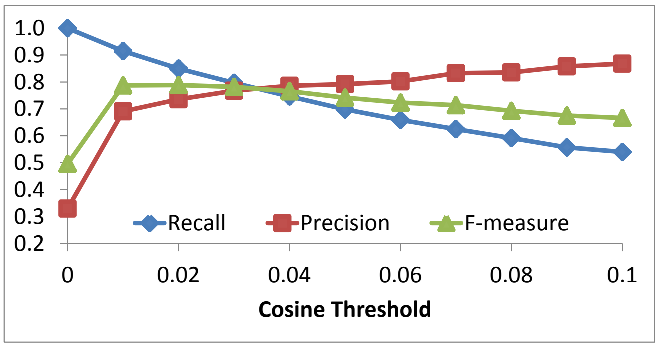


Figure 5.2: Graph Construction: Cosine Similarity Threshold

### 5.4.2 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 5.4.1.

**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 [157], we vary the visibility parameter $\tau$ 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 $\alpha$ 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 $(\alpha, \beta)$, 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.

Table 5.4: *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 5.5: *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 5.6: *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* |

**Results.** Table 5.4 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 5.4, 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, *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 5.5, and Table 5.6 respectively. The trends are essentially the same: the results and the observations echo those for *ZIP Codes*.

In Figure 5.3, 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*.

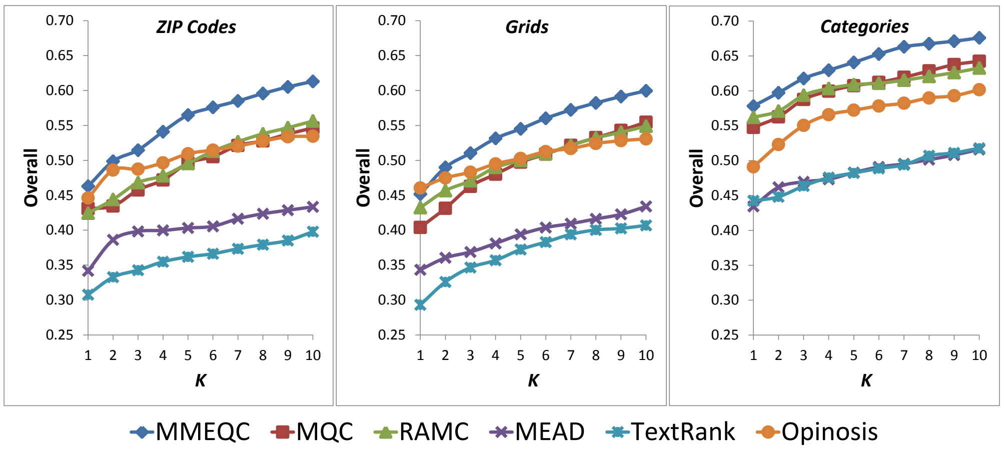


Figure 5.3: The Overall Scores: Vary the Number of Summary Tips $K$

## 5.4.3 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 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* [101]. *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

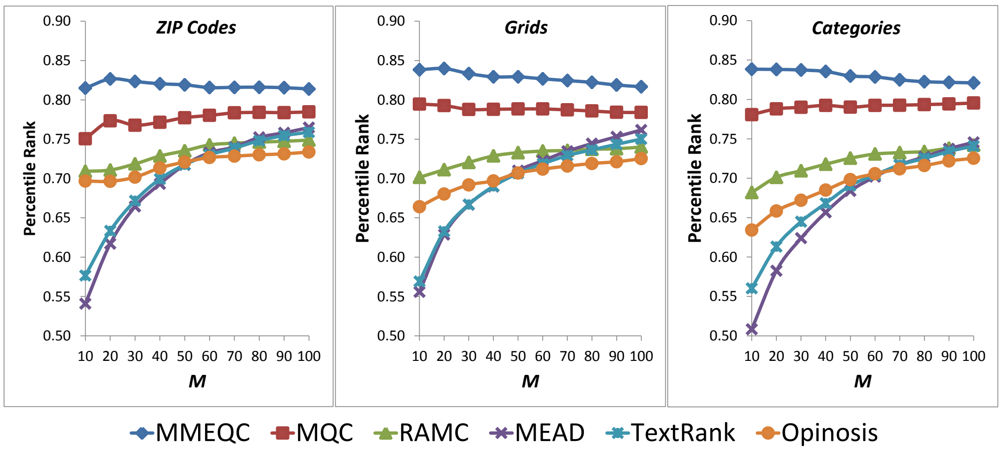


Figure 5.4: Relevance Ranking with $K = 10$: Vary the Number of Queries $M$

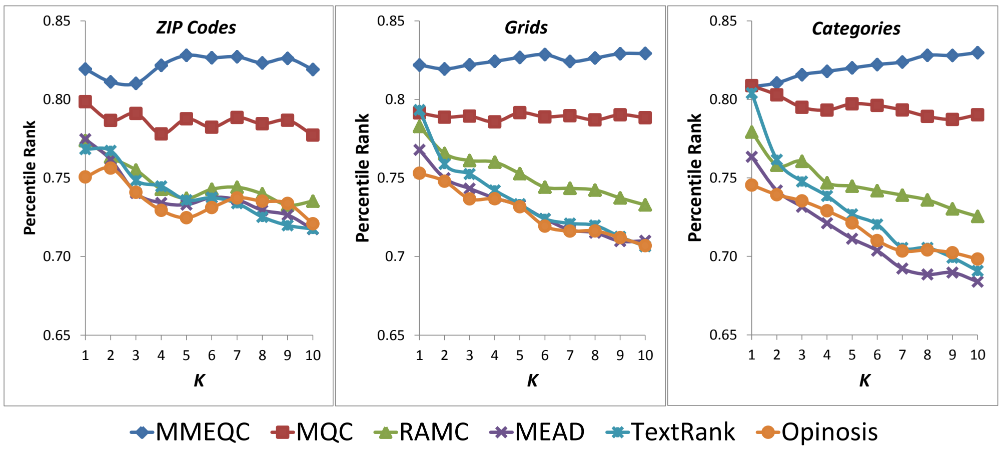


Figure 5.5: Relevance Ranking with $M = 50$: Vary the Number of Summary Tips $K$

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.

Figure 5.4 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 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. Figure 5.5 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.

### 5.4.4 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 5.7 shows the average readability scores by the human judges as well as the standard deviations. The first four listed in Table 5.7, 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 5.7 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.

Table 5.7: User Study on Readability

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

Table 5.8: The Correlation Between Readability and Representativeness, Diversity, and Coherence

| | *Representativeness* | *Diversity* | *Coherence* |
|---|---|---|---|
| Correlation with *Readability* | 0.14 | 0.43 | 0.21 |

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.

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. Moreover, as shown in Table 5.8, the readability scores are positively correlated with the three metrics individually. The results are reasonable that users consider the summaries which are representative, diverse and/or coherence to be readable. Especially, having the highest correlation with readability, diversity is shown to be an important factor that users prefer summaries that are diversified over the input entities.

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 5.7.

## 5.5 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.

### 5.5.1 Benchmarking against a Black-Box Optimization Algorithm

In Section 5.4, 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 [82] 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 5.4, namely: *ZIP Codes*, *Grids*, and *Categories*. Table 5.9 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 5.10. 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

Table 5.9: 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 5.10: 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 |

meeting the length restriction. The resulting 10 summary tips, one for each cluster of entities, make up the summary.

As shown in Table 5.9, this approach, referred to as *KMeans*, has the worst *Overall* performance among the algorithms. Because one or more than one entity 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 5.10 is at the expense of much lower *Overall* scores.

### 5.5.2 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 $6 \times 6$ grid in the center of the map. As shown in Table 5.11, 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 $20 \times 20$ 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 5.11 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. Figure 5.6 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 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 5.10). Moreover, as shown in Figure 5.6 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*[6] on this dataset. Figure 5.6 shows that *MMEQC* is consistently

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

Table 5.11: 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* |

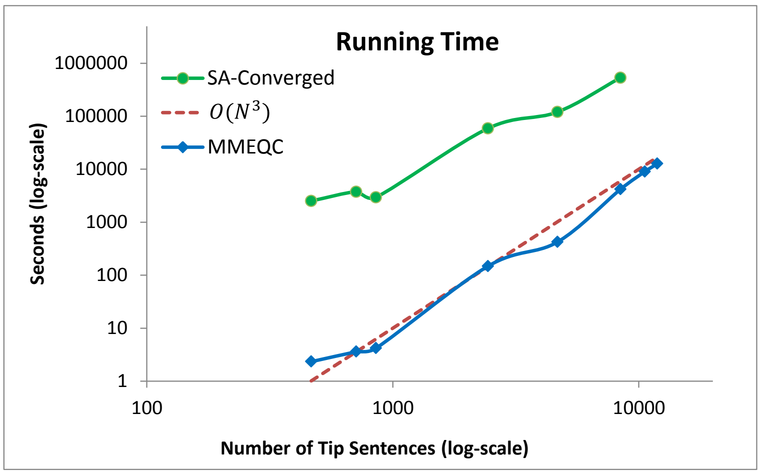


Figure 5.6: Increasing Grid Sizes: Running Times in seconds

much faster than *SA-Converged* across different input sizes.

## 5.6 Discussion

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 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.

# Chapter 6

# Query-Dependent Ranking of Entities

In this chapter, we consider the problem of ranking entities based on reviews and reviewer expertise. We propose two models for review-based ranking of entities, which treat both reviews and reviewers as first-class citizens, recognizing their primary roles in identifying the entities that are most apt at addressing the need expressed by the query. Both our proposed models are based on probabilistic language modeling techniques that have been shown to be useful for various information retrieval tasks, including *expertise retrieval* [7]. Each model ranks the entities based on the probability that an entity is apt for the query, but the two models differ in how this is done.

In our first model, *Review-Based Entity Ranking* or *ReBER*, we decompose the entity probability into review-level probabilities. In turn, these review-level probabilities seek to capture signals from reviews, such as how popular an entity is, how relevant a review is to the query, as well as the degree to which a review lends its support to the entity of interest.

However, reviews written by different reviewers are not identical in quality, since some reviewers might have more knowledge on a particular topic, or more respected than others. We associate such traits with the notion of *reviewer expertise*. In our second model, *User Expertise-Based Entity Ranking* or *UEBER*, we decompose the entity probability into user-level probabilities, to further incorporate the expertise of reviewers when deriving the rankings. Specifically, during the estimation, each review is now weighted by how likely its author is an "expert" with regards to the query.

In the next section, we discuss the development of our two proposed probabilistic entity-ranking models.

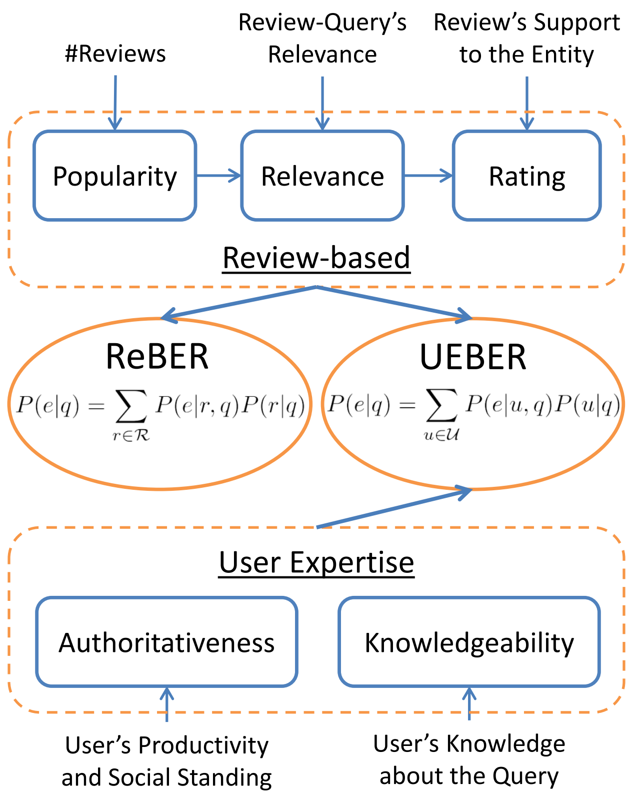
## 6.1 Entity Ranking Models

Let $\mathcal{E}$, $\mathcal{U}$, and $\mathcal{R}$ respectively denote the set of all entities, users, and reviews. Entities in $\mathcal{E}$ belong to the same domain (e.g., restaurants). A review $r \in \mathcal{R}$ is written by a user $u \in \mathcal{U}$ for an entity $e \in \mathcal{E}$, and is associated with a textual content $d_r$ as well as a numerical rating $\Gamma_r$. The subset of reviews concerning entity $e$ is denoted $\mathcal{R}_e \in \mathcal{R}$, while the subset of reviews written by user $u$ is denoted $\mathcal{R}_u \in \mathcal{R}$.

The problem scenario we are modeling is one where in response to a query $q$, we would like to return a ranked list of entities in $\mathcal{E}$, whereby a higher-ranked entity is considered more *apt* for the query. For instance, the query "frappuccino" likely ranks Starbucks outlets highly given that it is a trademarked brand of that corporation, whereas a more general query "soy latte" may list both popular chains as well as highly-regarded independent cafes in the locality.

To model the *aptness* of an entity $e$ for the query $q$, we associate each entity with a probability $P(e|q)$. One interpretation is the likelihood of $e$ being selected when someone is interested in $q$. Conceivably, there are various possible sources of information to estimate this probability, such as clickthrough, purchases, footfall, etc. However, such data tend to be opaque and proprietary, making it challenging to explain openly how the probabilities are derived. In this work, our approach is to estimate this probability from the corpus of re-

Figure 6.1: Entity Ranking Models


#Reviews
Review-Query's Relevance
Review's Support to the Entity
Popularity
Relevance
Rating
Review-based
ReBER
$P(e|q) = \sum_{r \in \mathcal{R}} P(e|r,q)P(r|q)$
UEBER
$P(e|q) = \sum_{u \in \mathcal{U}} P(e|u,q)P(u|q)$
User Expertise
Authoritativeness
Knowledgeability
User's Productivity and Social Standing
User's Knowledge about the Query

views, which are publicly available, providing some degree of transparency on the entity rankings.

We present two approaches to estimate $P(e|q)$. In the first approach, **R**eview-**B**ased **E**ntity **R**anking or *ReBER*, we emulate the experience of "reading" through reviews to gather positive signals about the entity. In the second approach, **U**ser **E**xpertise-**B**ased **E**ntity **R**anking or *UEBER*, we advocate a deeper reading of reviews, not only to consider *what* is being said within a review, but importantly also *who* says it, i.e., the user who writes a review. These two approaches, and the different factors that go into them are illustrated in Figure 6.1, which we will elaborate in the subsequent sections.

### 6.1.1 Review-Based Entity Ranking (*ReBER*)

There are various signals we can gather from reviews about the aptness of an entity for a given query.

*Popularity.* Intuitively, when we hear good comments about an entity frequently, we begin to develop a positive inclination towards the entity. Analogously, if many reviews "support" an entity $e$ for query $q$, the higher is $P(e|q)$. To incorporate this notion of popularity, our first step is to decompose the entity-level aptness $P(e|q)$ into review-level aptness $P(e, r|q)$, as shown in the following equation.

$$P(e|q) = \sum_{r \in \mathcal{R}} P(e, r|q) \tag{6.1}$$

We turn to the definition of $P(e, r|q)$. The degree to which review $r$ provides evidence in support of entity $e$'s aptness for query $q$ ultimately comes down to two main factors. First, whether $r$ is relevant for the query $q$, which we denote probabilistically as $P(r|q)$. Second, whether $r$ speaks positively about entity $e$, which we denote probabilistically as $P(e|r, q)$. We represent $P(e, r|q)$ as the product of these two probabilities, and reformulate Equation 6.1 into Equation 6.2.

$$P(e|q) = \sum_{r \in \mathcal{R}} P(e|r, q)P(r|q) \tag{6.2}$$

*Relevance.* The relevance of a review $r$ to a query $q$ or $P(r|q)$ depends primarily on how well its textual content $d_r$ matches the intent of $q$. Since we are developing the model probabilistically, we rely on probabilistic language modeling. As in [8], we estimate relevance between a document (review) and the query according to the smoothed language model of the document, as follows:

$$P(q|r) = \prod_{t \in q} ((1 - \alpha)P(t|d_r) + \alpha P(t))^{n(t,q)} \tag{6.3}$$

where $t$ is a term in query $q$, $\alpha$ is the smoothing parameter, $P(t|d_r)$ is the frequency of term $t$ in document $d_r$, $P(t)$ is the frequency of the term in the whole corpus and $n(t, q)$ denotes the number of times term $t$ appears in query $q$.

Based on the Bayes' Theorem, we can write $P(r|q) \propto P(q|r)P(r)$. Here,

$P(r)$ may be understood as the prior probability of review $r$, and can be used to encode review-specific characteristics such as recency. In this work, we rely mainly on the relevance, and estimate $P(r|q)$ from $P(q|r)$ alone (Equation 6.3), assuming that $P(r)$ is uniform.

*Support.* Though a review may be relevant for the context of the query, it may or may not "support" an entity. One interpretation of "support" is whether the review mentions an entity and expresses a positive sentiment. However, extracting the entity mentions and the sentiments expressed from review text is itself an open problem, potentially inducing uncertainties regarding the correctness of the extraction. Therefore, for a more definitive signal, in this work, we turn to the explicit rating expressed by a review. Since review $r$ is written for a specific entity, we make the simplifying assumption that $P(e|r,q) > 0$ only if $r \in \mathcal{R}_e$, and 0 otherwise. We further assume that $P(e|r,q) \propto \Gamma_r$, i.e., the higher the rating the higher is $P(e|r,q)$. To convert a rating to a probability, we normalize it with the maximum rating.

$$P(e|r,q) = \begin{cases} \frac{\Gamma_r}{max\_rating} & \text{if } r \in \mathcal{R}_e \\ 0 & \text{otherwise} \end{cases} \tag{6.4}$$

Both Equations 6.3 and 6.4 can be estimated from the corpus of reviews $\mathcal{R}$. These are then incorporated into Equation 6.2 to derive the probability $P(e|q)$ to be used for ranking entities.

### 6.1.2 User Expertise-Based Entity Ranking (*UEBER*)

While the review-based model above grasps signals from trawling the sea of reviews, it treats reviews merely as documents whose primary feature is their text content. Arguably, not all reviews are created equal. Some reviews may be created by novice users who may be trying out the review site for a short while, and never return. Others may be created by expert users who prolifically author reviews and carefully cultivate a reputation for themselves. A recommendation from the latter is worth more than from the former.

Therefore, we advocate another approach that pays attention to the person behind the review. Differently from *ReBER* that decomposes the aptness $P(e|q)$ into review-level aptness in Equation 6.2, *UEBER* decomposes it into user-level aptness, as shown in Equation 6.5. The summation implies that the more users support an entity, the higher is $P(e|q)$.

$$P(e|q) = \sum_{u \in \mathcal{U}} P(e|u,q)P(u|q) \tag{6.5}$$

The degree to which user $u$ provides evidence in support of entity $e$'s aptness for query $q$ ultimately comes down to the following main factors. One is whether $u$ speaks in favor of $e$ in the context of query $q$, denoted $P(e|u,q)$. Since this is expressed by a review $r$ that $u$ has written for $e$, we can decompose it into the review-level factors incorporated into *ReBER*, i.e., $P(e|r,q)$ expressed in Equation 6.4 and $P(r|q)$ expressed in Equation 6.3, as shown below.

$$P(e|u,q) = \sum_{r \in \mathcal{R}_{ue}} P(e|r,q)P(r|q) \tag{6.6}$$

$\mathcal{R}_{ue}$ is the set of reviews that user $u$ has written for entity $e$. In most cases, it contains a single review. For the restaurants dataset in our experiments, we observe that some users (less than 5% of the population) have written multiple reviews for an entity; in one case a user writes 19 reviews for a restaurant. This phenomenon happens due to repeated visits to the restaurant, each resulting in a review. The summation in the equation above essentially assumes that repeated visits constitute an endorsement. In another domain, e.g., electronics, repeated reviews may imply updated views due to factors such as durability that can only be observed later, in which case the appropriate way is to take the latest review.

The difference between *UEBER* and *ReBER* then comes down to the other main factor $P(u|q)$ in Equation 6.5, which we refer to as *user expertise.* $P(u|q)$ represents how much weight to place on what $u$ says about the subject matter expressed by the query $q$. On one hand, this depends on whether $u$ is *knowledgeable* about the subject matter $q$, denoted $P(q|u)$. The greater the likelihood that $u$ discourses on $q$, as opposed to other subject matters, the more knowledgeable $u$ is about $q$. On the other hand, it also depends on how *authoritative* $u$ is in general, commanding respect from others regardless of the query or topic at hand. We thus express user expertise $P(u|q)$ in terms of these two concepts of knowledgeable and authoritative, as shown in Equation 6.7.

$$P(u|q) \propto P(q|u)P(u) \tag{6.7}$$

*Knowledgeable.* User $u$'s knowledge of the query $q$ or $P(q|u)$ could be estimated from the degree to which $\mathcal{R}_u$ or reviews written by $u$ across various entities mention the words within $q$. We do so by decomposing $P(q|u)$ into review-level relevance $P(q|r)$ and the weight assigned to each review $P(r|u)$, as shown in Equation 6.8.

$$P(q|u) = \sum_{r \in R_u} P(q|r)P(r|u) \tag{6.8}$$

We estimate $P(q|r)$ the same way as in Equation 6.3. In turn, $P(r|u)$ is the averaging term, fractional to the number of reviews user $u$ has. To avoid too large differences in the averaging term between a review written by a user who has only one review and a review written by a user who has many reviews, we apply smoothing when computing $P(r|u)$ as follows:

$$P(r|u) = \begin{cases} \frac{1+\beta}{|R_u|+\beta|R|} & \text{if } r \in R_u \\ 0 & \text{otherwise} \end{cases} \tag{6.9}$$

where $|R_u|$ is the number of reviews written by $u$, $|\mathcal{R}|$ is the total number of reviews in the corpus, and $\beta$ is the smoothing parameter controlling the difference rate.

*Authoritative.* The authoritativeness of user $u$ or $P(u)$ is estimated based on her productivity and social standing. A user is productive when she has written many reviews. The social standing is implied by whether she has many friends, who are themselves authoritative.

The social relationship between users can be transformed into a graph, in which each node is a user and there is a link between two nodes if the two corresponding users are friend. To estimate the authoritativeness $P(u)$, we apply PageRank [120] on the social graph, using the information of the number of reviews and friends, as follows:

$$P(u) = (1-\lambda)\frac{|R_u|}{|\mathcal{R}|} + \lambda \sum_{u' \in \mathcal{U}} P(u|u')P(u') \tag{6.10}$$

where $\lambda$ is the damping factor.

The transition probability $P(u|u')$ is defined as follows:

$$P(u|u') = \begin{cases} \frac{1}{|F_{u'}|} & \text{if } u \text{ and } u' \text{are friend} \\ 0 & \text{otherwise} \end{cases} \tag{6.11}$$

where $F_{u'}$ denotes the set of the friends of user $u'$.

To confine the PageRank values to between 0 and 1, we normalize $P(u)$ as follows:

$$norm(x) = \frac{f(x) - min_{x'}(f(x'))/\tau}{max_{x'}(f(x')) - min_{x'}(f(x'))/\tau} \tag{6.12}$$

where $x$ is the original value of *P(u)*, $\tau$ is the normalizing factor (we find that $\tau = 1.05$ works well for our dataset[1]), and the function $f(x)$ is formulated as:

$$f(x) = -\frac{1}{\log_{10} x} \tag{6.13}$$

[1]As to avoid 0 value for P(u), $\tau$ is set to be greater than 1.0

Table 6.1: Number of Entities (i.e., Restaurants) and Number of Reviews in an Entity in the Dataset

| City | #Entities | #Reviews in an Entity | | | |
|---|---|---|---|---|---|
| | | Min | Max | Mean | Median |
| Charlotte | 2793 | 1 | 1044 | 41.6 | 17 |
| Edinburgh | 1993 | 1 | 166 | 11.6 | 7 |
| Las Vegas | 7268 | 1 | 5558 | 97.4 | 25 |
| Phoenix | 4365 | 1 | 1578 | 58.6 | 19 |
| Pittsburgh | 2177 | 1 | 1096 | 40.2 | 16 |

In summary, *UEBER* incorporates the review-level signals modeled by *ReBER* such as textual relevance and ratings, while also paying attention to the expertise of the user behind the review. The latter is informed by rich signals including the social network among review writers.

## 6.2 Experiments

The objective of experiments is to investigate the effectiveness of the proposed *ReBER* and *UEBER* models in ranking entities based on queries. After describing the setup of experiments, we investigate several research questions that provide insights to the workings of *ReBER* and *UEBER* models. This is then followed by comparison to baselines.

### 6.2.1 Experimental Setup

**Dataset.** We rely on the review dataset provided by Yelp[2], comprising more than a million reviews for more than 18,000 restaurants from 5 cities: Charlotte, Edinburgh, Las Vegas, Phoenix, and Pittsburgh. Table 6.1 shows the details about the number of entities (i.e., restaurants), as well as the statistics of the number of reviews in each entity.

**Queries and Ground-Truth Lists.** To assess the quality of the rankings generated by the models in response to a query, we would like to compare them to human-generated lists. There are various potential sources for such lists. For

[2]`https://www.yelp.com/dataset/challenge`

Table 6.2: Sample Queries for Each City.

| *Charlotte* | *Edinburgh* | *Las Vegas* | *Phoenix* | *Pittsburgh* |
|---|---|---|---|---|
| Buger | Beer | Buffets | Burger | Asian |
| Italian | Brunch | Donut | Coffee | Brunch |
| Vegan | Cake | Japanese | Mexican | Coffee |
| Vietnamese | Coffee | Sandwich | Vegan | Pizza |

Table 6.3: Statistic Information of Queries and Ground-Truth (GT) Lists.

| City | #Queries | #Restaurants in a GT List | | | |
|---|---|---|---|---|---|
| | | Min | Max | Mean | Median |
| Charlotte | 9 | 5 | 12 | 8.1 | 10 |
| Edinburgh | 4 | 5 | 10 | 7.3 | 7 |
| Las Vegas | 26 | 5 | 44 | 18.9 | 18 |
| Phoenix | 6 | 6 | 21 | 12.0 | 11 |
| Pittsburgh | 4 | 6 | 13 | 10.0 | 11 |

instance, a foodie may write a blog of their favorite restaurants, or a journalist may write an article recommending food places. However, there are not many such lists on the open Web, and matching the entities mentioned there to the entities in the dataset would be very challenging. In this work, we rely on lists created by Yelp users. Yelpers create lists for different purposes, such as for bookmarking their favorite places, or for sharing recommended venues with others. For example, Paul B. created a list of "*My top 5 buffets in Vegas*"[3] to share his top buffet restaurants in Las Vegas. There are multiple lists from different users, which can be aggregated to form a "consensus" ground truth. Note that these lists are not used directly by the ranking models.

For queries, we use the categories in Yelp. The sample queries for each city are shown in Table 6.2. To produce the ground-truth list for each query, we crawled all lists containing the query and the city in the title. Then, we aggregate the lists and record the aggregated agreement values (i.e., votes) for each entity. Any user can contribute at most one vote for a restaurant for each query. The entities are then ranked based on these agreement values. An entity recommended by many users is likely to be good indeed, and we only keep the entities that occur within at least 3 lists. Finally, we only keep the

[3]`https://www.yelp.com/list/my-top-5-buffets-in-vegas-las-vegas`

queries that have at least 5 restaurants in its ground-truth list. This results in 49 queries. Table 6.3 shows the number of queries in each city, as well as the statistical information of the number of restaurants in a ground-truth list.

**Metrics.** To evaluate the output rankings, we compare them with the ground-truth lists on the following metrics. *Precision@K* measures the proportion of the correctly-selected restaurants among in the top-$K$ list produced by a model. It considers all the entities in the ground-truth to be equally relevant to the query. In contrast, *Normalized Discounted Cumulative Gain at K (NDCG@K)* takes into account the ranking order of the results, i.e., the ones that are more relevant (i.e., ranked higher in the ground-truth list) should appear earlier in the output ranking list.

The discounted cumulative gain ($DCG$) of top-$K$ list is computed using the following equation:

$$DCG_K = \sum_{i=1}^{K} \frac{2^{rel_i} - 1}{\log_2(i+1)} \tag{6.14}$$

where $rel_i$ is the relevance score of the $i^{th}$ result to the query. $DCG_K$ is normalized as follows:

$$NDCG_K = \frac{DCG_K}{IDCG_K} \tag{6.15}$$

where $IDCG_K$ is the maximum possible $DCG$ at $K$. The relevance scores ($rel$) are assigned based on binning the aggregated agreement values: $rel = 1$ for the agreement values of 1, 2, or 3; $rel = 2$ for the agreement values of 4, 5, or 6; and $rel = 3$ for the agreement values greater than 6.

**Parameters.** The parameters are tuned based on 10-fold cross-validation using *NDCG@5*. The tuned settings are $\alpha = 0.0001$, $\lambda = 0.85$ and $\beta = 0.01$, which we use in subsequent experiments.

Table 6.4: Comparison between *ReBER* and *ReBER-w/o_Rating*. $^*$ indicates statistically significant improvements over *ReBER-w/o_Rating*.

| | *Precision@K* | | | | *NDCG@K* | | | |
|---|---|---|---|---|---|---|---|---|
| | @5 | @10 | @20 | @30 | @5 | @10 | @20 | @30 |
| *ReBER-w/o_Rating* | 0.567 | 0.476 | 0.351 | 0.278 | 0.518 | 0.552 | 0.594 | 0.618 |
| *ReBER* | 0.567 | 0.492$^*$ | 0.354 | 0.284$^*$ | 0.519 | 0.569$^*$ | 0.609$^*$ | 0.633$^*$ |

### 6.2.2 Research Questions

We begin by investigating several research questions exploring the effects of the factors incorporated in the proposed *ReBER* and *UEBER* models.

#### RQ1. Is *rating* important?

When developing the *ReBER* model, in addition to the relevance of a review to a query $P(r|q)$, we also consider the rating value $\Gamma_r$ of the review to estimate $P(e|r,q)$ or how supportive the review is to the concerned entity. To see the effect of rating information, we compare the results of *ReBER* with a variant without the rating information *ReBER-w/o_Rating*, whereby $P(e|r,q)$ is binary (1 if $r$ is about $e$, and 0 otherwise). Table 6.4 shows the comparison in terms of *Precision@K* and *NDCG@K* for $K \in \{5, 10, 20, 30\}$. Aided by the rating information, *ReBER* performs better than *ReBER-w/o_Rating*. Although in some cases the improvement is slight, the differences are statistically significant (at 0.05 level) for the top 10 and 30 in terms of *precision* and for the top 10, 20, and 30 in terms of *NDCG*. Since not all reviews are positive about an entity, a higher rating is a clearer signal of support. The answer to this research question is therefore affirmative.

#### RQ2. Do we need all the reviews?

In *ReBER* model, a ranking is generated based on all the reviews in the corpus. As indicated in Table 6.1, some entities have thousands of reviews. One alternative to using all reviews, is to rely on only the top $N$ reviews that are most relevant to the query (based on $P(r|q)$) to estimate the *aptness* of each entity. As such, only entities with at least one review in the top $N$ could have non-zero *aptness*. Figure 6.2 shows the trends of *Precision@K* and *NDCG@K*

Figure 6.2: Varying the Number of Top Query-Relevant Reviews Used.

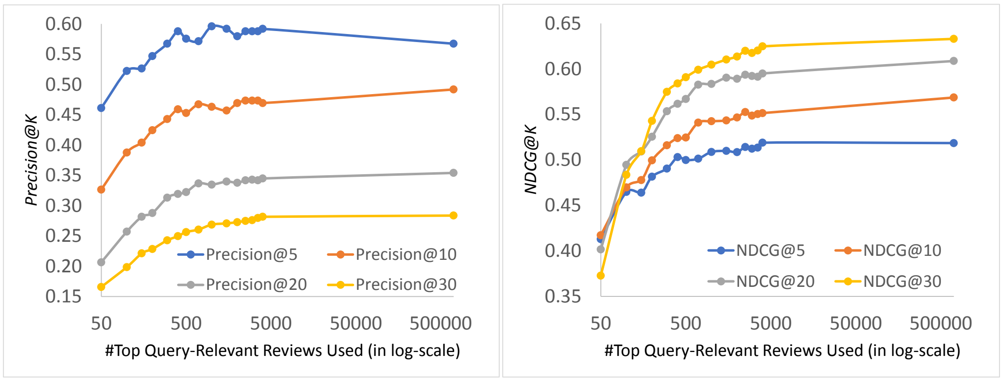


Table 6.5: Evaluating the Impacts of *User Expertise* on Entity Ranking. $^{*}$,$^{\dagger}$ ,$^{\ddagger}$ indicate statistically significant improvements over *ReBER*$^{*}$, *UEBER-Auth*$^{\dagger}$ and *UEBER-Knowl*$^{\ddagger}$, respectively.

| | *Precision@K* | | | | *NDCG@K* | | | |
|---|---|---|---|---|---|---|---|---|
| | @5 | @10 | @20 | @30 | @5 | @10 | @20 | @30 |
| *ReBER* | 0.57 | 0.49 | 0.35 | 0.28 | 0.52 | 0.57 | 0.61 | 0.63 |
| *UEBER-Auth* | 0.60$^{*}$ | 0.51$^{*}$ | 0.37$^{*}$ | 0.29 | 0.55$^{*}$ | 0.59$^{*}$ | 0.63$^{*}$ | 0.65$^{*}$ |
| *UEBER-Knowl* | 0.62$^{*}$ | 0.51 | 0.38$^{*}$ | 0.30$^{*}$ | 0.55 | 0.59 | 0.64$^{*}$ | 0.66$^{*}$ |
| *UEBER* | 0.65$^{*\dagger\ddagger}$ | 0.53$^{*\ddagger}$ | 0.39$^{*\dagger\ddagger}$ | 0.31$^{*\dagger\ddagger}$ | 0.57$^{*\ddagger}$ | 0.61$^{*\ddagger}$ | 0.65$^{*\dagger\ddagger}$ | 0.68$^{*\dagger\ddagger}$ |

($K = \{5, 10, 20, 30\}$) when varying the number of $N$ from 50 to the maximum number of reviews we have in a city (i.e., using all reviews). It shows that very low $N$ is not effective. As $N$ increases, there is significant performance gain, initially steeper and later gentler. For *Precision@5* and *NDCG@5*, there is a slight drop at the end (not statistically significant). However, for $K = \{10, 20, 30\}$, the trend is clearer: ultimately using all reviews performs the best. These curves show that though it may be possible to achieve similar results with fewer reviews, there is still some value to incorporating all reviews.

## RQ3. Does *user expertise* have an impact?

To answer this question, we compare the ranking performances between *ReBER* and *UEBER* model. As shown in Table 6.5, *UEBER* (last row) significantly outperforms *ReBER* (first row) in terms of both *precision* and *NDCG*. Taking into account the expertise of the reviewer behind each review, *UEBER* model could place more weight on the support from "expert" users as opposed to those from "novice" users.

## RQ4. Which expertise factor is more useful, *authoritativeness* or *knowledgeability*?

*User expertise* is composed of two factors: *authoritativeness* and *knowledgeability*. To examine the influence of each factor, we evaluate the rankings resulted by "turning off" either authoritativeness or knowledgeability:

- *UEBER* with *authoritativeness* only (*UEBER-Auth*)
- *UEBER* with *knowledgeability* only (*UEBER-Knowl*)

Table 6.5 shows that the complete model *UEBER* significantly outperforms both *UEBER-Auth* and *UEBER-Knowl*. This implies that both expertise factors are useful, and removing either one degrades the performance of the model. It is also instructive to see how *UEBER-Auth* and *UEBER-Knowl* compare to *ReBER* that does not incorporate user expertise at all. Here, we also see that both expertise factors result in some performance gains underscoring their usefulness. Between the two, *UEBER-Knowl* appears to be slightly better than *UEBER-Auth*, indicating that being knowledgeable may have an edge over being authoritative, but the differences are not significant. Both factors are useful, and the proposed model *UEBER* benefits by incorporating them.

## RQ5. Would it have helped to focus on only "Elite" reviews?

Yelp recognizes some users with an "*Elite*" badge[4]. Conceivably, an elite user likely produces high-quality reviews. In our dataset, there are more than a million reviews in total, and about one third were written by elite users. In our models, we do not use eliteness information, because this is Yelp-specific. However, we wonder if by doing so, we are missing out on important signals.

In Table 6.6, we first see a comparison between the original review-based *ReBER* that uses all reviews, and a variant that considers only elite reviews.

[4]`https://www.yelp-support.com/article/What-is-Yelps-Elite-Squad?l=en_GB`

Table 6.6: "User Expertise" vs. "Elite". Evaluating *ReBER* and *UEBER* Models on Elite Reviews Only and All Reviews. * indicates statistically significant improvements over *UEBER* Running on Elite Reviews Only.

| Model | Review Set | Precision@K | | | | NDCG@K | | | |
|---|---|---|---|---|---|---|---|---|---|
| | | @5 | @10 | @20 | @30 | @5 | @10 | @20 | @30 |
| *ReBER* | Elite Only | 0.59 | 0.49 | 0.36 | 0.27 | 0.51 | 0.56 | 0.60 | 0.62 |
| | All Reviews | 0.57 | 0.49 | 0.35 | 0.28 | 0.52 | 0.57 | 0.61 | 0.63 |
| *UEBER* | Elite Only | 0.62 | 0.52 | 0.39 | 0.30 | 0.55 | 0.58 | 0.63 | 0.65 |
| | All Reviews | 0.65 | 0.53 | 0.39 | 0.31* | 0.57 | 0.61* | 0.65* | 0.68* |

Here, the outcomes are similar, and there does not seem to be much gain coming from limiting the model to only elite reviews. We then turn to the expertise-based model *UEBER*, for which Table 6.6 shows that using all the reviews performs better than using only elite reviews. The improvements are significant in terms of *precision* (top 30) and *NDCG* (top 10, 20 and 30). This recognizes the contributions by non-elite users to the ranking task. In particular, user expertise helps to discriminate among the different users, hence the *elite* information does not offer much, if any, additional information.

### 6.2.3 Comparisons to Baselines

We now evaluate our user-based entity ranking model with baseline approaches from *expert finding* and *opinion-based entity ranking* respectively.

#### Comparison to Expert Finding Models

In a way, our ranking problem is related to *expert finding* or *expertise retrieval*, in which the objective is to identify experts in a particular area or topic. In expert finding, we seek to identify experts from among a population of candidates based on a query [7, 8]. The closest expert finding work to ours, which is based on probabilistic modeling, is Balog et al.'s [7] that proposes two different ways to model the expertise of a candidate expert and both are based on generative probabilistic model. The first approach (using candidate models, *Model 1*) directly estimates the expertise of the candidates based on the documents that they are associated with. The basic idea is to use the

Table 6.7: Compare *UEBER* against Expert Finding Models. $^*$,$^\dagger$ indicate statistically significant improvements over *Balog-M1*$^*$ and *Balog-M2*$^\dagger$, respectively.

| | *Precision@K* | | | | *NDCG@K* | | | |
|---|---|---|---|---|---|---|---|---|
| | @5 | @10 | @20 | @30 | @5 | @10 | @20 | @30 |
| *Balog-M1* | 0.57 | 0.48 | 0.35 | 0.28 | 0.52 | 0.55 | 0.59 | 0.62 |
| *Balog-M2* | 0.57 | 0.48 | 0.35 | 0.28 | 0.52 | 0.55 | 0.59 | 0.62 |
| *UEBER* | $0.65^{*\dagger}$ | $0.53^{*\dagger}$ | $0.39^{*\dagger}$ | $0.31^{*\dagger}$ | $0.57^{*\dagger}$ | $0.61^{*\dagger}$ | $0.65^{*\dagger}$ | $0.68^{*\dagger}$ |

language model generated from all the documents of a candidate to estimate the likelihood of generating the query. The second approach (using document models, *Model 2*) estimate the probability of generating the query for each document, and then based on the likelihood of having a document given a candidate to estimate the expertise of the candidate. The authors found that *Model 2* performs better than *Model 1*.

We compare our best ranking model *UEBER* against the two expert finding models in [7]. In its original setting, their problem was to identify experts from a population of candidates, based on documents about the expert. Applied to our context, the "candidates" are entities, and "documents" are reviews. We compare to both their candidate-centric model (*Model 1*, denoted as *Balog-M1*) and document-centric model (*Model 2*, denoted as *Balog-M2*). After tuning the language model smoothing parameters, we choose the value of 0.1 for both baseline models as they achieve the best performance with this value.

Table 6.7 shows the comparison results. The two models are approximately equivalent for one-term queries, and most of the queries in our evaluation data are one-term queries. That is why as you can see in Table 6.7, *Balog-M1*'s evaluation results and *Balog-M2*'s look identical in two decimal number. *UEBER* significantly outperforms *Balog-M1* and *Balog-M2* in terms of both *precision* and *NDCG* for all the values of top rankings. The two models from [7] are only based on the relevance of the reviews' content and the query to form the rankings, whereas, in addition to the relevance, *UEBER* also uses the information of how supportive each review is to the entity (i.e., rating) and

Table 6.8: Compare *UEBER* against Opinion-Based Entity Ranking Model. $^{*}$,$^{\dagger}$ indicate statistically significant improvements over *OBER-NoExp*$^{*}$ and *OBER-Exp*$^{\dagger}$, respectively

| | *Precision@K* | | | | *NDCG@K* | | | |
|---|---|---|---|---|---|---|---|---|
| | @5 | @10 | @20 | @30 | @5 | @10 | @20 | @30 |
| *OBER-NoExp* | $0.41^{\dagger}$ | $0.33^{\dagger}$ | $0.26^{\dagger}$ | $0.22^{\dagger}$ | $0.31^{\dagger}$ | $0.35^{\dagger}$ | $0.40^{\dagger}$ | $0.43^{\dagger}$ |
| *OBER-Exp* | 0.04 | 0.03 | 0.03 | 0.03 | 0.03 | 0.03 | 0.05 | 0.05 |
| *UEBER* | $0.65^{*\dagger}$ | $0.53^{*\dagger}$ | $0.39^{*\dagger}$ | $0.31^{*\dagger}$ | $0.57^{*\dagger}$ | $0.61^{*\dagger}$ | $0.65^{*\dagger}$ | $0.68^{*\dagger}$ |

more importantly, the expertise of the users behind the reviews. That is why *UEBER* could achieve significantly better ranking results than the two expert finding models.

### Comparison to Opinion-Based Entity Ranking Models

For opinion-based entity ranking (*OBER*) model, we compare our best ranking model *UEBER* to the approach proposed by [53], in which they apply two extensions to standard retrieval models as base models: *query aspect modeling* and *opinion expansion.* The authors showed that highest performance was achieved with *BM25* as the base model. Since all the queries used in our evaluations are single-aspect queries, we do not apply query aspect modeling. For opinion expansion, we compare to two versions of *OBER*:

- No expansion (*OBER-NoExp*): no expansion, the original queries are used.[5]

- With expansion (*OBER-Exp*): each query is expanded by adding all the praise words and all the intensifiers as listed in [53].

Table 6.8 shows that *UEBER* significantly outperforms *OBER* both with and without expansion. Surprisingly, *OBER-Exp* (with expansion) is significantly worse than *OBER-NoExp* (no expansion). Expanding queries with praise words and intensifiers was shown to be helpful for the ranking task in “car” and “hotel” domains [53], but in “restaurant” domain, it is reversed.

[5]This effectively reduces to BM25 on concatenated reviews as documents, which is mentioned as the naive approach in the introduction.

This could be due to the lists of praise words and intensifiers used. To examine this, we removed praise words and intensifiers that are less likely to be used for restaurant aspects, such as "dearly" (an intensifier), "stupendous" (a praise word). We also varied the number of praise words and intensifiers to be added (the smallest value is 1). But *OBER-Exp* was always worse than *OBER-NoExp*. We investigate the issue further by carefully looking at the ranking outputs of *OBER-Exp* and see that the praise words and intensifiers dominated the original aspect keyword(s) in computing the ranking scores.

These comparisons show that user expertise helps to significantly improve the ranking performance over those based on only reviews as it guides the model to rely more on high quality reviews. In the next section, we present a case study that showcases how user expertise helps to include relevant entities, exclude irrelevant ones and improve the rankings.

## 6.3 Case Study

To gain some insight on the workings of the proposed model, we describe a case study involving the query "bar" in Las Vegas. For this case study, we include both our proposed models *ReBER* and *UEBER* to examine the contribution of user expertise in the latter. We also include the strongest baseline *Balog-M2*, which outperforms the other baseline *OBER*. Table 6.9 summarizes the top-15 results for each model, as well as those of the ground truth. The smaller a ranking value is, the higher the rank. For example, the top-ranked "The Griffin" in the ground-truth list was ranked at the 8th by *Balog-M2*, 7th by *ReBER*, and 3rd by *UEBER* respectively. Ties are given the same ranking value.

The greater the agreement between a model's ranking list and the ground-truth's, the better the model is. A greyed cell means that the restaurant does not appear in the top-15 of a particular model's ranking list. Observing the

Table 6.9: Case Study for Query "*Bar*" (in Las Vegas): Rankings for the Restaurants in the Ground-Truth and in the Top 15 from the Outputs of *Balog-M2*, *ReBER* and *UEBER* (The smaller value, the higher ranked)

| *Restaurant Name* | *Ground Truth* | *Balog Model2* | *ReBER* | *UEBER* |
|---|---|---|---|---|
| The Griffin | 1 | 8 | 7 | 3 |
| Frankie's Tiki Room | 1 | 5 | 4 | 2 |
| Commonwealth | 3 | | | 11 |
| Downtown Cocktail Room | 3 | | | |
| Vanguard Lounge | 3 | | | |
| Beauty Bar | 6 | | | |
| Dino's Lounge | 7 | | | 6 |
| Atomic Liquors | 7 | | | 7 |
| Insert Coins | 7 | | | 9 |
| The Chandelier | 10 | 2 | 2 | 1 |
| Gold Spike | 10 | | | |
| Double Down Saloon | 12 | 6 | 6 | 4 |
| Money Plays | 12 | | | |
| Freakin' Frog | 12 | | | |
| Herbs & Rye | 12 | | | |
| Vesper Bar | 16 | | | 5 |
| Velveteen Rabbit | 16 | | | 14 |
| Mandarin Bar | 16 | | | 13 |
| McMullan's Irish Pub | 16 | | | 15 |
| The Lady Silvia | 16 | | | |
| Bar+Bistro @ The Arts Factory | 16 | | | |
| Davy's Locker | 16 | | | |
| Dispensary Lounge | 16 | | | |
| Huntridge Tavern | 16 | | | |
| Hogs and Heifers | 16 | | | |
| Hard Hat Lounge | 16 | | | |
| Crown & Anchor British Pub | 16 | | | |
| Decatur Package Liquor & Cocktails | 16 | | | |
| Aces & Ales | 16 | | | |
| Park on Fremont | 16 | | | |
| Burger Bar | | 1 | 1 | 8 |
| Oyster Bar | | 3 | 3 | 10 |
| Wicked Spoon | | 4 | 5 | |
| Bacchanal Buffet | | 7 | 9 | |
| Mon Ami Gabi | | 9 | 11 | |
| Fogo de Chao Brazilian Steakhouse | | 10 | 8 | |
| XS Nightclub | | 11 | 10 | |
| Minus 5 Ice Lounge | | 12 | 14 | |
| Ghostbar | | 13 | 13 | |
| Gordon Ramsay BurGR | | 14 | | |
| Toby Keith's I Love This Bar & Grill | | 15 | | |
| The Golden Tiki | | | 12 | |
| Born And Raised | | | 15 | 12 |

upper half of Table 6.9, where the ground-truth restaurants appear, we notice that *UEBER* produces the most number of items, with the fewest greyed cells among the models. For example, restaurant "Commonwealth" is ranked at the top-3 in the ground-truth and top-11 in *UEBER*, but it was not included in the top-15 of either *Balog-M2* and *ReBER*. In turn, observing the lower half of Table 6.9, where the ground-truth restaurants do not appear, we see many top-15 restaurants from *Balog-M2* and *ReBER*.

We hypothesize that *user expertise* plays an important role in the improvement of *UEBER* over the other two models. To examine the hypothesis, we look into the log files where we recorded the "contribution" of each review and the corresponding user expertise to the final aptness value $P(e|q)$.

From the log files we observe that user expertise could help to lower the rankings of irrelevant entities. Let us take "Wicked Spoon", a buffet restaurant, as an example. Wicked Spoon is not included in the ground truth but it is included in the top-15 of the ranking lists from *Balog-M2* and *ReBER*. The reason could be that Wicked Spoon has many reviews (more than 4000 reviews) and many of them have a certain level of textual relevance to the query "*bar*", e.g., dessert bar, noodles bar, etc. Nevertheless, *UEBER* is able to exclude Wicked Spoon from its top-15 ranking[6] because many of the relevant reviews were written by users with low expertise. Table 6.10 shows the information of the reviewers who wrote the most relevant reviews to the query "bar". All four users who wrote the 4 most relevant reviews for Wicked Spoon do not have many reviews (less than 100 reviews), and have very few friends. The low expertise of non-expert users weighed down these relevant reviews, decreasing their impact on estimating the aptness of the restaurant.

We also observe that user expertise could help to increase the ranking for relevant entities. For example, "Dino's Lounge" is a dive bar in the ground truth, but is not included in the top-15 of *Balog-M2* and *ReBER*. With user

[6]Wicked Spoon is ranked very low, at the 55th position, in *UEBER*'s ranking for query "Bar" (Las Vegas).

Table 6.10: Case Study. Authors of the Top 4 Relevant Reviews for Query "*Bar*" (in Las Vegas).

| *Restaurant* | *User* | *#Reviews* | *#Friends* |
|---|---|---|---|
| Wicked Spoon | Coolit | 76 | 1 |
| | Ines | 33 | 0 |
| | Cindy | 41 | 0 |
| | Kaila | 7 | 2 |
| Dino's Lounge | EJay | 346 | 42 |
| | Our | 102 | 0 |
| | Jason | 231 | 45 |
| | David | 297 | 571 |

expertise information, *UEBER* is able to include Dino's Lounge in the top-6. As seen in Table 6.10, 3 of the 4 users who wrote the 4 most relevant reviews have hundreds of reviews and they also have good social standing (except for "Our").

## 6.4 Discussion

In this chapter, we consider the problem of ranking review-based entities. Our first model *ReBER* considers the contributions of each review based on its relevance and degree of support. Our second model *UEBER* builds on that, and introduces the intuition that reviews written by "reviewer experts" for a particular query should be considered more important when constructing the ranking for the query. We estimate the expertise of a reviewer based on her productivity and social connections, as well as her knowledge about the topic of interest.

There are parallels, as well as notable differences between expert finding and the entity ranking problem we addressed in this work. If we consider each entity as a candidate expert, and the aptness of an entity as the expertise level of a candidate, then it may be possible to use expert finding as an approach to solve the entity ranking problem. However, there are major differences between the two problems that motivate the development of our methodologies. For expert finding, the documents "representative" of a candidate are usually generated by the candidates themselves (e.g., papers, resumes, emails). Hence, they are presumed to connote positively. For review-based entity ranking, the documents are reviews, and they are not written by the entities themselves; rather they are written by the reviewers, potentially expressing not just positive, but also negative evaluation of the entity. To adjudicate between diverse evaluations, we differentiate reviewers based on their "reviewer expertise", which is a distinctive concept to that in expert finding.

Experiments on Yelp dataset showcase the impact of user expertise on addressing the entity ranking problem. Our user-based entity ranking model, *UEBER*, significantly outperforms the review-based model and the baseline approaches based on expert finding and opinion-based entity ranking.

# Chapter 7

# Conclusions and Future Work

## 7.1 Conclusions

In this dissertation, we present our research on summarizing for a single entity or multiple entities. We show how to incorporate two different sources of review content: reviews and micro-reviews for the task of single-entity summarization.

In Chapter 3, we deal with the problem of review selection for summarizing micro-reviews of an entity. The objective is based on the coverage of micro-reviews with efficiency constraint. We describe an optimal algorithm based on Integer Linear Programming (ILP). Since the problem is NP-hard, we design a greedy algorithm for selecting reviews. Evaluation over a corpus of restaurants' reviews and micro-reviews shows that our approach outperforms the baselines in discovering review sets consisting of compact, yet informative reviews.

In Chapter 4, instead of selecting reviews, we summarize micro-reviews by synthesize a review that is representative of the micro-reviews, yet compact and readable. We formulate the problem based on the Minimum Description Length (MDL) framework as to balance the conflicting objectives of representativeness and compactness holistically. 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.

In Chapter 5, we work on the problem of multi-entity summarization, in which, we synthesize micro-reviews for a collection of entities, using the tips of the underlying entities. To identify the common-popular aspects of the entities, we model the problem as maximal multi-entity quasi-cliques. 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.

In Chapter 6, we deal with the problem of ranking entities based on reviews and reviewer expertise. We propose two ranking models that are formulated based on probabilistic language modeling techniques. The common idea is to rank entities based on the probability that an entity is apt for the query. The difference is how each model derives the probability. In the first model (*ReBER*), the entity probability is decomposed into review-level probabilities that are formulated based on signals from reviews only, such as how popular an entity is, how relevant a review is to the query, and how supportive a review is to the entity. In the second model (*UEBER*), motivated by the observation that reviews written by different users are different in quality, the entity probability is decomposed into user-level probabilities, to further incorporate users' expertise when generating the rankings. Each review is now weighted by how likely its author is an "expert" with regards to the query. The expertise of a user is estimated based on how authoritative the user is in general (e.g., having many (expert) friends, having written many reviews) and how knowledgeable the user is to the given query (e.g., having written many reviews about the query). The experiments on Yelp reviews showcase the impact of *user expertise* on improving the ranking performance. Both authoritativeness and knowledgeability factors are shown to be useful for estimating user expertise. The proposed user-based entity ranking model, *UEBER*, benefits the most when incorporating both of them.

## 7.2 Future Work

There are several potential future research directions on mining reviews and micro-reviews that would further improve the performance and applications of the current works.

*Review synthesis: ordering selected snippets.* For the work of review synthesis (Chapter 4), we have not considered how to order the selected snippets to make the synthesized review to look as "human-created" as possible. There might be some patterns in how reviewers position aspects (i.e., topics) in their reviews. If those patterns could be learnt, we could apply to order the selected snippets. One potential approach is to learn how reviewers position the aspects (topics) in their reviews. For example, we could treat each review as a permutation of topics and learn the permutation distribution (e.g., Mallows model [99]). The ordering of the selected snippets is then created by maximizing the probability given the model. Another approach is to select the order for the selected snippets such that it minimizes the *dynamic time warping* [12] distance to all the existing reviews in the collection.

*Update summarization.* So far, the summaries are generated using a static set of reviews. However, the corpora of reviews are continuously added new reviews, it could be useful to solve the problem of automatically updating summaries for an entity, or a set of entities. One possible approach is to base on mining the graph representation of all the reviews (e.g., [159]). Every time when there are new reviews, instead of updating the whole graph, we only need to update the relevant sub-graph and re-generate the summary.

*Extracting micro-reviews from Twitter.* Twitter could be adopted as a source for extracting micro-reviews as it provides micro-blogging services that allow users to check-in and post length-constrained tweets, which could contain opinions. However, extracting micro-reviews from tweets is not trivia as tweets are mixing various type of content. One potential approach is to formulate the problem as a binary classification problem. We could use micro-reviews from Foursquare as the positive labels and use tweets from news organizations (e.g., BBC, NBC News, CNN) Twitter accounts (who usually post objective contents) as negative labels. There are various types of features that could be used, e.g., lexicon-based (does the tweet contain opinion words) or whether the tweet contains check-in information.

# Bibliography

[1] James Abello, Mauricio GC Resende, and Sandra Sudarsky. Massive quasi-clique detection. In *Latin American Symposium on Theoretical Informatics*, pages 598–612. Springer, 2002.

[2] Anupama Aggarwal, Jussara Almeida, and Ponnurangam Kumaraguru. Detection of spam tipping behaviour on foursquare. In *Proceedings of the 22nd International Conference on World Wide Web*, pages 641–648. ACM, 2013.

[3] EA Akkoyunlu. The enumeration of maximal cliques of large graphs. *SIAM Journal on Computing*, 2(1):1–6, 1973.

[4] Nikolay Archak, Anindya Ghose, and Panagiotis G Ipeirotis. Deriving the pricing power of product features by mining consumer reviews. *Management science*, 57(8):1485–1509, 2011.

[5] Rachit Arora and Balaraman Ravindran. Latent dirichlet allocation based multi-document summarization. In *Proceedings of the second workshop on Analytics for noisy unstructured text data*, pages 91–97. ACM, 2008.

[6] Stefano Baccianella, Andrea Esuli, and Fabrizio Sebastiani. Sentiwordnet 3.0: An enhanced lexical resource for sentiment analysis and opinion mining. In *LREC*, volume 10, pages 2200–2204, 2010.

[7] Krisztian Balog, Leif Azzopardi, and Maarten De Rijke. Formal models for expert finding in enterprise corpora. In *Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 43–50. ACM, 2006.

[8] Krisztian Balog, Yi Fang, Maarten de Rijke, Pavel Serdyukov, and Luo Si. Expertise retrieval. *Foundations and Trends® in Information Retrieval*, 6(2–3):127–256, 2012.

[9] Regina Barzilay, Kathleen R McKeown, and Michael Elhadad. Information fusion in the context of multi-document summarization. In *Proceedings of the 37th annual meeting of the Association for Computational Linguistics on Computational Linguistics*, pages 550–557. Association for Computational Linguistics, 1999.

[10] Shariq Bashir. Ranking entities on the basis of users opinions. *Multimedia Tools and Applications*, 76(1):59–81, 2017.

[11] Shariq Bashir, Wasif Afzal, and Abdul Rauf Baig. Opinion-based entity ranking using learning to rank. *Applied Soft Computing*, 38:151–163, 2016.

[12] Donald J Berndt and James Clifford. Using dynamic time warping to find patterns in time series. In *KDD workshop*, volume 10, pages 359–370. Seattle, WA, 1994.

[13] Lidong Bing, Piji Li, Yi Liao, Wai Lam, Weiwei Guo, and Rebecca J Passonneau. Abstractive multi-document summarization via phrase selection and merging. *arXiv preprint arXiv:1506.01597*, 2015.

[14] Eivind Bjørkelund, Thomas H Burnett, and Kjetil Nørvåg. A study of opinion mining and visualization of hotel reviews. In *Proceedings of the 14th International Conference on Information Integration and Web-based Applications & Services*, pages 229–238. ACM, 2012.

[15] David M Blei, Thomas L Griffiths, and Michael I Jordan. The nested chinese restaurant process and bayesian nonparametric inference of topic hierarchies. *Journal of the ACM (JACM)*, 57(2):7, 2010.

[16] David M Blei, Andrew Y Ng, and Michael I Jordan. Latent dirichlet allocation. *Journal of Machine Learning Research*, 3(Jan):993–1022, 2003.

[17] Petko Bogdanov, Ben Baumer, Prithwish Basu, Amotz Bar-Noy, and Ambuj 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.

[18] Leo Breiman. Random forests. *Machine learning*, 45(1):5–32, 2001.

[19] Mauro Brunato, Holger H Hoos, and Roberto Battiti. On effectively finding maximal quasi-cliques in graphs. In *International Conference on Learning and Intelligent Optimization*, pages 41–55. Springer, 2007.

[20] Ercan Canhasi and Igor Kononenko. Weighted archetypal analysis of the multi-element graph for query-focused multi-document summarization. *Expert Systems with Applications*, 41(2):535–543, 2014.

[21] Qing Cao, Wenjing Duan, and Qiwei Gan. Exploring determinants of voting for the helpfulness of online user reviews: A text mining approach. *Decision Support Systems*, 50(2):511–521, 2011.

[22] Ziqiang Cao, Furu Wei, Li Dong, Sujian Li, and Ming Zhou. Ranking with recursive neural networks and its application to multi-document summarization. In *AAAI*, pages 2153–2159, 2015.

[23] Jaime Carbonell and Jade Goldstein. The use of mmr, diversity-based reranking for reordering documents and producing summaries. In *Proceedings of the 21st annual international ACM SIGIR conference on Research and development in information retrieval*, pages 335–336. ACM, 1998.

[24] Giuseppe Carenini, Jackie Chi Kit Cheung, and Adam Pauls. Multi-document summarization of evaluative text. *Computational Intelligence*, 29(4):545–576, 2013.

[25] Robert D. Carr, Srinivas Doddi, Goran Konjevod, and Madhav V. Marathe. On the red-blue set cover problem. In David B. Shmoys, editor, *SODA*, pages 345–353. ACM/SIAM, 2000.

[26] Asli Celikyilmaz and Dilek Hakkani-Tur. A hybrid hierarchical model for multi-document summarization. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 815–824. Association for Computational Linguistics, 2010.

[27] Deepayan Chakrabarti, Spiros Papadimitriou, Dharmendra S Modha, and Christos Faloutsos. Fully automatic cross-associations. In *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 79–88. ACM, 2004.

[28] Pimwadee Chaovalit and Lina Zhou. Movie review mining: A comparison between supervised and unsupervised classification approaches. In *System Sciences, 2005. HICSS'05. Proceedings of the 38th Annual Hawaii International Conference on*, pages 112c–112c. IEEE, 2005.

[29] Li Chen, Luole Qi, and Feng Wang. Comparison of feature-level learning methods for mining online consumer reviews. *Expert Systems with Applications*, 39(10):9588–9601, 2012.

[30] Ning Chen, Jialiu Lin, Steven CH Hoi, Xiaokui Xiao, and Boshen Zhang. Ar-miner: mining informative reviews for developers from mobile app marketplace. In *Proceedings of the 36th International Conference on Software Engineering*, pages 767–778. ACM, 2014.

[31] Chen Cheng, Haiqin Yang, Irwin King, and Michael R Lyu. Fused matrix factorization with geographical and social influence in location-based social networks. In *Aaai*, volume 12, pages 17–23, 2012.

[32] Judith A Chevalier and Dina Mayzlin. The effect of word of mouth on sales: Online book reviews. *Journal of marketing research*, 43(3):345–354, 2006.

[33] Wen-Haw Chong, Bing Tian Dai, and Ee-Peng 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.

[34] Wen-Haw Chong, Bing-Tian Dai, and Ee-Peng Lim. Prediction of venues in foursquare using flipped topic models. In *European Conference on Information Retrieval*, pages 623–634. Springer, 2015.

[35] Janara Christensen, Stephen Soderland Mausam, Stephen Soderland, and Oren Etzioni. Towards coherent multi-document summarization. In *HLT-NAACL*, pages 1163–1173, 2013.

[36] Kunuch Chutmongkolporn, Bundit Manaskasemsak, and Arnon Rungsawang. Graph-based opinion entity ranking in customer reviews. In *Communications and Information Technologies (ISCIT), 2015 15th International Symposium on*, pages 161–164. IEEE, 2015.

[37] Reuven Cohen and Liran Katzir. The generalized maximum coverage problem. *Information Processing Letters*, 108(1):15–22, 2008.

[38] Gérard Cornuéjols, George L Nemhauser, and Lairemce A Wolsey. The uncapacitated facility location problem. Technical report, Defense Technical Information Center (DTIC) Document, 1983.

[39] Helen Costa, Fabricio Benevenuto, and Luiz HC Merschmann. Detecting tip spam in location-based social networks. In *Proceedings of the 28th*

*Annual ACM Symposium on Applied Computing*, pages 724–729. ACM, 2013.

[40] Thomas M Cover and Joy A Thomas. *Elements of information theory.* John Wiley & Sons, 2012.

[41] Kushal Dave, Steve Lawrence, and David M Pennock. Mining the peanut gallery: Opinion extraction and semantic classification of product reviews. In *Proceedings of the 12th international conference on World Wide Web*, pages 519–528. ACM, 2003.

[42] Milind Dawande, Pinar Keskinocak, Jayashankar M Swaminathan, and Sridhar Tayur. On bipartite and multipartite clique problems. *Journal of Algorithms*, 41(2):388–403, 2001.

[43] Jean-Yves Delort and Enrique Alfonseca. Dualsum: a topic-model based approach for update summarization. In *Proceedings of the 13th Conference of the European Chapter of the Association for Computational Linguistics*, pages 214–223. Association for Computational Linguistics, 2012.

[44] Xiaowen Ding and Bing Liu. The utility of linguistic rules in opinion mining. In *Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 811–812. ACM, 2007.

[45] Xiaowen Ding, Bing Liu, and Philip S Yu. A holistic lexicon-based approach to opinion mining. In *Proceedings of the 2008 international conference on web search and data mining*, pages 231–240. ACM, 2008.

[46] Günes Erkan and Dragomir R Radev. Lexrank: Graph-based lexical centrality as salience in text summarization. *Journal of Artificial Intelligence Research*, 22:457–479, 2004.

[47] Oren Etzioni, Michael Cafarella, Doug Downey, Ana-Maria Popescu, Tal Shaked, Stephen Soderland, Daniel S Weld, and Alexander Yates. Unsupervised named-entity extraction from the web: An experimental study. *Artificial intelligence*, 165(1):91–134, 2005.

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

[49] Gregory Ference, Mao Ye, and Wang-Chien 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.

[50] Rafael Ferreira, Luciano de Souza Cabral, Frederico Freitas, Rafael Dueire Lins, Gabriel de França Silva, Steven J Simske, and Luciano Favaro. A multi-document summarization system based on statistics and linguistic treatment. *Expert Systems with Applications*, 41(13):5780–5787, 2014.

[51] Katja 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.

[52] Douglas H Fisher. Knowledge acquisition via incremental conceptual clustering. *Machine learning*, 2(2):139–172, 1987.

[53] Kavita Ganesan and Chengxiang Zhai. Opinion-based entity ranking. *Information retrieval*, 15(2):116–150, 2012.

[54] Kavita Ganesan, ChengXiang Zhai, and Jiawei Han. Opinosis: a graph-based approach to abstractive summarization of highly redundant opinions. In *Proceedings of the 23rd International Conference on Compu-*

*tational Linguistics*, pages 340–348. Association for Computational Linguistics, 2010.

[55] Kavita Ganesan, ChengXiang Zhai, and Evelyne Viegas. Micropinion generation: an unsupervised approach to generating ultra-concise summaries of opinions. In *Proceedings of the 21st international conference on World Wide Web*, pages 869–878. ACM, 2012.

[56] Huiji Gao, Jiliang Tang, Xia Hu, and Huan Liu. Exploring temporal effects for location recommendation on location-based social networks. In *Proceedings of the 7th ACM conference on Recommender systems*, pages 93–100. ACM, 2013.

[57] Anindya Ghose and Panagiotis G Ipeirotis. Designing novel review ranking systems: predicting the usefulness and impact of reviews. In *Proceedings of the ninth international conference on Electronic commerce*, pages 303–310. ACM, 2007.

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

[59] Goran Glavaš and Jan Šnajder. Event graphs for information retrieval and multi-document summarization. *Expert systems with applications*, 41(15):6904–6916, 2014.

[60] Jade Goldstein, Vibhu Mittal, Jaime Carbonell, and Mark Kantrowitz. Multi-document summarization by sentence extraction. In *Proceedings of the 2000 NAACL-ANLPWorkshop on Automatic summarization-Volume 4*, pages 40–48. Association for Computational Linguistics, 2000.

[61] Panagiotis Gourgaris, Andreas Kanavos, Christos Makris, and Georgios Perrakis. Review-based entity-ranking refinement. In *WEBIST*, pages 402–410, 2015.

[62] Thomas L Griffiths, Michael I Jordan, Joshua B Tenenbaum, and David M Blei. Hierarchical topic models and the nested chinese restaurant process. In *Advances in neural information processing systems*, pages 17–24, 2004.

[63] Peter D Grünwald, In Jae Myung, and Mark A Pitt. *Advances in minimum description length: Theory and applications.* MIT Press, 2005.

[64] Aria Haghighi and Lucy Vanderwende. Exploring content models for multi-document summarization. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 362–370. Association for Computational Linguistics, 2009.

[65] Ruifang He, Bing Qin, and Ting Liu. A novel approach to update summarization using evolutionary manifold-ranking and spectral clustering. *Expert Systems with Applications*, 39(3):2375–2384, 2012.

[66] Andrew Hickl, Kirk Roberts, and Finley Lacatusu. Lccs gistexter at duc 2007: Machine reading for update summarization. In *Proc. of DUC*, volume 7, 2007.

[67] Dorit S Hochbaum. Heuristics for the fixed cost median problem. *Mathematical Programming*, 22(1):148–162, 1982.

[68] Kai Hong and Ani Nenkova. Improving the estimation of word importance for news multi-document summarization. In *EACL*, pages 712–721, 2014.

[69] Minqing Hu and Bing Liu. Mining and summarizing customer reviews. In *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 168–177. ACM, 2004.

[70] Minqing Hu and Bing Liu. Mining opinion features in customer reviews. In *AAAI*, volume 4, pages 755–760, 2004.

[71] Xiao Hu, J Stephen Downie, Kris West, and Andreas F Ehmann. Mining music reviews: Promising preliminary results. In *ISMIR*, pages 536–539, 2005.

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

[73] Bernard J Jansen, Mimi Zhang, Kate Sobel, and Abdur Chowdury. Twitter power: Tweets as electronic word of mouth. *JASIST*, 60(11), 2009.

[74] Heng Ji, Benoit Favre, Wen-Pin Lin, Dan Gillick, Dilek Hakkani-Tur, and Ralph Grishman. Open-domain multi-document summarization via information extraction: Challenges and prospects. In *Multi-source, multilingual information extraction and summarization*, pages 177–201. Springer, 2013.

[75] Daxin Jiang and Jian Pei. Mining frequent cross-graph quasi-cliques. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 2(4):16, 2009.

[76] Nitin Jindal and Bing Liu. Mining comparative sentences and relations. In *AAAI*, volume 22, pages 1331–1336, 2006.

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

[78] Samir Khuller, Anna Moss, and Joseph Seffi Naor. The budgeted maximum coverage problem. *Information Processing Letters*, 70(1):39–45, 1999.

[79] Hyun Duk Kim and ChengXiang 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.

[80] Soo-Min Kim, Patrick Pantel, Tim Chklovski, and Marco Pennacchiotti. Automatically assessing review helpfulness. In *Proceedings of the 2006 Conference on empirical methods in natural language processing*, pages 423–430. Association for Computational Linguistics, 2006.

[81] Won Young Kim, Joon Suk Ryu, Kyu Il Kim, and Ung Mo Kim. A method for opinion mining of product reviews using association rules. In *Proceedings of the 2nd International Conference on Interaction Sciences: Information Technology, Culture and Human*, pages 270–274. ACM, 2009.

[82] Scott Kirkpatrick, C Daniel Gelatt, Mario P Vecchi, et al. Optimization by simulated annealing. *Science*, 220(4598):671–680, 1983.

[83] Efthymios Kouloumpis, Theresa Wilson, and Johanna D Moore. Twitter sentiment analysis: The good the bad and the omg! *ICWSM*, 11(538-541):164, 2011.

[84] Yogan Jaya Kumar, Naomie Salim, Albaraa Abuobieda, and Ameer Tawfik Albaham. Multi document summarization based on news components using fuzzy cross-document relations. *Applied Soft Computing*, 21:265–279, 2014.

[85] John Lafferty, Andrew McCallum, and Fernando CN Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. 2001.

[86] Theodoros Lappas, Mark Crovella, and Evimaria 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.

[87] Theodoros Lappas and Dimitrios 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.

[88] Nevena Lazic, Brendan J Frey, and Parham Aarabi. Solving the uncapacitated facility location problem using message passing algorithms. In *AISTATS*, 2010.

[89] Cane Wing-Ki Leung, Stephen Chi-Fai Chan, Fu-Lai Chung, and Grace Ngai. A probabilistic rating inference framework for mining user preferences from reviews. *World Wide Web*, 14(2):187–215, 2011.

[90] Xuan Li, Liang Du, and Yi-Dong Shen. Update summarization via graph-based sentence ranking. *IEEE transactions on Knowledge and Data Engineering*, 25(5):1162–1174, 2013.

[91] Yanhua Li, Moritz Steiner, Limin Wang, Zhi-Li Zhang, and Jie Bao. Exploring venue popularity in foursquare. In *INFOCOM, 2013 Proceedings IEEE*, pages 3357–3362. IEEE, 2013.

[92] Hui Lin and Jeff Bilmes. A class of submodular functions for document summarization. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*, pages 510–520. Association for Computational Linguistics, 2011.

[93] Marina Litvak and Mark Last. Graph-based keyword extraction for single-document summarization. In *Proceedings of the workshop on Multi-source Multilingual Information Extraction and Summarization*, pages 17–24. Association for Computational Linguistics, 2008.

[94] Guimei Liu and Limsoon 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.

[95] Yang Liu, Xiangji Huang, Aijun An, and Xiaohui Yu. Modeling and predicting the helpfulness of online reviews. In *Data mining, 2008. ICDM'08. Eighth IEEE international conference on*, pages 443–452. IEEE, 2008.

[96] Yue Lu, Panayiotis Tsaparas, Alexandros Ntoulas, and Livia Polanyi. Exploiting social context for review quality prediction. In *Proceedings of the 19th International Conference on World Wide Web*, pages 691–700. ACM, 2010.

[97] Yue Lu, ChengXiang Zhai, and Neel Sundaresan. Rated aspect summarization of short comments. In *Proceedings of the 18th international conference on World wide web*, pages 131–140. ACM, 2009.

[98] Christos Makris and Panagiotis Panagopoulos. Improving opinion-based entity ranking. In *WEBIST (2)*, pages 223–230, 2014.

[99] Colin L Mallows. Non-null ranking models. i. *Biometrika*, 44(1/2):114–130, 1957.

[100] Christopher Manning and Dan Klein. Optimization, maxent models, and conditional estimation without magic. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology: Tutorials-Volume 5*, pages 8–8. Association for Computational Linguistics, 2003.

[101] Christopher D Manning, Prabhakar Raghavan, Hinrich Schütze, et al. *Introduction to Information Retrieval*, volume 1. Cambridge University Press Cambridge, 2008.

[102] Christopher D Manning, Hinrich Schütze, et al. *Foundations of statistical natural language processing*, volume 999. MIT Press, 1999.

[103] Edison Marrese-Taylor, Juan D Velásquez, and Felipe Bravo-Marquez. A novel deterministic approach for aspect-based opinion mining in tourism products reviews. *Expert Systems with Applications*, 41(17):7764–7775, 2014.

[104] Andrew Kachites McCallum. Mallet: A machine learning for language toolkit. In *http://mallet.cs.umass.edu*, 2002.

[105] Ryan McDonald. A study of global inference algorithms in multi-document summarization. *Advances in Information Retrieval*, pages 557–564, 2007.

[106] Kathleen R McKeown, Vasileios Hatzivassiloglou, Regina Barzilay, Barry Schiffman, David Evans, and Simone Teufel. Columbia multi document summarization: Approach and evaluation. Technical report, Columbia University New York United States, 2001.

[107] Martha Mendoza, Susana Bonilla, Clara Noguera, Carlos Cobos, and Elizabeth León. Extractive single-document summarization based on genetic operators and guided local search. *Expert Systems with Applications*, 41(9):4158–4169, 2014.

[108] Xinfan Meng and Houfeng Wang. Mining user reviews: from specification to summarization. In *Proceedings of the ACL-IJCNLP 2009 Conference Short Papers*, pages 177–180. Association for Computational Linguistics, 2009.

[109] Rada Mihalcea and Paul 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.

[110] Rada Mihalcea and Paul Tarau. A language independent algorithm for single and multiple document summarization. In *Proceedings of IJCNLP*, volume 5, 2005.

[111] George Miller and Christiane Fellbaum. Wordnet: An electronic lexical database, 1998.

[112] Felipe Moraes, Marisa Vasconcelos, Patrick Prado, Jussara Almeida, and Marcos Gonçalves. Polarity analysis of micro reviews in foursquare. In *Proceedings of the 19th Brazilian symposium on multimedia and the web*, pages 113–120. ACM, 2013.

[113] Felipe Moraes, Marisa Vasconcelos, Patrick Prado, Daniel Dalip, Jussara M Almeida, and Marcos Gonçalves. Polarity detection of foursquare tips. In *International Conference on Social Informatics*, pages 153–162. Springer, 2013.

[114] George L Nemhauser, Laurence A Wolsey, and Marshall L Fisher. An analysis of approximations for maximizing submodular set functions-I. *Mathematical Programming*, 14(1):265–294, 1978.

[115] Thanh-Son Nguyen, Hady W Lauw, and Panayiotis Tsaparas. Using micro-reviews to select an efficient set of reviews. In *Proceedings of the 22nd ACM international conference on Information & Knowledge Management*, pages 1067–1076. ACM, 2013.

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

[117] Thanh-Son Nguyen, Hady W Lauw, and Panayiotis 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.

[118] Thanh-Son Nguyen, Hady W Lauw, and Panayiotis Tsaparas. Micro-review synthesis for multi-entity summarization. *Data Mining and Knowledge Discovery*, pages 1–29, 2017.

[119] Anastasios Noulas, Salvatore Scellato, Cecilia Mascolo, and Massimiliano Pontil. An empirical study of geographic user activity patterns in Foursquare. *International Conference on Weblogs and Social Media*, 11:70–573, 2011.

[120] Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. The pagerank citation ranking: Bringing order to the web. Technical report, Stanford InfoLab, 1999.

[121] Foad Mahdavi Pajouh, Zhuqi Miao, and Balabhaskar Balasundaram. A branch-and-bound approach for maximum quasi-cliques. *Annals of Operations Research*, 216(1):145–161, 2014.

[122] Bo Pang, Lillian Lee, and Shivakumar Vaithyanathan. Thumbs up?: sentiment classification using machine learning techniques. In *Proceedings of the ACL-02 conference on Empirical methods in natural language processing-Volume 10*, pages 79–86. Association for Computational Linguistics, 2002.

[123] Jeffrey Pattillo, Alexander Veremyev, Sergiy Butenko, and Vladimir Boginski. On the maximum quasi-clique problem. *Discrete Applied Mathematics*, 161(1):244–257, 2013.

[124] Michael J Paul, ChengXiang Zhai, and Roxana 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.

[125] David Peleg. Approximation algorithms for the label-cover max and red-blue set cover problems. *Journal of Discrete Algorithms*, 5(1):55–64, 2007.

[126] Filipa Peleja, João Santos, and João Magalhães. Ranking linked-entities in a sentiment graph. In *Web Intelligence (WI) and Intelligent Agent Technologies (IAT), 2014 IEEE/WIC/ACM International Joint Conferences on*, volume 2, pages 118–125. IEEE, 2014.

[127] Maxime Peyrard and Judith Eckle-Kohler. Optimizing an approximation of rouge-a problem-reduction approach to extractive multi-document summarization. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 1825–1836, 2016.

[128] Tatiana Pontes, Marisa Vasconcelos, Jussara Almeida, Ponnurangam Kumaraguru, and Virgilio 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.

[129] Ana-Maria Popescu and Orena Etzioni. Extracting product features and opinions from reviews. In *Natural language processing and text mining*, pages 9–28. Springer, 2007.

[130] Dragomir R. Radev, Hongyan Jing, Małgorzata Styś, and Daniel Tam. Centroid-based summarization of multiple documents. *Information Processing Management*, 40(6):919–938, November 2004.

[131] Jorma Rissanen. Modeling by shortest data description. *Automatica*, 1978.

[132] Alexander M Rush, Sumit Chopra, and Jason Weston. A neural attention model for abstractive sentence summarization. *arXiv preprint arXiv:1509.00685*, 2015.

[133] Mohammad Salehan and Dan J Kim. Predicting the performance of online consumer reviews: A sentiment mining approach to big data analytics. *Decision Support Systems*, 81:30–40, 2016.

[134] Dou Shen, Jian-Tao Sun, Hua Li, Qiang Yang, and Zheng Chen. Document summarization using conditional random fields. In *IJCAI*, volume 7, pages 2862–2867, 2007.

[135] David B Shmoys, Éva Tardos, and Karen 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.

[136] Ruben Sipos and Thorsten 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.

[137] Ion Smeureanu and Cristian Bucur. Applying supervised opinion mining techniques on online user reviews. *Informatica economica*, 16(2):81, 2012.

[138] Gamgarn Somprasertsri and Pattarachai Lalitrojwong. Mining feature-opinion in online customer reviews for opinion summarization. *J. UCS*, 16(6):938–955, 2010.

[139] Huan Sun, Alex Morales, and Xifeng 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.

[140] Jianshu Sun, Chong Long, Xiaoyan Zhu, and Minlie Huang. Mining reviews for product comparison and recommendation. *Polibits*, (39):33–40, 2009.

[141] Ivan Titov and Ryan 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.

[142] Jorge V Tohalino and Diego R Amancio. Extractive multi-document summarization using multilayer networks. *arXiv preprint arXiv:1711.02608*, 2017.

[143] Panayiotis Tsaparas, Alexandros Ntoulas, and Evimaria 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.

[144] Charalampos Tsourakakis, Francesco Bonchi, Aristides Gionis, Francesco Gullo, and Maria 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.

[145] Peter D Turney. Thumbs up or thumbs down?: semantic orientation applied to unsupervised classification of reviews. In *Proceedings of the 40th annual meeting on association for computational linguistics*, pages 417–424. Association for Computational Linguistics, 2002.

[146] Takeaki Uno. An efficient algorithm for solving pseudo clique enumeration problem. *Algorithmica*, 56(1):3–16, 2010.

[147] Ramakrishna Varadarajan and Vagelis Hristidis. A system for query-specific document summarization. In *Proceedings of the 15th ACM inter-*

*national conference on Information and knowledge management*, pages 622–631. ACM, 2006.

[148] Marisa Vasconcelos, Jussara M Almeida, and Marcos André Gonçalves. Predicting the popularity of micro-reviews: A foursquare case study. *Information Sciences*, 325:355–374, 2015.

[149] Marisa Affonso Vasconcelos, Saulo Ricci, Jussara Almeida, Fabrício Benevenuto, and Virgílio Almeida. Tips, dones and todos: uncovering user profiles in foursquare. In *Proceedings of the fifth ACM international conference on Web search and data mining*, pages 653–662. ACM, 2012.

[150] Vijay V Vazirani. *Approximation algorithms*. Springer, 2004.

[151] Xiaojun Wan and Jianwu Yang. Improved affinity graph based multi-document summarization. In *Proceedings of the Human Language Technology Conference of the NAACL, Companion Volume: Short Papers*, pages 181–184. Association for Computational Linguistics, 2006.

[152] Xiaojun Wan and Jianwu Yang. Multi-document summarization using cluster-based link analysis. In *Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval*, pages 299–306. ACM, 2008.

[153] Xiaojun Wan, Jianwu Yang, and Jianguo Xiao. Towards an iterative reinforcement approach for simultaneous document summarization and keyword extraction. In *Association for Computational Linguistics*, volume 7, pages 552–559, 2007.

[154] Dingding Wang and Tao Li. Document update summarization using incremental hierarchical clustering. In *Proceedings of the 19th ACM international conference on Information and knowledge management*, pages 279–288. ACM, 2010.

[155] Dingding Wang, Tao Li, Shenghuo Zhu, and Chris Ding. Multi-document summarization via sentence-level semantic analysis and symmetric matrix factorization. In *Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval*, pages 307–314. ACM, 2008.

[156] Dingding Wang, Shenghuo Zhu, Tao Li, and Yihong Gong. Multi-document summarization using sentence-based topic models. In *Proceedings of the ACL-IJCNLP 2009 Conference Short Papers*, pages 297–300. Association for Computational Linguistics, 2009.

[157] Jia Wang, James Cheng, and Ada Wai-Chee 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.

[158] Lu Wang, Hema Raghavan, Vittorio Castelli, Radu Florian, and Claire Cardie. A sentence compression based framework to query-focused multi-document summarization. *arXiv preprint arXiv:1606.07548*, 2016.

[159] Li Wenjie, Wei Furu, Lu Qin, and He Yanxiang. Pnr 2: ranking sentences with positive and negative reinforcement for query-oriented update summarization. In *Proceedings of the 22nd International Conference on Computational Linguistics-Volume 1*, pages 489–496. Association for Computational Linguistics, 2008.

[160] Kaiquan Xu, Stephen Shaoyi Liao, Jiexun Li, and Yuxia Song. Mining comparative opinions from customer reviews for competitive intelligence. *Decision support systems*, 50(4):743–754, 2011.

[161] Guangbing Yang, Dunwei Wen, Nian-Shing Chen, Erkki Sutinen, et al. A novel contextual topic model for multi-document summarization. *Expert Systems with Applications*, 42(3):1340–1352, 2015.

[162] Mao Ye, Peifeng Yin, and Wang-Chien Lee. Location recommendation for location-based social networks. In *SIGSPATIAL*, 2010.

[163] Qiang Ye, Ziqiong Zhang, and Rob Law. Sentiment classification of online reviews to travel destinations by supervised machine learning approaches. *Expert systems with applications*, 36(3):6527–6535, 2009.

[164] Surender Reddy Yerva, Flavia Adelina Grosan, Alexandru Octavian Tandrau, and Karl Aberer. Tripeneer: User-based travel plan recommendation application. In *7th International AAAI Conference on Weblogs and Social Media*, number EPFL-CONF-185877, 2013.

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

[166] Xiaohui Yu, Yang Liu, Xiangji Huang, and Aijun An. Mining online reviews for predicting sales performance: A case study in the movie domain. *IEEE Transactions on Knowledge and Data engineering*, 24(4):720–734, 2012.

[167] Zhiwen Yu, Yun Feng, Huang Xu, and Xingshe Zhou. Recommending travel packages based on mobile crowdsourced data. *IEEE Communications Magazine*, 52(8):56–62, 2014.

[168] Quan Yuan, Gao Cong, Zongyang Ma, Aixin Sun, and Nadia Magnenat Thalmann. Time-aware point-of-interest recommendation. In *SIGIR*, 2013.

[169] Zhiping Zeng, Jianyong Wang, Lizhu Zhou, and George 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.

[170] Chengxiang Zhai and John Lafferty. A study of smoothing methods for language models applied to information retrieval. *TOIS*, 2004.

[171] ChengXiang Zhai, Atulya Velivelli, and Bei 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.

[172] Kunpeng Zhang, Yu Cheng, Wei-keng Liao, and Alok Choudhary. Mining millions of reviews: a technique to rank products based on importance of reviews. In *Proceedings of the 13th international conference on electronic commerce*, page 12. ACM, 2011.

[173] Kunpeng Zhang, Ramanathan Narayanan, and Alok N Choudhary. Voice of the customers: Mining online customer reviews for product feature-based ranking. *WOSN*, 10:11–11, 2010.

[174] Qi Zhang, Yuanbin Wu, Tao Li, Mitsunori Ogihara, Joseph Johnson, and Xuanjing Huang. Mining product reviews based on shallow dependency parsing. In *Proceedings of the 32nd international ACM SIGIR conference on Research and development in information retrieval*, pages 726–727. ACM, 2009.

[175] Zhu Zhang and Balaji Varadarajan. Utility scoring of product reviews. In *Proceedings of the 15th ACM international conference on Information and knowledge management*, pages 51–57. ACM, 2006.

[176] Li Zhuang, Feng Jing, and Xiao-Yan Zhu. Movie review mining and summarization. In *Proceedings of the 15th ACM international conference on Information and knowledge management*, pages 43–50. ACM, 2006.