Decision Support Systems xxx (2018) xxx–xxx

Contents lists available at ScienceDirect

Decision Support Systems

journal homepage: www.elsevier.com/locate/dss

Unsupervised tip-mining from customer reviews

Di Zhu[a], Theodoros Lappas[a,*], Juheng Zhang[b]

[a] Stevens Institute of Technology, 1 Castle Point Terrace, Hoboken, NJ 07030, USA
[b] University of Massachusetts Lowell, 220 Pawtucket St, Lowell, MA 01854, USA

ARTICLE INFO

Article history:
Received 3 October 2017
Received in revised form 30 December 2017
Accepted 30 January 2018
Available online xxxx

Keywords:
Online reputation
Reviews
Unsupervised learning
Tips

ABSTRACT

In recent years, large review-hosting platforms have extended their functionality to allow their users to submit tips: short pieces of text that deliver valuable insight on a specific aspect of the reviewed business. These tips are meant to serve as a concise source of information that complements the often overwhelming number of customer reviews. Recent work has tackled the problem of automatically generating tips by mining review text. The motivation for this effort is to obtain tips for businesses or business aspects that have been overlooked by users. Another motivating factor is the quality of the user-submitted tips, which often provide trivial or redundant information. Existing tip-mining methods are limited by a reliance on training data, which is unlikely to be available and is also very costly to create for different domains. In this work, we present TipSelector, a completely unsupervised algorithm that delivers high quality-tips without the need for annotated training data. We verify the efficacy of TipSelector via an evaluation that includes real data from the hospitality industry and comparisons with the state-of-the-art. A secondary contribution of our work is a method for automatically evaluating tip-mining algorithms without humans in the loop. As we demonstrate in our experiments, this method can be used to enable large-scale evaluations and complement the user studies that are typically used for this purpose.

1. Introduction

Customer reviews have been established as an integral part of modern e-commerce. Recent studies reveal that 91% of consumers regularly or occasionally read online reviews, while 84% of people trust online reviews as much as personal recommendations [5]. Previous work has repeatedly verified the strong influence of reviews on the sales and success of products across domains [9, 16, 38, 57, 58]. Despite the established benefits, the volume and unstructured format of customer reviews have introduced challenges for consumers who are trying to collect information on competitive options prior to making a purchase decision. In recent years, the number of reviews on products and services has been rapidly increasing to the point where a platform can host hundreds or even thousands of reviews on a single item or business. For instance, the *Flamingo Las Vegas Hotel & Casino* had around 10,000 reviews on TripAdvisor.com in 2011, and the number was over 30,000 reviews as of 2017. Given that it is clearly impossible for a user to process such overwhelming volumes of information, both platforms and researchers have explored various methods for addressing this challenge. Examples include methods for ranking [2, 33, 39, 52], selection [35, 36, 44, 51], and summarization [13, 27, 48, 60] of customer reviews.

In an effort to further help their users make a purchase decision, leading online platforms, such as TripAdvisor and Yelp, have recently extended their functionality by hosting user-submitted tips: short text snippets that deliver useful information on a specific aspect of the reviewed entity. Consider the following examples of hotel tips:

- *The street noise is an issue, as the fire station is about a block away (with engines coming and going).*
- *Whole Foods is in the neighborhood, along with the naval observatory, where the vice president resides.*

Even if such information is hidden within the large volumes of available reviews or somewhere on the hotel's website, it is unlikely that the user will be able to retrieve them. In fact, studies show that 90% of consumers read less than 10 reviews before forming an opinion about a business, while 68% form an opinion by reading just 1–6 reviews [5]. User-submitted tips serve as a concise way to encode such valuable insight, thus facilitating the user's information-processing task.

Despite their benefits, user-submitted tips may overlook key aspects of the reviewed entity and can also suffer from issues that

* Corresponding author.
E-mail addresses: dzhu1@stevens.edu (D. Zhu), tlappas@stevens.edu (T. Lappas), Juheng_Zhang@uml.edu (J. Zhang).

https://doi.org/10.1016/j.dss.2018.01.011
0167-9236/© 2018 Elsevier B.V. All rights reserved.

are common with user generated content (UGC), such as redundancy and low information quality [1, 15, 25]. In addition, as is the case with online reviews and UGC in general, it is likely that the platfrom will receive numerous tips for a small set of popular businesses, while new or less prominent options will be overlooked. In order to address such issues, recent work has introduced a method for automatically extracting tips from customer reviews [25]. The proposed method starts by extracting meaningful templates from a set of pre-existing tips that are assumed (or manually verified) to be informative. After human annotators filter out generic and irrelevant templates, the final template-set is used to locate tip candidates in customer reviews. Human annotators label each candidate as useful or not useful. The annotated tips serve as a training dataset for a classifier, which can then be used to automatically label new candidates. We describe the process in detail in Section 2.

The pioneering method proposed by Guy et al. [25] is the first of its kind and, as its creators demonstrate, can deliver informative tips. However, its applicability is limited by its supervised nature, its reliance on existing tips, and the need for manual annotations. In this work, we overcome such limitations and extend the literature by making the following contributions:

1. We present TipSelector: a completely unsupervised algorithm for automated tip-mining from customer reviews. As we demonstrate in our experiments, TipSelector delivers a low-redundancy set of highly useful tips, as evaluated by human annotators.
2. We describe a method for evaluating the informativeness of a given set of tips without humans in the loop. Our method can be used to complement studies with human annotators, which are currently the only option for evaluating tip-mining algorithms.

The rest of paper is organized as follows. We review the relevant literature in Section 2. In Section 3, we propose an algorithm that extracts informative tips from customer reviews. In Section 4, we verify the efficacy of the proposed algorithm via an evaluation on real data, which includes multiple experiments and competitive baselines. Finally, we conclude in Section 5 with a discussion on managerial implications and a note on future work.

2. Background and related work

Online reviews have been established as the primary source of information for consumers [6, 14, 23, 29, 56]. Previous work has verified that reviews have a strong effect on customer decisions across industries [9, 16, 38, 57, 58]. Despite the established benefits of online reviews, their ever-increasing number has also given rise to challenges. Fake reviews have emerged as a troubling phenomenon that has motivated a significant body of work [28, 30, 37]. Relevant research has also focused on studying information quality in customer reviews [7, 33, 59].

Another relevant challenge is information overload, an inevitable consequence of the hundreds or even thousands of reviews that are nowadays available on a single entity [17, 31, 46]. A number of studies have verified that information overload leads to an increase in the time required to make a decision [50], a decrease in decision quality [10, 49], and lack of confidence about the decision [12]. Research has focused on both conversational overload and information entropy [31]. Conversational overload is the state when an individual fails to react adequately in view of the fact that too many messages are delivered [54], whereas information entropy represents the condition when incoming messages are not systematically arranged [26].

Previous work has explored three different directions toward addressing information overload. The first is review selection, which focuses on selecting a compact and representative subset of reviews [35, 36, 44, 51]. The second direction is review summarization, which aims to report aggregate statistics of negative and positive opinions about different features of the reviewed entity [13, 27, 48, 60]. The third direction is review ranking, which ranks the available reviews according to various measures, such as user-assigned helpfulness votes [2, 33, 39, 52].

In recent years, review-hosting platforms have explored user-submitted tips an alternative mechanism for addressing information overload. Tips are short pieces of text (typically no more than 1–2 sentences long) that deliver valuable insight on a specific aspect of the reviewed entity. Previous work on text mining has focused on extracting tips from sources such as question-answering websites. A relevant effort is that by Weber et al. [53], who define tips as “short, concrete and self-contained bits of non-obvious advice”. The authors proposed a tip-extraction method to answer “how-to” questions. After collecting answers starting with a verb as tip candidates, they conducted a user study to evaluate the quality of each candidate. The annotated tips served as a training set for a classification algorithm, which could then be used to evaluate new candidates. Recently, Guy et al. [25] presented a similarly inspired tip-mining algorithm. We present the pseudocode in Algorithm 1.

Algorithm 1. Baseline tip extractor [25].

```
1: Input: sentence-similarity function $g()$, set of businesses $\mathcal{B}$, set of reviews $\mathcal{R}_b, \forall b \in \mathcal{B}$, set of noisy tips $\mathcal{T}$, frequency threshold $FT$, similarity threshold $ST$.
2: Output: Set of tips $\mathcal{T}_b$ for each business $b \in \mathcal{B}$
3: Manually select a subset of $\mathcal{T}^* \subseteq \mathcal{T}$ of useful tips.
4: Process $\mathcal{T}^*$ to extract a set of n-gram $(4 \leq n \leq 7)$ sequences $\mathcal{W}$ that have at most 1 wildcard and occur at least $FT$ times.
5: Create a final set of templates $\mathcal{W}^*$ by manually eliminating trivial templates from $\mathcal{W}$.
6: Process all available reviews to identify the set of sentences $\mathcal{M}$ that match at least one of the templates in $\mathcal{W}$.
7: Manually annotate a subset $\mathcal{M}^* \subseteq \mathcal{M}$ as useful/not useful.
8: Train a binary usefulness classifier $CL_u$ on $\mathcal{M}^*$
9: for $b \in \mathcal{B}$ do
10:     Use $CL_u$ to identify the set $\mathcal{U}_b$ of useful sentences in $\mathcal{R}_b$.
11:     Let $CL_u(s)$ be the score assigned by $CL_u$ to a sentence $s$.
12:     $\mathcal{T}_b = \{s \in \mathcal{U}_b : \nexists s' \in \mathcal{U}_b \text{ with } g(s,s') \geq ST \text{ and } CL_u(s') > CL_u(s)\}$
13: Return $\{\mathcal{T}_b, \forall b \in \mathcal{B}\}$
```

The first phase of the algorithm focuses on the generation of tip templates. In order to do so, the authors collected a set of user-submitted tips $\mathcal{T}^*$ from 9847 travel guides on 54 different cities from TripAdvisor. Human annotators then identified a subset $\mathcal{T}^* \subseteq \mathcal{T}$ of useful tips. The second phase identifies n-gram sequences of 4–7 words that frequently appear in $\mathcal{T}^*$. These sequences are converted to a set of templates $\mathcal{W}$ by allowing one of the included words to be a wildcard that could be matched to *any* word. Irrelevant and trivial templates are eliminated by hand to produce a final set $\mathcal{W}^*$ of 150 templates.[1] The set of templates was then applied on a large dataset of over 3.5 million TripAdvisor reviews on various points of interest (POIs), such as tourist attractions across the USA. Out of the full set $\mathcal{M}$ of about 415,000 matching sentences, the authors sampled a subset $\mathcal{M}^*$ of 7500. The sentences in $\mathcal{M}^*$ were again annotated by

[1] http://www.ise.bgu.ac.il/downloadMe/templates.txt.

humans as useful or not useful. The annotated subset served as a training corpus for a classifier CL_u, which could then be used to automatically annotate new sentences as useful or not useful tips. The authors complete their approach by ranking the tips according to the score assigned by the classifier and eliminating sentences that are highly similar to another sentence with a higher score.

The supervised nature of the approach by Guy et al. introduces a significant limitation, as appropriate training data are unlikely to be available and certainly costly to create across domains. A second limitation is the need for humans in the loop during the various phases of the algorithm, such as during the template-filtering process. This further limits the method's applicability. The TipSelector algorithm that we describe in this work overcomes such limitations, as it is completely unsupervised and does not require human intervention. Nonetheless, we utilize the approach by Guy et al. as a baseline in our experiments.

Our work also has ties to the literature on detecting "experience revealing" sentences from user-generated text [41, 43, 45, 47]. Min and Park [41] proposed a method for identifying helpful reviews based on mentions about experiences. Park et al. [45] defined experience as knowledge embedded in a collection of undergone activities or events. In their work, the experience-detection task was treated as a classification problem based on features such as verb class, tense, aspect, and mood. While relevant, our tip-mining paradigm differs significantly from this type of work; while it is possible that some of the extracted tips could be described in an experiential manner, this is by no means a requirement.

3. The TipSelector algorithm

In this section we describe the TipSelector algorithm for tip-mining from customer reviews. For the remaining of the discussion, we assume that the reviewed entity is a business. However, the algorithm can be applied to any type of entity attached to a set of customer reviews. We present the full pseudocode of TipSelector in Algorithm 2.

Algorithm 2. TipSelector.

```
1: Input: business-similarity function $sim()$, set of businesses $\mathcal{B}$, set of reviews $\mathcal{R}_b, \forall b \in \mathcal{B}$
2: Output: Set of tips $\mathcal{T}_b$ for each business $b \in \mathcal{B}$
3: for business $b \in \mathcal{B}$ do
4:     $\mathcal{S}_b^m = \operatorname{argmax}_{b' \in \mathcal{B} \setminus b}\, {}_m sim(b, b')$ // The m businesses that maximize sim(b, b')
5:     $N_b = \{\}$ // Dictionary, maps tokens to their frequency
6:     $X_b = \{\}$ // Dictionary, maps tokens to sentences that include them
7:     for sentence $s$ in $\mathcal{R}_b$ do // considers all sentences from all reviews on b
8:         for token $t$ in $s$ do
9:             $X_b[t].append(s)$ // remember the sentences that cover each token
10:            if $t$ in $N_b$ then $N_b[t] += 1$
11:            else $N_b[t] = 1$
12: for business $b \in \mathcal{B}$ do
13:     $\mathcal{I}_b = \{t \in N_b : N_b[t] >> N_{b'}[t], \forall b' \in \mathcal{S}_b^m\}$ // based on Fisher's Test
14:     $\mathcal{T}_b = SetCoverSolver(\mathcal{I}_b, X_b)$
15: Return $\{\mathcal{T}_b, \forall b \in \mathcal{B}\}$
```

The input consists of a business-to-business similarity function *sim*(), a set of focal businesses $\mathcal{B}$, and the set $\mathcal{R}_b$ of all the sentences from all the reviews on $b, \forall b \in \mathcal{B}$. The output includes a set of tips $\mathcal{T}_b$ for each business $b \in \mathcal{B}$.

Line 4: The algorithm processes each business $b \in \mathcal{B}$ independently. In line 4, it computes $\mathcal{S}_b^m$: the set of m most similar business to b, as computed by the *sim*() function. We discuss the value of m in detail in Section 3.1. This set provides a baseline from which we expect the business to deviate in meaningful ways. For instance, when we are trying to mine tips for an expensive, 5-star hotel in a city, we are interested in *information tokens* (pieces of information) that distinguish the hotel from other hotels in the same tier and neighborhood. A comparison with lower-tier or distant options would yield anticipated and, thus, trivial distinctive information (e.g., information on luxury services or on nearby dining and shopping options). TipSelector is compatible with any definition of business similarity. This flexibility is essential, as the concept of similarity is likely to differ across domains. We discuss our own definition of similarity in Section 3.1.

Lines 5–11: In these lines, TipSelector creates and populates two dictionaries N_b and X_b, which will be used to map each information token (i.e. useful piece of information) to its frequency and to the set of sentences in $\mathcal{R}_b$ that include it. The two dictionaries are populated in lines 7–11, in which the algorithm iterates over each token in each of the sentences in $\mathcal{R}_b$. We discuss our definition of information tokens in Section 3.2.

Lines 13–15: In these lines, TipSelector iterates over each business b. In Line 13, it computes $\mathcal{I}_b$: the set of information tokens that have a significantly higher frequency in the reviews of b than in any of its similar business in $\mathcal{S}_b^m$. Let N_b^{tot} be the sum of all the frequencies of all the tokens for a business b. Then, given a token t and two businesses b, b' we construct a 2×2 contingency table T with $A[0][0] = N_b[t]$, $A[0][1] = N_{b'}[t]$, $A[1][0] = N_b^{tot} - N_b[t]$, and $A[1][0] = N_{b'}^{tot} - N_{b'}[t]$. The null hypothesis is then that the token is equally likely to appear in the reviews for both businesses. We can evaluate the hypothesis using Fisher's exact test [21], which is appropriate for such tables. For large sample sizes, the faster χ^2 square test offers a good approximation of Fisher's exact results. In our experiments, we use Fisher's test with $\alpha = 0.05$ to determine if $N_b[t] \gg N_{b'}[t]$. We also exclude the terms included in the name of the business (e.g., *Hilton Arlington*), as their presence in $\mathcal{I}_b$ is anticipated and uninformative. In lines 14–15, TipSelector computes and returns the final set of tips $\mathcal{T}_b$ for each business b. $\mathcal{T}_b$ is defined as a set of sentences that *cover* the distinctive tokens in $\mathcal{I}_b$. We describe our definition of the *SentenceCover*() function in Section 3.3.

3.1. Finding similar businesses

Next, we describe a definition of the business similarity function *sim*() for the hotels domain. We process the hotels from each city separately. This allows us to incorporate the local context that comes with each city (e.g., touristic attractions, dining, shopping, transportation). Within each city, we independently process hotels from different prices levels. This allows us to control for the hotel's tier. Finally, we compare two hotels within the same city-price stratum via their Euclidean distance within a multi-dimensional feature space that includes

- A binary feature (available/not available) for each amenity offered by any of the hotels in the stratum.
- 40 numeric features with values in [0, 1], with each feature representing the prevalence of a different topic within the review corpus of the hotel.

We compute the topic features by using Latent Dirichlet Allocation [4] (LDA) on the entire collection of reviews from all the hotels within the city-price stratum. LDA delivers the percentage of each review that is allocated to each of the 40 topics. The value of a hotel

h for a feature corresponding to topic t is then set to be the mean percentage over all the reviews for h. We set the number of topics K to 40 via the standard perplexity measure [42], after experimenting with $K \in \{10, 20, 30, 40, 50, 60, 70, 80, 90, 100\}$.

Following the evaluation of similarity, we compute the set S_b^m of the m nearest neighbors of each hotel (line 4). An easy way to estimate m would be to set an intuitive lower bound for the similarity function $sim()$, such as 1 or 2 standard deviations over the average similarity of all the business in $\mathcal{B}$. In order to further simplify and automate the process for practitioners, we studied the effect of m on the set of delivered tips. We found that, for all hotels, incrementally increasing the value of m up to a value m_x led to the pruning of information tokens from the distinctive set $\mathcal{I}_b$. This, in turn, resulted in a different set of tips $\mathcal{T}_b$ for each value of m. In contrast, increasing m beyond m_x and up to a value m_y did not alter $\mathcal{I}_b$ and thus led to the same $\mathcal{T}_b$ set. Finally, increasing m beyond m_y changed the set of tips due to the pruning of additional tokens following the introduction of new businesses. However, the similarity of these new businesses to b rapidly decreased as we increased m, leading to the unwanted consideration of distant neighbors. Our analysis revealed that the average length of the $[m_x, m_y]$ interval was around 4.5. We also found that, while the endpoints of the interval varied across hotels, the number 5 was included for 98% of all the hotels. Hence, we set $m = 5$ in our experiments. This type of analysis can help practitioners in cases where finding an intuitive similarity threshold can be a challenge (e.g. for complex similarity functions).

3.2. Mining information tokens

The definition of an information token can be extended to include any linguistic or semantic feature. In this work, we consider four types of tokens:

1. Named Entities (e.g., White House, Johnny Rockets), as extracted by the Stanford Named Entity Recognizer (NER) [20].
2. Compound n-grams (e.g., *ceiling fan*, *toilet paper*, *street noise*). We extract these by applying the dependency parser by Chen & Manning [8] on each sentence of the given texts and identifying terms connected via the "compound" dependency.
3. Adjective-modified nouns (e.g., *great service*, *cheap restaurant*, *soft pillow*). Following the application of the dependency parser, we identify terms of pairs connected via the "amod" dependency.
4. Singleton nouns that are not included in any of the above types and satisfy trivial lower and upper frequency bounds (e.g., *bathroom*, *pool*, *location*). Following best practices, we eliminate nouns that appear in less than 1% and more than 99% of the reviews in our dataset [24]. This excludes uninformative candidates (e.g. spelling errors) and generic terms (e.g. "hotel").

3.3. Selecting sentences

The final step computes a minimal set of sentences that covers all the tokens in $\mathcal{I}_b$. We use $s \prec t$ to denote that a sentence s covers a token t.

- If t is a Named Entity, then $s \prec t$ if s includes t verbatim.
- If t is a single noun, compound n-gram or modifier-noun pair, then $s \prec t$ if s includes any of the tokens in the *extended synset* of t.

We compute the extended synset $esyn(t)$ of t as follows. First, we use WordNet to map each component term of t to its synset [18]. We use the established LESK Algorithm for Word Sense Disambiguation [3] to identify the appropriate Sense (and corresponding synset) of each term. LESK chooses the Sense whose definition (a short text snippet) has the highest number of overlapping words with the context sentence s. We then extend the term's synset with semantically similar terms that appear in s. We use the established formula by Wu & Palmer [55] to evaluate semantic similarity. The formula calculates relatedness by considering the depths of two synsets sn_1, sn_2 in the WordNet taxonomy, along with the depth of their longest common subsumer (lcs), defined as the most specific concept they share as an ancestor. It takes values in $[0, 1]$ and is formally defined as $score = 2 \times depth(lcs(sn_1, sn_2)/(depth(sn_1) + depth(sn_2))$. Guided by previous work [19], we set the threshold to 0.8. This approach, which we evaluate in Section 4.4, allows us to capture non-verbatim but intuitive coverage cases, such as the following: *great view* $\Longleftrightarrow$ *amazing view*, *free meal* $\Longleftrightarrow$ *complimentary meal*, *restroom* $\Longleftrightarrow$ *bathroom*, *metro terminal* $\Longleftrightarrow$ *train station*. Finding a minimal sentence-cover is then equivalent to solving an instance of the Minimum Set Cover Problem [32]. Formally, given the set of tokens to be covered $\mathcal{I}_b$ and the set of candidate sentences $\mathcal{R}_b$, we want to find a subset $\mathcal{R}^* \subseteq \mathcal{R}_b$ such that $\exists s \in \mathcal{R}^* : s - < t, \forall t \in \mathcal{I}_b$ and $|\mathcal{R}^*|$ is minimized.

Even though the problem is NP-complete, we can effectively approximate it via a greedy heuristic that iteratively adds the sentence that covers the maximum number of uncovered tokens [11]. During the computation of $\mathcal{I}_b$, we use a dictionary X_b to map each token to the sentences that include it (defined in line 6). This structure allows us to efficiently compute the minimal sentence cover. In order to speed-up the process and reduce memory requirements, we can ignore sentences that cover exactly the set of tokens covered by another sentence in X_b. In our own implementation, we keep only the longest sentence among those that cover the same set of tokens. This approach can be trivially extended to choose sentences according to alternative criteria, such as readability or the number of helpfulness votes assigned to the review from which the sentence was extracted.

3.4. Qualitative evidence

Table 1 shows examples of tips mined by TipSelector. For each tip, we highlight the information tokens that it includes and mark each token with its type. We use the superscripts A, C, E, and N for Adjective-modified Nouns, Compounds nouns, Named Entities, and singleton Nouns.

4. Evaluation

In this section we evaluate the proposed TipSelector algorithm. In Section 4.5, we introduce a new method for the automatic evaluation of tip-mining algorithms, which can be used to complement the user studies that are typically used for this purpose.

4.1. The TripAdvisor dataset

Our dataset of hotel reviews comes from TripAdvisor.com, one of the largest review platforms and travel portals. The dataset was collected during the last week of January 2015, and includes over 2.8 million reviews on 7625 hotels from 17 cities across the world. Fig. 1 shows the distribution of hotels across cities. The cities with the largest and smallest number of hotels are Paris (1755 hotels) and Philadelphia (93 hotels), respectively. Fig. 2 shows a histogram of the reviews over different ranges. For instance, we observe that 3452 hotels had at most 100 reviews, while 499 hotels had more than 1000. Our dataset also includes the price range and amenities offered by each hotel, as listed on the website. We utilize this information to evaluate the similarity between hotels, as we described in detail in Section 3.1. An archive with the full text of the parsed reviews of each hotel can be found here: https://tinyurl.com/TipSelectorData. The archive also includes the raw landing page for each hotel,

Table 1
Examples of tips generated by TipSelector.

- **Great location**[A], walk to **Whole Foods**[E], **Kavanaughs's**[E] for **good pizza**[A], **shuttle service**[C], **discount**[N] for **American University**[E].
- There were **trees**[N] blocking the view which was fine b/c on weekends the park seemed to fill up with **homeless people**[A]
- What I really like is the **underground parking**[A] available along with a key needed to go up the **elevator**[N].
- There is an awesome **deli**[N] across the street for quick **sandwiches**[N] that we always enjoy.
- Expert guidance by the very professional **concierge**[N] **Peter Gill**[E] added tremendous value to our recent stay at this wonderful hotel.
- We received a complimentary **sleep kit**[C] with **earplugs**[N], an **eye mask**[C], and **lavender spray**[C] for the bedding.
- One caveat, make sure when booking a room that it does not house an **hvac compressor**[C].
- When I called the hotel to make my reservation, they informed me that there is a $500 **refundable deposit**[A] for having a **small dog**[A] in the room.

which includes all information on amenities, price range, and other attributes.

4.2. Baselines

We compare TipSelector with the state-of-the-art for tip mining, recently proposed by Guy et al. [25]. As we discussed in detail in Section 2, a key limitation of this algorithm is that it assumes the availability of a set of tips from the domain of interest. The first phase of the algorithm mines templates from the given set of tips, which are then used to identify new tips from the unstructured text of customer reviews. In our evaluation, we consider following two baselines based on the paper by Guy et al. Our implementation follows the process described in the original paper.

POI-Miner: This is the original implementation by Guy et al. [25], which we described in detail in Section 2.

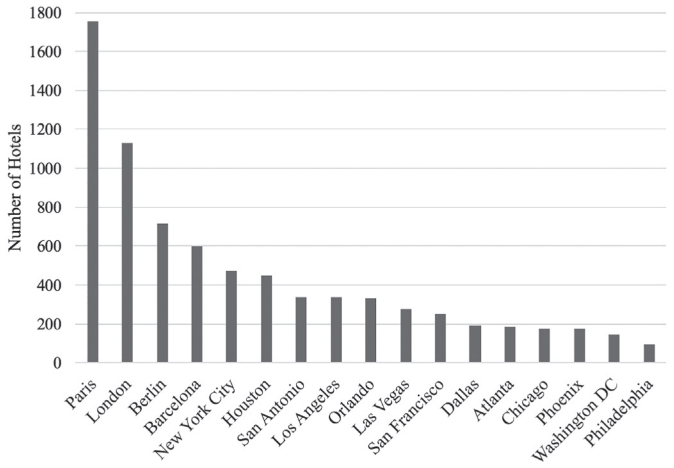

Fig. 1. Number of hotels per city.

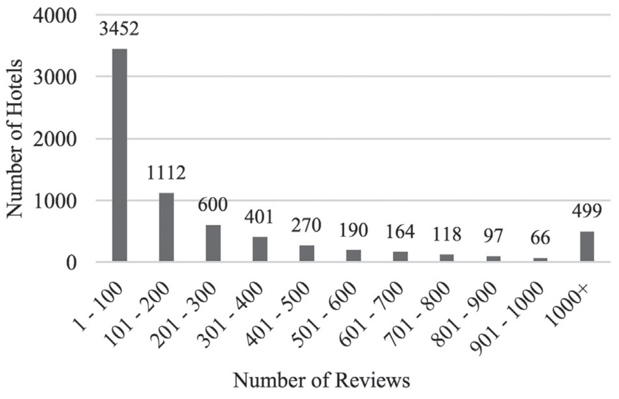

Fig. 2. Number of reviews per hotel.

Hotel-Miner: Given that our dataset includes hotels rather than POIs, we also implement a version of the algorithm by Guy et al. that mines templates from a large set of hotel tips. The set includes tips for the hotels in our TripAdvisor dataset, as well as hotel tips from the openly available Yelp Dataset Challenge Dataset.[2] Including tips from Yelp is necessary, as the hotel tips from TripAdvisor are limited to hotel rooms.

4.3. Evaluating tip usefulness and novelty

Next, we describe the user study that we conducted to evaluate and compare the tips generated by each of the three algorithms (TipSelector, POI-Miner, and Hotel-Miner). First, we use each algorithm to generate tips for a randomly selected set of 5 hotels from the TripAdvisor dataset. Hotel similarity was computed as described in Section 3. We recruited 5 student annotators. In the first phase of the study, the annotators were given access to the dedicated web page for each hotel on TripAdvisor, which includes reviews, user-submitted tips, and information on the hotel's services and amenities. They were given access to the pages for 1 h, with the task of recording as much useful information as they could for each hotel.

In the second phase, the annotators were given the tips generated by each algorithm. During the annotation, we hid the identity and randomized the evaluation order of the algorithms. For each tip, the annotators were asked to assign a usefulness and a novelty rating on a Likert scale from 1 to 5, with higher values representing more favorable ratings. We defined usefulness as the extent to which the tip provides information that might be of use to a person staying at or considering the hotel. We defined novelty as the extent to which the tip offers information that is not covered by either (i) the information collected during the first phase of the study or (ii) the other tips that the algorithm delivered for the hotel.

We present the results for usefulness and novelty in Figs. 3 and 4, respectively. The figures report the mean novelty and usefulness ratings for each hotel. The first observation is that TipSelector consistently outperforms the two baselines in terms of both the novelty and the usefulness of the generated tips, across hotels. As anticipated, Hotel-Miner achieves superior results to POI-Miner, as it employs relevant templates from the hotels domain. It is important to stress that this improvement is not without cost, as the availability of the tips required for template extraction cannot be guaranteed for every domain. In addition, the template-extraction process is not fully automated and requires the manual refinement of candidate templates. On the other hand, TipSelector is free from such limitations while achieving superior results. In Fig. 5, we show the number of tips that each approach generated for each hotel. We observe that the two baselines consistently produce more tips than TipSelector. However, the annotator ratings reveal that these additional tips are either redundant or fail to deliver useful information, and are thus not beneficial for potential customers.

[2] https://www.yelp.com/dataset_challenge/.

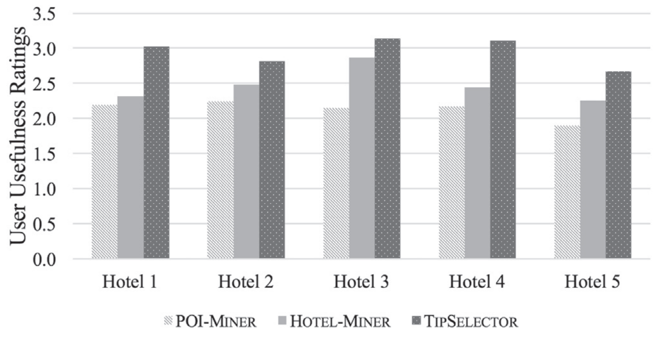

Fig. 3. Usefulness results.

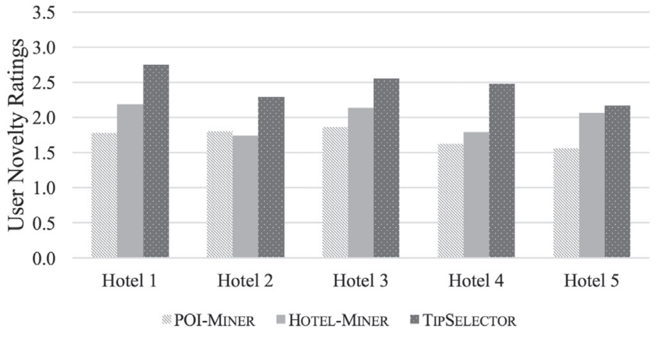

Fig. 4. Novelty results.

Finally, in Fig. 6 we present the Weighted Fleiss' Kappa (WFK) values for each hotel. WFK is a statistical measure for assessing agreement between a fixed number of raters who are tasked with assigning ordinal ratings to multiple items [22,40]. We use the weighted version of the measure which is appropriate for ordinal ratings. We observe that the values were consistently above 0.9, indicating near-perfect agreement among the annotators [34].

4.4. Evaluating sentence-to-token coverage

As we described in Section 3.3, TIPSELECTOR uses WordNet to map the constituent terms of token types 2–4 to their respective synsets. This enables the computation of a minimal set of sentences by using tips that could include alternative forms for each token, consisting of synonyms or semantically similar terms. Our implementation uses a minimum similarity threshold of 0.8. In order to validate that this does not lead to false positives or false negatives and can accurately identify the tokens covered in a sentence, we conduct a user study as follows.

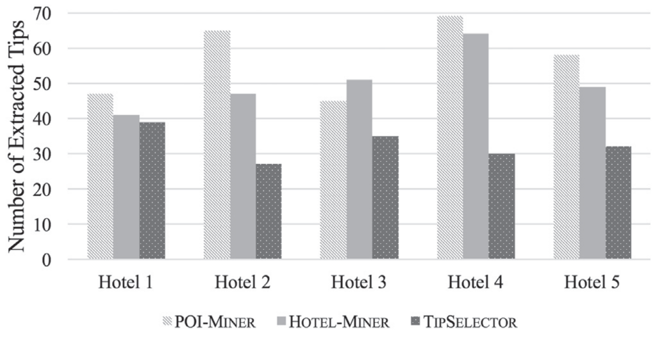

Fig. 5. Number of tips per hotel.

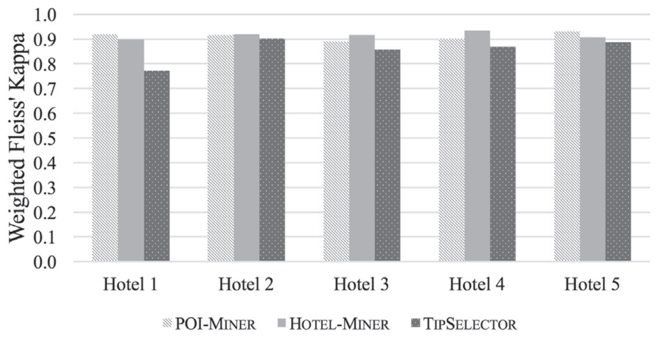

Fig. 6. Fleiss kappa results.

We begin by randomly selecting 50 hotels from the TripAdvisor dataset. For each sampled business b, let $\mathcal{R}_b, \mathcal{I}_b$ be the set of reviews and the set of distinctive information tokens (as determined by TIPSELECTOR), respectively. We then sample a sentence s from the reviews in $\mathcal{R}_b$, such that TIPSELECTOR has determined that s covers exactly K of the tokens in $\mathcal{I}_b$. We repeat the process for $K \in \{0, 1, 2, 3\}$, for a total of $4 \times 50 = 200$ sentences. We do not explore larger values of K, as less than 1.5% of the sentences in our dataset included more than 3 tokens. We focus on a sample of 50 hotels for practical reasons, as it would be infeasible to manually annotate sentences from all 7625 hotels in our dataset. Two human annotators were then tasked with independently processing the sentences and marking the actual distinctive tokens covered by each sentence. For each value of K, we compute the error as $Err = |\mathcal{X} \backslash \mathcal{Y}| + |\mathcal{Y} \backslash \mathcal{X}|$, where $\mathcal{X}$ and $\mathcal{Y}$ are the set of tokens identified by annotators and by TIPSELECTOR, respectively. The error function thus computes the sum of false positives ($|\mathcal{Y} \backslash \mathcal{X}|$) and false negatives ($|\mathcal{X} \backslash \mathcal{Y}|$) of TIPSELECTOR. We only consider sentences for which the annotators were in agreement. We report the results in Table 2.

We observe that, for all values of K, the strong majority of the sentences had an error of zero, meaning that they included exactly the tokens marked by the annotators. This validates our algorithm's ability to select a minimal set of tips that cover the distinctive tokens for a given business. A second observation is that the error is higher for larger values of K. This is intuitive, as the probability of making a mistake is higher for sentences with more tokens. However, the error was never above 3. In fact, it was 3 for only one sentence with $K = 3$, and the number of sentences with $Err = 0$ was consistently over 40 for all values of K. Finally, we note that the number of discarded sentences was trivial. This indicates the straightforward nature of the labeling task for the human annotators and validates the set of ground-truth sentences that we used in this experiment.

4.5. Automatic evaluation of tip-mining algorithms

In Section 4.3 we described a user study that compared the tips produced by our method with those by competitive baselines. Even though human annotations can indeed help evaluate such methods, the necessity of including humans in the loop limits their applicability to a small portion of the available data. In order to address this,

Table 2
Results of the user study on sentence-to-token coverage.

		$Err = 0$	$Err = 1$	$Err = 2$	$Err = 3$	Discarded
# real tokens	$K = 0$	**48**	2	0	0	0
	$K = 1$	**44**	6	0	0	0
	$K = 2$	**44**	3	3	0	1
	$K = 3$	**41**	4	2	1	2

we describe a quantitative measure based on the concept of coverage that we described in Section 3.3 and evaluated in Section 4.4. The measure, which we refer to as *TCov*, can be used to automatically evaluate tip-mining methods without humans in the loop. It is computed as the percentage of the information tokens encoded in a set of tips $\mathcal{T}$ that is covered by a set of reviews $\mathcal{R}$. Formally

$$TCov(\mathcal{T},\mathcal{R}) = \frac{|tokens(\mathcal{R}) \cap tokens(\mathcal{T})|}{|tokens(\mathcal{T})|}$$

Here, $tokens(\mathcal{R})$ returns the union of the tokens covered by all the sentences of all the reviews in $\mathcal{R}$. Given the definition of the *TCov* measure, we can evaluate the informativeness of a set of tips $\mathcal{T}$ in the context of a set of reviews $\mathcal{R}$. Our evaluation is designed to address the following question: *How many reviews from $\mathcal{R}$ would a user have to read in order to cover at least α% the information tokens in $\mathcal{T}$?*

Rather than be limited to a single arbitrary review-processing order, we instead adopt the following randomized design:

1. Assume an arbitrary order of the reviews in $\mathcal{R}$.
2. Shuffle the order and iteratively process the reviews until $TCov(\mathcal{T},\mathcal{R}') \geq \alpha$%, where $\mathcal{R}' \subseteq \mathcal{R}$ is the set of reviews processed so far. Here, α is a coverage parameter that we tune in our experiment.
3. Repeat the second step M times and report the average number of reviews required for coverage (we set $M = 1000$).

This delivers a holistic evaluation that considers the entire dataset of available reviews. In the context of our TripAdvisor dataset, we compute the mean and standard deviation of the number of required reviews, as computed over the entire population of hotels. We report the results in Fig. 7.

As anticipated, higher levels of the coverage threshold α require more reviews. We observe that, even to achieve a coverage as low as 60%, one would have to read around 50 reviews. The number rises rapidly to over 130 and 255 reviews for a coverage of 80% and 90%, respectively. However, previous work has demonstrated that consumers are not willing to consider such large number of reviews, especially since they typically have to consider multiple options prior to making a decision. Specifically, 90% of consumers read less than 10 reviews before forming an opinion about a business, while 68% of consumers form an opinion by reading just 1–6 reviews [5]. The large gap between the consumers' capacity and the number of reviews required to achieve even moderate coverage levels reveals the usefulness of the tips delivered by the TipSelector algorithm. In addition, the methodology that we described in this section can be used to evaluate any tip-mining approach without humans in the loop.

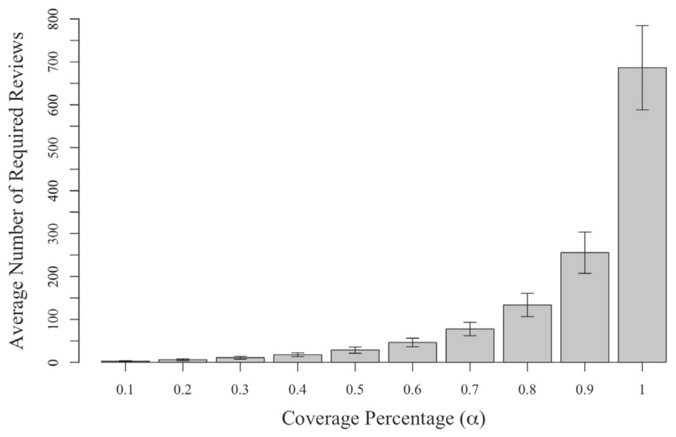

Fig. 7. Average number of required reviews to achieve α% of coverage within 1000 runs.

5. Managerial implications and future work

In this work we presented TipSelector, a new algorithm for mining tips from customer reviews. User-generated tips are one of the primary sources of information that support the customers' purchase decisions, and are often hosted on large review platforms to complement online reviews. Previous work has recently tackled the problem of automatically generating tips for a given business, in order to encode useful information that may have been overlooked by users, especially for less popular or new businesses. One of the primary shortcomings of the state-of-the-art for tip generation is its reliance on training data and manual annotations. This limits the applicability of the approach, as such resources are typically unavailable and hard to obtain in practice. The proposed TipSelector algorithm overcomes such limitations, as it is completely unsupervised. It thus emerges as an attractive choice for both platforms and practitioners. In addition, our experimental evaluation verifies that TipSelector outperforms the state-of-the-art in terms of both the informativeness and novelty of the generated tips. A secondary contribution of our work is a method for evaluating the informativeness of the set $\mathcal{T}$ of tips generated by a method, based on the number of reviews that a potential customer would have to read in order to obtain the information provided by $\mathcal{T}$. This method offers an alternative to expensive and time-consuming user studies, and can be used to complement the evaluation of any method for automated tip-generation.

5.1. Implications

Next, we discuss the implications of our work for e-commerce platforms, business managers, and customers.

E-commerce platforms: For platform managers, tips are an effective alternative to existing techniques for managing review content (i.e. ranking, selection, and summarization) and can serve as a valuable decision-support tool for the platform's customers. Unfortunately, user-submitted tips can often be redundant or uninformative. In addition, user-submitted tips tend to focus on a limited subset of popular businesses, creating a shortage of information on less popular or new businesses. Hence, a platform that uses TipSelector to automatically generate high-quality tips can gain a competitive advantage over platforms that do not offer any tips or are limited to tips submitted by users. By eliminating any dependencies on training data and manual annotations, TipSelector overcomes key limitations of previous tip-mining methods. In addition, TipSelector requires minimal parameter tuning. In fact, the only real parameterization is on the computation of the similarity neighborhood for the focal business. This can be easily addressed via tuning an intuitive similarity threshold or via the convergence approach that we describe in Section 3.1.

Business managers: Managers can use the tips mined by TipSelector to improve their products and services to meet customer expectations. Customer reviews reflect customer perception on product features and service quality. While they provide valuable information, reviews can vary in terms of informativeness and comprehensibility. In addition, it is impossible for business managers to read the hundreds or thousands of reviews on their business in order to find useful information and filter out redundant and noisy content. The tips generated by our algorithm can be used by business managers to identify key points hidden in the large volumes of reviews. If such points are negative, such as service flaws and management problems, the manager can then effectively address them to improve the quality and image of her business. Positive points can be prominently placed on the business website, to ensure that they are visible by potential customers. Even though TipSelector can be intuitively used even by practitioners with minimal technical skills, we do not expect that the businesses owners themselves will run the algorithm to generate tips. Instead, they will benefit by processing the tips that the review-hosting portal generates via TipSelector and then publishes on the portal's profile for their business.

Customers: Similar to business managers, we do not consider the average customer to be a direct user of our algorithm. However, by visiting a platform that utilizes TipSelector, customers will have access to high-quality tips on any business. These tips can support the customers' purchase decision by allowing them to quickly identify positive and negative key facts that are often obscured by the large volumes of noisy review content.

5.2. Future work

Our algorithmic design and evaluation focused on the novelty and usefulness of the produced tips. Future work could explore additional dimensions, such as the tip's relevance to specific user interests. Another interesting dimension is whether or not the tip includes a subjective opinion or is limited to providing information. A relevant study could measure the association between a tip's subjectivity and its usefulness, as perceived by users. Further, our implementation of TipSelector considers four types of information tokens, which we describe in detail in Section 3.2. While our results validate the use of these tokens, future work could expand this set to include additional linguistic or even semantic constructs. Finally, our work provides the first method for the automatic evaluation of tip-mining algorithms. Additional contributions in this area could significantly improve our ability to evaluate and compare such algorithms without humans in the loop and allow for large-scale evaluations that are typically prohibitive for user studies.

References

[1] M. Anderka, B. Stein, N. Lipka, Predicting quality flaws in user-generated content: the case of Wikipedia, Proceedings of the 35th International ACM SIGIR Conference on Research and Development in Information Retrieval, ACM. 2012, pp. 981–990.

[2] S. Baccianella, A. Esuli, F. Sebastiani, Multi-facet rating of product reviews, ECIR, vol. 9, Springer. 2009, pp. 461–472.

[3] S. Banerjee, T. Pedersen, An adapted Lesk algorithm for word sense disambiguation using WordNet, International Conference on Intelligent Text Processing and Computational Linguistics, Springer. 2002, pp. 136–145.

[4] D.M. Blei, A.Y. Ng, M.I. Jordan, Latent Dirichlet allocation, Journal of Machine Learning Research 3 (Jan) (2003) 993–1022.

[5] BrightLocal, Local Consumer Review Survey, 2017, https://www.brightlocal.com/learn/local-consumer-review-survey/. [Accessed: 2017-09-26].

[6] Q. Cao, W. Duan, Q. Gan, Exploring determinants of voting for the "helpfulness" of online user reviews: a text mining approach, Decision Support Systems 50 (2) (2011) 511–521.

[7] C.C. Chen, Y.-D. Tseng, Quality evaluation of product reviews using an information quality framework, Decision Support Systems 50 (4) (2011) 755–768.

[8] D. Chen, C.D. Manning, A fast and accurate dependency parser using neural networks, EMNLP, 2014. pp. 740–750.

[9] J.A. Chevalier, D. Mayzlin, The effect of word of mouth on sales: online book reviews, Journal of Marketing Research 43 (3) (2006) 345–354.

[10] E.G. Chewning, A.M. Harrell, The effect of information load on decision makers' cue utilization levels and decision quality in a financial distress decision task, Accounting, Organizations and Society 15 (6) (1990) 527–542.

[11] V. Chvatal, A greedy heuristic for the set-covering problem, Mathematics of Operations Research 4 (3) (1979) 233–235.

[12] S. Cohen, Aftereffects of stress on human performance and social behavior: a review of research and theory., Psychological bulletin 88 (1). (1980)

[13] P. Cremonesi, R. Facendola, F. Garzotto, M. Guarnerio, M. Natali, R. Pagano, Polarized review summarization as decision making tool, Proceedings of the 2014 International Working Conference on Advanced Visual Interfaces, ACM. 2014, pp. 355–356.

[14] C. Dellarocas, The digitization of word of mouth: promise and challenges of online feedback mechanisms, Management Science 49 (10) (2003) 1407–1424.

[15] V. Dhar, E.A. Chang, Does chatter matter? The impact of user-generated content on music sales, Journal of Interactive Marketing 23 (4) (2009) 300–307.

[16] W. Duan, B. Gu, A.B. Whinston, Do online reviews matter? An empirical investigation of panel data, Decision Support Systems 45 (4) (2008) 1007–1016.

[17] W. Duan, B. Gu, A.B. Whinston, Informational cascades and software adoption on the internet: an empirical investigation, MIS Quarterly (2009) 23–48.

[18] C. Fellbaum, WordNet, Wiley Online Library. 1998.

[19] S. Fernando, M. Stevenson, A semantic similarity approach to paraphrase detection, Proceedings of the 11th Annual Research Colloquium of the UK Special Interest Group for Computational Linguistics, 2008. pp. 45–52.

[20] J.R. Finkel, T. Grenager, C. Manning, Incorporating non-local information into information extraction systems by Gibbs sampling, Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics, Association for Computational Linguistics. 2005, pp. 363–370.

[21] R.A. Fisher, On the interpretation of χ^2 from contingency tables, and the calculation of P, Journal of the Royal Statistical Society 85 (1) (1922) 87–94.

[22] J.L. Fleiss, Measuring nominal scale agreement among many raters., Psychological Bulletin 76 (5) (1971) 378.

[23] C. Forman, A. Ghose, B. Wiesenfeld, Examining the relationship between reviews and sales: the role of reviewer identity disclosure in electronic markets, Information Systems Research 19 (3) (2008) 291–313.

[24] J. Grimmer, B.M. Stewart, Text as data: the promise and pitfalls of automatic content analysis methods for political texts, Political analysis 21 (3) (2013) 267–297.

[25] I. Guy, A. Mejer, A. Nus, F. Raiber, Extracting and ranking travel tips from user–generated reviews, Proceedings of the 26th International Conference on World Wide Web, International World Wide Web Conferences Steering Committee. 2017, pp. 987–996.

[26] S.R. Hiltz, M. Turoff, Structuring computer-mediated communication systems to avoid information overload, Communications of the ACM 28 (7) (1985) 680–689.

[27] M. Hu, B. Liu, Mining and summarizing customer reviews, Proceedings of the 10th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, ACM. 2004, pp. 168–177.

[28] N. Hu, I. Bose, N.S. Koh, L. Liu, Manipulation of online reviews: an analysis of ratings, readability, and sentiments, Decision Support Systems 52 (3) (2012) 674–684.

[29] N. Hu, N.S. Koh, S.K. Reddy, Ratings lead you to the product, reviews help you clinch it? The mediating role of online review sentiments on product sales, Decision Support Systems 57 (2014) 42–53.

[30] N. Hu, L. Liu, V. Sambamurthy, Fraud detection in online consumer reviews, Decision Support Systems 50 (3) (2011) 614–626.

[31] Q. Jones, G. Ravid, S. Rafaeli, Information overload and the message dynamics of online interaction spaces: a theoretical model and empirical exploration, Information Systems Research 15 (2) (2004) 194–210.

[32] R.M. Karp, Reducibility among combinatorial problems, Complexity of Computer Computations, Springer. 1972, pp. 85–103.

[33] S.-M. Kim, P. Pantel, T. Chklovski, M. Pennacchiotti, Automatically assessing review helpfulness, Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing, Association for Computational Linguistics. 2006, pp. 423–430.

[34] J.R. Landis, G.G. Koch, The measurement of observer agreement for categorical data, Biometrics (1977) 159–174.

[35] T. Lappas, M. Crovella, E. Terzi, Selecting a characteristic set of reviews, Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2012. pp. 832–840.

[36] T. Lappas, D. Gunopulos, Efficient confident search in large review corpora, Machine Learning and Knowledge Discovery in Databases (2010) 195–210.

[37] T. Lappas, G. Sabnis, G. Valkanas, The impact of fake reviews on online visibility: a vulnerability assessment of the hotel industry, Information Systems Research 27 (4). (2016)

[38] S.W. Litvin, R.E. Goldsmith, B. Pan, Electronic word-of-mouth in hospitality and tourism management, Tourism Management 29 (3) (2008) 458–468.

[39] Y. Lu, P. Tsaparas, A. Ntoulas, L. Polanyi, Exploiting social context for review quality prediction, Proceedings of the 19th International Conference on World Wide Web, ACM. 2010, pp. 691–700.

[40] D. Marasini, P. Quatto, E. Ripamonti, Assessing the inter-rater agreement for ordinal data through weighted indexes, Statistical Methods in Medical Research 25 (6) (2016) 2611–2633.

[41] H.-J. Min, J.C. Park, Identifying helpful reviews based on customer's mentions about experiences, Expert Systems with Applications 39 (15) (2012) 11830–11838.

[42] D. Newman, J.H. Lau, K. Grieser, T. Baldwin, Automatic evaluation of topic coherence, Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics, 2010. pp. 100–108.

[43] Q. Nguyen, Detecting Experience Revealing Sentences in Product Reviews, University of Amsterdam. 2012.

[44] T.-S. Nguyen, H.W. Lauw, P. Tsaparas, Using micro-reviews to select an efficient set of reviews, Proceedings of the 22nd ACM International Conference on Information & Knowledge Management, ACM. 2013, pp. 1067–1076.

[45] K.C. Park, Y. Jeong, S.H. Myaeng, Detecting experiences from weblogs, Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics, Association for Computational Linguistics. 2010, pp. 1464–1472.

[46] E.M. Rogers, R. Agarwala-Rogers, Organizational communication, Communication Behaviour (1975) 218–239.

[47] C.S. Sauer, T. Roth-Berghofer, Solution mining for specific contextualised problems: towards an approach for experience mining, Proceedings of the 21st International Conference on World Wide Web, ACM. 2012, pp. 729–738.

[48] R. Sipos, T. Joachims, Generating comparative summaries from reviews, Proceedings of the 22nd ACM International Conference on Information & Knowledge Management, ACM. 2013, pp. 1853–1856.
[49] D. Snowball, Some effects of accounting expertise and information load: an empirical study, Accounting, Organizations and Society 5 (3) (1980) 323–338.
[50] C. Speier, J.S. Valacich, I. Vessey, The influence of task interruption on individual decision making: an information overload perspective, Decision Sciences 30 (2) (1999) 337–360.
[51] P. Tsaparas, A. Ntoulas, E. Terzi, Selecting a comprehensive set of reviews, Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2011. pp. 168–176.
[52] O. Tsur, A. Rappoport, RevRank: A fully unsupervised algorithm for selecting the most helpful book reviews, International AAAI Conference on Web and Social Media, 2009.
[53] I. Weber, A. Ukkonen, A. Gionis, Answers, not links: extracting tips from Yahoo! Answers to address how-to web queries, Proceedings of the 5th ACM International Conference on Web Search and Data Mining, ACM. 2012, pp. 613–622.
[54] S. Whittaker, L. Terveen, W. Hill, L. Cherny, The dynamics of mass interaction, From Usenet to CoWebs, Springer. 2003, pp. 79–91.
[55] Z. Wu, M. Palmer, Verbs semantics and lexical selection, Proceedings of the 32nd Annual Meeting on Association for Computational Linguistics, Association for Computational Linguistics. 1994, pp. 133–138.
[56] K. Xu, S.S. Liao, J. Li, Y. Song, Mining comparative opinions from customer reviews for competitive intelligence, Decision Support Systems 50 (4) (2011) 743–754.
[57] Q. Ye, R. Law, B. Gu, The impact of online user reviews on hotel room sales, International Journal of Hospitality Management 28 (1) (2009) 180–182.
[58] Q. Ye, R. Law, B. Gu, W. Chen, The influence of user-generated content on traveler behavior: an empirical investigation on the effects of e-word-of-mouth to hotel online bookings, Computers in Human Behavior 27 (2) (2011) 634–639.
[59] Z. Zhang, B. Varadarajan, Utility scoring of product reviews, Proceedings of the 15th ACM International Conference on Information and Knowledge Management, ACM. 2006, pp. 51–57.
[60] L. Zhuang, F. Jing, X.-Y. Zhu, Movie review mining and summarization, Proceedings of the 15th ACM International Conference on Information and Knowledge Management, ACM. 2006, pp. 43–50.

Dr. Theodoros Lappas received the PhD degree in computer science from the University of California, Riverside, in 2011. He is currently an Assistant Professor of Information Systems in the School of Business at Stevens Institute of Technology. His research focuses on large-scale Reputation Systems, as well as on scalable Data Mining and Machine Learning algorithms for the analysis of natural language and social graphs. Among others, Dr. Lappas has published multiple papers on various aspects of review-based reputation systems, including review summarization, opinion mining, and fake reviews.