Micropinion Generation: An Unsupervised Approach to Generating Ultra-Concise Summaries of Opinions

Kavita Ganesan
Dept. of Computer Science
University of Illinois at
Urbana-Champaign
Urbana, IL
kganes2@cs.uiuc.edu

ChengXiang Zhai
Dept. of Computer Science
University of Illinois at
Urbana-Champaign
Urbana, IL
czhai@cs.uiuc.edu

Evelyne Viegas
Microsoft Research
Redmond, WA
evelynev@microsoft.com

ABSTRACT

This paper presents a new unsupervised approach to generating ultra-concise summaries of opinions. We formulate the problem of generating such a micropinion summary as an optimization problem, where we seek a set of concise and non-redundant phrases that are readable and represent key opinions in text. We measure representativeness based on a modified mutual information function and model readability with an n-gram language model. We propose some heuristic algorithms to efficiently solve this optimization problem. Evaluation results show that our unsupervised approach outperforms other state of the art summarization methods and the generated summaries are informative and readable.

Categories and Subject Descriptors

H.3.3 [**Information Search and Retrieval**]: Text Mining

General Terms

Algorithm

Keywords

micropinion, web n-gram, opinion summarization

1. INTRODUCTION

Summarization of opinions is crucial in helping users digest the many opinions expressed on the web. Previous studies have primarily focused on the task of generating highly structured summaries. This could be a simple sentiment summary such as 'positive' or 'negative' on a topic of interest [17, 16, 26, 29] or a multi-aspect summary such as *battery life: 1 star*, *screen: 3.5 stars*, etc for an mp3 player [14, 24, 22, 13, 9]. While structured summaries can be useful in conveying the general sentiments about a person, product, or service, such summaries lack the level of detail that an unstructured (textual) summary could offer, often forcing users to go back to the original text to get more information. Textual summaries are thus critical in conveying key opinions and reasons for those opinions at different granularities (i.e. entity level or topic level).

Unfortunately, generating textual opinion summaries is a hard task. First, the summaries have to be representative of the key opinions (to avoid bias) and then, it has to be

Table 1: **Example of micropinion summaries on two different topics given a constraint of 10 words.**

Mp3 Player Y	Restaurant X
Very short battery life. Big and clear screen. **(8 words)**	Good service. Delicious soup dishes. Very noisy at nights. **(9 words)**

readable so that it can be easily understood by the reader. Further, with the increased use of hand-held devices for various online activities such as shopping and finding places to eat, the conciseness or compactness of such summaries is also crucial. Indeed, there are many scenarios where concise summaries would be very beneficial. Consider shopping sites where there could be hundreds of reviews per product. In this case, a concise *pros and cons* summary consisting of a list of short opinion phrases would help convey critical information about a product, thus facilitating decision making. Such concise summaries are also suitable as *tweets* that are automatically generated based on blogs or news articles.

In this paper, we explore the task of generating a set of very concise phrases, where each phrase (micropinion) is a summary of a key opinion in text. The ultra-concise nature of the phrases allows for flexible adjustment of summary size according to the display constraints. Our emphasis on generating **concise** abstractive summaries (rather than extractive summaries), makes this a unique summarization problem which has not been previously studied. In Table 1, we show examples of envisioned micropinion summaries.

On the surface, our summarization task appears to be similar to a keyphrase extraction problem. However, since the goal is to help users digest the underlying opinions, there are some important aspects that are unique to this task. In traditional keyphrase extraction, the goal is primarily to select a set of phrases to characterize documents. Thus, phrases such as *battery life* and *screen* from a set of reviews about a phone may be selected as candidate keyphrases. For our task, such keyphrases are meaningless without the associated opinions. In addition, since we want readers to *understand* the opinions in the summary, the phrases in the summary need to be fairly well-formed and grammatically sound. Consider a phrase such as *'short battery life'* in contrast to one such as *'life short battery'*. Even though both phrases contain the same words, the ordering is different, changing their meaning, where the former is readable but the latter would make no sense to the reader. This readability aspect is less of a concern in traditional keyphrase extraction as the phrases are only used to 'tag' documents.

In this paper, we present a novel *unsupervised* approach to generating *micropinion* summaries for different display constraints. Our idea is to start with a set of high frequency seed words from the input text, gradually forming meaningful higher order n-grams. At each step the n-grams are scored based on their *representativeness* (measured based a modified mutual information function) and *readability* (measured based on an n-gram language model). We frame this problem as an optimization problem, where we attempt to find a set of concise, non-redundant phrases (not necessarily occurring in the original text) that are readable and represent the major opinions. We propose heuristic algorithms to solve this optimization problem efficiently.

Evaluation results using a set of product reviews shows that our approach is effective in generating micropinion summaries and outperforms other summarization approaches. Further, the proposed approach is lightweight and general, requiring no linguistics or domain knowledge. It can thus be used in a variety of domains and could even be used with other languages. The dataset and demo related to this work can be found at `http://timan.cs.uiuc.edu/downloads.html`.

2. RELATED WORK

Research in opinion summarization has been quite extensive with much of the focus being on generating structured summaries of opinions. Early opinion summarization systems focused on predicting the overall sentiment class (positive or negative) on a given piece of text [17, 16, 26, 29]. In later years, this definition was generalized to a multipoint rating scale where ratings are predicted on a set of aspects [14, 24, 22, 13, 9, 8, 27]. In contrast to structured summarization approaches, studies addressing unstructured summarization of opinions are notably fewer and *abstractive* approaches are less common than *extractive* approaches. To the best of our knowledge, no previous work has studied the generation of *ultra-concise* and *abstractive* summaries using an optimization framework such as ours with special emphasis on compactness, readability and representativeness.

The works closest to ours are perhaps the work of Branavan et. al [1] and Ganesan et. al [7] (Opinosis) where both attempt to generate concise summaries of opinions. Branavan et. al implemented a hierarchical Bayesian model reusing existing pros and cons phrases to train their keyphrase extraction model. Unlike their approach, ours is unsupervised and domain independent where we try to generate concise and informative phrases using only the existing text and a publicly available n-gram model. While Opinosis is also unsupervised, it uses a graph data structure that relies on the structural redundancies in text to discover informative phrases. In our method, we do not rely on structural redundancies but rather grow seed words from the input text into longer and meaningful phrases. With this, it is possible to generate new phrases that are more concise to convey essential opinions.

Carenini et. al [2, 3] compared both the use of extractive and abstractive summarization methods for opinion summarization. A key difference between our abstractive approach and theirs is that we focus on generating *ultra-concise summaries* (set of short phrases) using a domain-independent and lightweight approach, while they focus on generating paragraph level summaries made up of *full length sentences* using a complex natural language generation architecture. Other types of textual opinion summaries include contrastive summarization [10, 18] where the summaries are meant to highlight contradicting sentence pairs in opinionated text. Note that these summaries are not meant to summarize key opinions in text.

As discussed in Section 1, while there may be some key differences between our task and traditional keyphrase extraction, conceptually, the tasks are quite similar. Thus, presumably many of the keyphrase extraction approaches [25, 30, 6, 15] can be used for the task of micropinion generation. Most of the keyphrase extraction approaches are highly *supervised* and rely on various linguistics annotations in order to find candidate phrases. Our approach is not only *unsupervised* and domain independent, but we also do not rely on linguistics annotations to discover meaningful phrases, which makes our approach more general. Further, our optimization framework which gives special emphasis to the readability and representativeness aspects, has not been explored by any of the keyphrase extraction approaches.

3. AN OPTIMIZATION FORMULATION FOR MICROPINION SUMMARIZATION

The goal of our task is to generate a compact and informative summary using a set of micropinions. A micropinion is a short phrase (between 2 and 5 words) summarizing a key opinion in text. The minimum phrase length of 2 is based on the observation that a meaningful opinion is often targeted towards an object [12] (e.g. *clear screen* vs. *clear*). The maximum phrase length of just 5 is to allow for flexibile adjustment of summary size according to the display requirements. For example, a small phone may have stricter display requirements than a full sized PDA. Thus, compared with most existing work in text summarization, a unique aspect of our goal is to maximize information conveyed in the given constraints.

Formally, given a set of sentences $Z = \{z_i\}_{i=1}^n$ from an opinion document, our goal is to generate a micropinion summary, $M = \{m_i\}_{i=1}^k$, where $|m_i| \in [2,5]$ words and each $m_i \in M$ conveys a key opinion from Z. Note that while we require m_i to use words that have occurred at least once in Z, we do *not* require m_i to be an exact subsequence of any of the sentences in Z. This makes our task setup more of an *abstractive summarization* problem. In contrast to the predominantly popular extractive summarization task, this raises a new interesting challenge in how to *compose* concise and meaningful summaries using only words from the original text. This requirement is not restrictive since user generated content such as opinions have the benefit of volume and with this there would be many choices of words to describe a major opinion.

Intuitively, we would like the generated micropinion summary, $M = \{m_i\}_{i=1}^k$ to be: (1) *representative* (i.e., each m_i should reflect the major opinions in the original text), (2) *readable* (i.e., each m_i should be well formed according to the language's grammatical rules), and (3) *compact* (i.e., M should use as few words as possible to convey the major opinions). Thus, in theory, we can solve this new summarization problem by enumerating all possible summaries and evaluating each one to see how well it satisfies these three criteria, which suggests that we can formulate micropinion summarization as the following optimization problem:

$$M^* = \arg\max_{M=\{m_1,\ldots,m_k\}} \sum_{i=1}^k [S_{rep}(m_i) + S_{read}(m_i)]$$

$$subject\ to \qquad \sum_{i=1}^{k} |m_i| \leq \sigma_{ss}$$
$$S_{rep}(M) \geq \sigma_{rep}$$
$$S_{read}(M) \geq \sigma_{read}$$
$$sim(m_i, m_j) \leq \sigma_{sim} \forall, i, j \in [1, k]$$

where
1. $S_{rep}(m_i)$ is a scoring function measuring the representativeness of m_i;
2. $S_{read}(m_i)$ is a scoring function measuring the readability of m_i;
3. $sim(m_i, m_j)$ is a similarity function measuring the similarity of m_i and m_j;
4. σ_{ss} and σ_{sim} are two user adjustable parameters to control the maximum length of the summary and the amount of redundancy allowed in the summary;
5. σ_{rep} and σ_{read} are minimum representativeness and readability thresholds to improve efficiency of the algorithm.

The rationale for this optimization formulation is the following: First, the objective function captures the intention of optimizing both *representativeness* and *readability* which ensures that the summaries reflect opinions from the original text and are also reasonably well-formed. Second, the compactness is captured by setting a threshold on the maximum length of the summary and a threshold on the similarity between any two phrases in the summary so as to minimize redundancy. These parameters are indeed necessary to accommodate different needs of the user or the application. For example, when devices have very small screens, the user may desire smaller summaries with good coverage of information (i.e. less redundancies). On the other hand, with bigger screens, users may desire longer summaries and may be willing to tolerate more redundancies in order to get more detailed information. In both cases, these application or user specific requirements can be easily adjusted using the σ_{ss} and σ_{sim} thresholds. Finally, we use two thresholds for minimum representativeness and readability respectively which ensures efficiency of the algorithm by reducing the exploration of non-promising candidate expansion paths. In turn, this also ensures a summary to be both representative and readable (i.e., avoid "skewed tradeoffs").

Clearly, the main challenge now is to define the representativeness function $S_{rep}(m_i)$, the readability function $S_{read}(m_i)$, and the similarity function $sim(m_i, m_j)$. In this paper, we focus on studying how to define $S_{read}(m_i)$ and $S_{rep}(m_i)$ as they represent interesting new challenges raised by micropinion summarization. For $sim(m_i, m_j)$, we simply use the Jaccard similarity [21] to measure and eliminate redundancy since it is not the focus of our study.

In Section 3.1, we will explain the proposed method to measure the representativeness of m_i, $S_{rep}(m_i)$, based on the modified pointwise mutual information of the words in m_i. This captures how well m_i aligns with major opinions in the original text. Then in Section 3.2, we describe how we estimate readability, $S_{read}(m_i)$, of phrases based on a general n-gram language model with the assumption that m_i is more readable if m_i is more frequent according to the n-gram model.

3.1 Representativeness

In general, opinions are often quite redundant and may contain contradicting viewpoints. Hence, generating a few highly representative phrases is a challenge. Since we are mainly interested in summarizing the *major* opinions in text, a representative summary would be one that can accurately bring to surface the most common *complaint, praise* or *critical information*. For example, assuming we have 10 sentences in the input document that talks about "battery life being <u>short</u>" and one about "battery life being <u>excellent</u>", by our definition, the former would be the representative opinion phrase.

In determining the representativeness of a phrase m_i, we have defined two key properties of a highly representative phrase: (1) the words in each m_i, should be strongly associated within a narrow window in the original text and (2) the words in m_i should be sufficiently frequent in the original text.

The first property ensures that only a set of related words are used in the generated phrases to avoid conveying incorrect information. As an example, if we were to generate a phrase containing the word *short*, it is important that *short* is used with the right set of words or we may convey information not present in the original text (e.g. *the phone is short* instead of *battery life is short*). While this is not a problem for methods that use existing n-grams from the input document, it is important for our method as we form potentially new n-grams from seed words. This first property of strong association can be captured by computing the *pointwise mutual information* (PMI) of words in m_i based on its alignment with the original text. PMI was shown to be the best metric to measure strength of association of word pairs [23]. For a set of strongly associated words to be considered representative, these words should also be significant in the input text. The second property thus rewards n-grams containing words that occur frequently in the original text. Formally, suppose $m = w_1...w_n$ is a candidate phrase. We define $S_{rep}(m)$ as follows:

$$S_{rep}(w_1...w_n) = \frac{1}{n} \sum_{i=1}^{n} pmi_{local}(w_i) \tag{1}$$

where $pmi_{local}(w_i)$ is a local pointwise mutual information function defined as:

$$pmi_{local}(w_i) = \frac{1}{2C} \sum_{j=i-C}^{i+C} pmi'(w_i, w_j), i \neq j \tag{2}$$

where C is a contextual window size. The $pmi_{local}(w_i)$ measures the average strength of association of a word w_i with all its C neighboring words (on the left and on the right). So, for the phrase *short battery life*, assuming $C = 1$, for *short* we would obtain the average PMI score of *short* with 'battery' and for *battery* we would obtain the average PMI of *battery* with 'short' and *battery* with 'life'. When this is done for each $w_i \in m$, this would give a good estimate of how strongly associated the words are in m, which is the rationale for Equation 1. To capture the second property, we use a modified PMI scoring referred to as pmi' where the pmi' between two words, w_i and w_j, is defined as:

$$pmi'(w_i, w_j) = \log_2 \frac{p(w_i, w_j) \cdot c(w_i, w_j)}{p(w_i) \cdot p(w_j)} \tag{3}$$

where $c(w_i, w_j)$ is the frequency of two words co-occurring in a sentence from the original text within the context window of C (in any direction) and $p(w_i, w_j)$ is the corresponding joint probability. We later show the influence of C on the generated summaries. The co-occurrence frequency,

$c(w_i, w_j)$, which is not part of the original PMI formula, is integrated into our PMI scoring to reward frequently occurring words from the original text. The problem with the original PMI scoring is that it yields in high scores for low frequency words. By adding $c(w_i, w_j)$ into the PMI scoring, we ensure that low frequency words do not dominate and moderately associated words with high co-occurrences have relatively high scores.

3.2 Readability

For a summary to be readable, it will have to be fairly well-formed and grammatically sound according to the language's grammatical rules. The readability aspect is an important requirement in any summarization task as this allows a reader to easily digest information. In extractive summarization, the readability of a summary is less of a problem as *existing sentences* or *phrases* are reused to form summaries. In our approach, we do not reuse sentences or phrases directly from Z, but rather attempt to generate new phrases using existing words from Z. Hence, there is no guarantee that the generated phrases would be well-formed and readable. For example, without readability scoring, it will be hard to distinguish the grammatical difference between the phrases *clear is screen* and *screen is clear*.

Without any form of supervision, measuring the readability of a phrase is difficult. We address this problem by leveraging the publicly available Microsoft N-gram service[1] [28], to score all our system generated phrases. The abundance of textual content on the web which includes blogs, news articles, user reviews, tutorials, etc makes an n-gram model estimated based on all such content an approximate judge of how readable the system generated phrases are. The intuition is that, if a generated phrase occurs frequently on the web, then this phrase is more likely readable. This approximation to determining readability is fair, since the web as a corpus, is extremely large and there would be enough evidence to segregate a well-constructed phrase from a poorly constructed one.

Specifically, we use the Microsoft's trigram language model trained on the body text of documents to obtain conditional probabilities of the candidate phrases. In scoring each phrase, we first obtain the conditional probability of different sets of trigrams in the phrase. These scores are combined and averaged to generate the final readability score. Suppose $m = w_1...w_n$ is a candidate phrase, $S_{read}(m)$ is thus defined as follows:

$$S_{read}(w_i...w_n) = \frac{1}{K} \cdot \log_2 \prod_{k=q}^{n} P(w_k|w_{k-q+1}...w_{k-1}) \quad (4)$$

where q represents the n-gram order of the model used and in our case, $q = 3$. K represents the number of conditional probabilities computed.

Equation 4 is essentially the chain rule used to compute the joint probability in terms of conditional probabilities, which is then averaged. Averaging the scores accounts for the differences in phrase lengths and enables us to set cutoffs to prune non-promising candidates.

3.3 Parameter Settings

The optimization formulation involves several parameters to be empirically set. Some of these have to be set in an application-specific way based on tradeoffs between multiple criteria of summarization. The σ_{ss} and σ_{sim} parameters

are required to enable users or the application to control the desired total length of a summary and the amount of redundancy allowed in the summary. These values can be set, for example based on the screen size of a mobile device if the summary is to be displayed on such a device; smaller σ_{ss} for shorter summaries and smaller σ_{sim} for less redundant summaries. Note that it does not make much sense for a system to automatically set these two parameters due to their inherent dependency on user preferences.

σ_{rep} and σ_{read} mainly help with the efficiency of the optimization algorithm and also helps ensure the minimum representativeness and readability of phrases; we will later show that these two values can be set to reasonably small values and has little effect on performance unless the thresholds are too restrictive. With this setup, the remaining challenge is to solve the optimization problem efficiently, which we will discuss in the next section.

4. SUMMARIZATION ALGORITHM

Since we want to *compose* new phrases using words from the original text, the potential solution space for our optimization problem, is huge. In practice, it is infeasible to enumerate all the possible summary candidates and score each one of them. In this section, we propose a greedy algorithm to solve the optimization problem by systematically exploring the solution space with heuristic pruning so that we only touch the most promising candidates.

At a high level, we start with a set of high frequency unigrams from the original text. We then gradually merge them to generate higher order n-grams as long as their *readability* and *representativeness* remain reasonably high. This process of generating candidates stops when an attempt to grow an existing candidate leads to phrases that are low either in readability or representativeness (i.e. does not satisfy σ_{rep} or σ_{read}).

Specifically, the input to our summarization algorithm is a set of sentences from an opinion containing document. For example, all review sentences about the *iPhone*. We denote these sentences as $Z = \{z_i\}_{i=1}^n$. The output is a micropinion summary with a set of n-gram phrases $M = \{m_1, ..., m_k\}$, where the number of micropinions is determined based on the constraints of the optimization problem. The summarization algorithm consists of the following three steps:

Step 1. Generation of seed bigrams: The first step takes the original text, Z as input and generates a set of promising bigrams based on combinations of high frequency unigrams.

Step 2. Generation of scored n-grams: The second step takes the seed bigrams as input and further grows them into a set of promising n-grams by concatenating bigrams that share an overlapping word. While generating n-grams, we also compute their representativeness and readability scores S_{rep} and S_{read}, and prune all the cases where any of these scores is below the corresponding threshold. We further check redundancy of the generated candidates, and if two phrases have a similarity higher than the σ_{sim} threshold, we would discard the one with a lower combined score of $S_{rep} + S_{read}$.

Step 3. Generation of micropinion summary: The final step is to sort all the candidate n-grams based on their objective function values (i.e., sum of S_{rep} and S_{read}) and generate a micropinion summary M by gradually adding phrases with the highest scores to our summary until the

---

[1] http://web-ngram.research.microsoft.com

accumulated summary length reaches the length threshold σ_{ss}.

We will now focus on elaborating steps 1 and 2 since step 3 is straightforward.

4.1 Generation of seed bigrams

As redundancies are inherent in opinionated documents, this property can be leveraged to shortlist a set of seed words that can be used to generate higher order n-grams. This way, we avoid having to try every combination of words. Our assumption is that, if a word is not frequent in the original text, it is unlikely a good candidate word to be included in any phrase of a micropinion summary (presumably we will have other better candidate words to work with).

Assuming that the input text is a set of sentences, we shortlist a set of high frequency unigrams from the input text, where the unigrams should have counts larger than the *median* count (after ignoring words with frequency of 1). This ensures that the cutoff is adjusted according to the input size. The low threshold serves as an initial reduction in search space; further reduction happens when S_{rep} and S_{read} are computed later.

Each of these high frequency unigrams is then paired with every other unigram to form bigrams. For example, if we have the words 'battery' and 'life', the bigrams generated would be 'battery life' and 'life battery'. We then compute the representativeness score S_{rep} of each bigram and keep only those bigrams whose S_{rep} passes the threshold σ_{rep}. Although the combination of words could be quite random, the S_{rep} function helps prune invalid combinations (as it demands co-occurrences of two words within a window C of the original text).

4.2 Depth-first search for candidate generation

Using the seed bigrams described in the previous section, we attempt to generate higher order n-grams that will finally serve as candidate micropinions. If there are a large number of seed bigrams (for large input documents), the starting seeds can further be shortlisted by their representativeness scores. For example, we can use only the top 500 seeds as the starting point. Algorithm 1 outlines the steps involved in candidate micropinion generation.

Algorithm 1 $GenerateCandidates(p)$

1: **Input:** A candidate phrase (bigrams initially)
2: **Output:** A set of micropinions
3: **if** $(S_{read}(p) < \sigma_{read} || S_{rep}(p) < \sigma_{rep})$ **then**
4:   **return**
5: **end if**
6: **if** $ValidCandidate(p, candList)$ **then**
7:   $candList \leftarrow \{candList \cup p\}$
8: **end if**
9: $joinList \leftarrow GetJoinList(seedBigrams)$
10: **for** $bigram \in joinList$ **do**
11:   **if** $NotMirror(p, bigram)$ **then**
12:     $newPhrase \leftarrow Merge(p, bigram)$
13:     $Score(newPhrase)$
14:     $GenerateCandidates(newPhrase)$
15:   **end if**
16: **end for**

First, for any incoming phrase, p, a check is done to determine if p is a promising phrase. This is done by checking the $S_{read}(p)$ and $S_{rep}(p)$ scores with the corresponding thresholds (line 3). If the score constraints are not fulfilled, then p will not be further expanded. This step ensures that we explore only reasonable phrases, thus improving efficiency.

If p fulfills the score constraints, then a check is done to determine if it can become a candidate micropinion (line 6). This is based on the similarity between p and the existing candidate phrases from the pool of candidates. If p is not similar to any of these phrases, then p will automatically be added to this candidate pool. If p is similar to a phrase X in the pool, p will replace X if $S_{rep}(p) + S_{read}(p) > S_{rep}(X) + S_{read}(X)$. In other words, at any given time, we will have a set of non-redundant candidate phrases.

Once the validity of a phrase has been determined, the algorithm proceeds recursively in a depth first fashion in an attempt to expand p to a higher order n-gram (line 9-16). We expand phrases using concepts used for pattern growth as shown in [19]. In particular, we impose a merge requirement between a candidate phrase p and a bigram, b (from the set of seed bigrams) as follows: (1) the ending word in p should overlap with the starting word in b and (2) p should not be a mirror of b. With this, all seed bigrams that satisfy this requirement will be merged with p. Consider a phrase *very long **battery*** and a seed bigram ***battery** life*. The overlapping word *battery*, connects the two phrases and since one is not a mirror of the other, the two phrases can be merged to form *very long battery life*. Such a pattern growth approach eliminates the need for an exhaustive search and it also avoids exploration of n-grams that are extremely random and unlikely useful. The newly merged phrases are then scored for their readability and representativeness prior to being further expanded (line 13).

5. EXPERIMENTAL SETUP

5.1 Dataset

To evaluate this micropinion generation task, we leverage user generated reviews from CNET[2] for 330 different products. Each product has 5 associated reviews at the minimum. The CNET review structure is such that a user writes a *full length review* about a product followed by a brief summary about the *pros and cons* (PC). The PCs are usually a set of short phrases such as "*bright screen, fast download, etc*" where these phrases tend to summarize the full reviews and hence are quite redundant. In evaluating our summarization task, we ignore the PCs and use only the *full reviews* for a product to make the summarization task more realistic (i.e we eliminate obvious redundancies). Thus, for a given product X, we generate a *review document*, r_x, where r_x holds all sentences from the full reviews related to X. On average, there were 500 sentences per review document. Out of the 330 review documents generated, we use only 230 documents for evaluation as 100 were withheld to train one of the baseline approaches.

While the PCs seem ideal as the gold standard summary, in some cases the PCs are just an enumeration of features that the user likes or dislikes and contain no specific opinions. For example, "**Pros**: *battery, sound*; **Cons**: *hard disk, screen*". Since we need a set of opinion phrases that are more complete such as "*improved battery life*; *crystal clear sound*", we leverage these PCs to help human summarizers compose meaningful summaries. Specifically, for each product X, we find the top 10 high frequency phrases from the PCs which are then presented as hints to the human summarizers. Since we have a large number of review documents to summarize, such hints help the human summarizer with topic coverage, thus reducing bias in the summaries. Two

---

[2] http://www.cnet.com

human summarizers, were asked to read the reviews of each product presented to them and then compose a set of phrases summarizing key opinions on the product. Out of the 330 products, 165 were assigned to one summarizer and the remaining to the other. With this, for each r_x, we obtained a corresponding human summary, h_x.

5.2 Quantitative Evaluation

In demonstrating the effectiveness of our approach, we need to quantify to what extent our summaries are representative of key opinions and how readable these summaries are so that they can be understood by human readers. One way of assessing the quality of summaries is to measure how well system summaries resemble human composed summaries. Keyphrase extraction tasks are typically evaluated based on the number of overlapping keyphrases between system generated keyphrases and the gold standard ones. This requires exact matches which is unlikely, as there could be many ways to describe one opinion and subtle variations may result in an unfair 'no match'.

We thus chose to use ROUGE [11], an evaluation method based on n-gram overlap statistics found to be highly correlated with human evaluations. ROUGE was also the standard measure used in the DUC 2004 summarization task[3] on generating very short summaries such as headline summaries (< 10 words). ROUGE is ideal for our task as it does not demand exact matches but it can measure both representativeness and readability of summaries. As an example, ROUGE-1 measures the overlap of unigrams between system summaries and human summaries, thus measuring **representativeness**. Higher order ROUGE-N ($N > 1$) captures the match of subsequences, which measures the fluency or **readability** of summaries. In our experiments, we primarily use ROUGE-1, ROUGE-2 (bigram overlap) and ROUGE-SU4 (skip-bigram with maximum gap length of 4).

5.3 Qualitative Evaluation

In addition to the automatic ROUGE evaluation, we also performed a manual evaluation to assess the potential utility of the micropinion summaries to real users. Specifically, two assessors were asked to read the micropinion summaries presented to them (using the original reviews as reference) and then fill out a questionnaire assessing the summary on several aspects related to its effectiveness. Note that these assessors are different from those that composed the gold-standard summaries. The questionnaire consisted of three key questions rated on a scale from 1 (Very Poor) to 5 (Very Good). The questions are as follows:
Grammaticality: Are the phrases in the summary readable with no obvious errors? Score (1) - None of the phrases are readable or comprehensible; Score (5) - I don't see any issues with the phrases in the summary.
Non-redundancy: Are the phrases in the summary unique with unnecessary repetitions such as repeated facts or repeated use of noun phrases? Score (1) - All the phrases mean the same thing; Score (5) - The phrases are very unique, summarizing very different topics or issues.
Informativeness: Do the phrases convey important information regarding the product? This can be positive/negative opinions about the product or some critical facts about the product. Score (1) - None of the phrases contain useful or accurate information about the product. Score (5) - All

the phrases contain accurate opinions or critical information about the product.

Note that the first two aspects, *grammaticality* and *non-redundancy* are linguistic questions used at the 2005 Document Understanding Conference (DUC) [4]. The last aspect, *informativeness*, has been used in other summarization tasks [5] and is key in measuring how much users would learn from the summaries.

For this evaluation, we used the micropinion summaries for 70 different products that were randomly selected from our dataset. The assessors were not informed about which method was used to generate the summaries.

5.4 Baselines

To assess how well our approach compares with existing approaches, we use three representative baselines. As our first baseline, we use TF-IDF, an unsupervised language-independent method commonly used for keyphrase extraction tasks. To make the TF-IDF baseline competitive, we limit the n-grams in consideration to those that contain at least one adjective (i.e. favoring opinion containing n-grams). Note that the performance is much worse without this selection step. Then, we used the same redundancy removal technique used in our approach to allow a set of non-redundant phrases to be generated.

For our second baseline, we use KEA[4] [30], a highly cited, state-of-the-art keyphrase extraction method. KEA builds a Naive Bayes model with known keyphrases, and then uses the model to find keyphrases in new documents. The KEA model was trained using the 100 review documents withheld from our dataset (explained in Section 5.1). With KEA as a baseline, we would gain insights into the applicability of existing keyphrase extraction approaches for the task of micropinion generation. Note that since KEA uses training data and our method does not, the comparison is, strictly speaking, an unfair comparison.

As our final representative baseline, we use Opinosis [7], an abstractive summarizer designed to generate textual summary of opinions. Opinosis was shown to be more effective in generating concise opinion summaries compared to extractive approaches like MEAD [20]. We turned off the optional *collapse* feature in Opinosis which attempts to merge several short phrases into longer ones to simulate the task of micropinion generation. All other settings were set at their default values.

In addition to these baselines, we also performed a run to examine the benefit of our strategy of composing potentially new phrases as opposed to relying solely on phrases that occurred in the original text. For this run (WebNgram$_{seen}$), we force our search algorithm to return n-grams that have occurred at least once in the reviews. To give a fair comparison, all the n-grams in each of our baseline are 2-5 words long. Our approach is referred to as *WebNgram* in all our experiments.

6. RESULTS

By default, the efficiency parameters in our approach are set as follows: minimum readability score, $\sigma_{read} = -2$ (log of probabilities); minimum representativeness score, $\sigma_{rep} = 4$. As will be shown later, the performance of summarization is not very sensitive to the settings of these parameters. The contextual window size is set to $C = 3$, which is the opti-

---

Table 2: ROUGE-1 & ROUGE-2 recall scores at different σ_{ss}; *statistically significant with p-value< 0.01, **p-value < 0.001

ROUGE-1							ROUGE-2						
σ_{ss}	**5**	**10**	**15**	**20**	**25**	**30**	σ_{ss}	**5**	**10**	**15**	**20**	**25**	**30**
Tfidf	0.032	0.062	0.085	0.104	0.122	0.137	Tfidf	0.006	0.010	0.014	0.016	0.018	0.020
KEA	**0.046**	0.079	0.104	0.122	0.142	0.164	KEA	0.010	0.021	0.026	0.029	0.033	0.038
Opinosis	0.034	0.094	0.130	0.157	0.185	0.209	Opinosis	0.012	0.031	0.044	0.049	0.055	0.056
WebNGram$_{seen}$	0.033	0.092	0.121	0.151	0.173	0.190	WebNGram$_{seen}$	0.013	0.035	0.043	0.050	0.056	0.060
WebNGram	0.037	**0.102**	**0.135**	**0.173**	**0.195**	**0.219**	**WebNGram**	**0.017**	**0.047**	**0.055**	**0.069**	**0.074**	**0.080**
% Improvement							% Improvement						
WebNgram over **TfIdf**	+13.25*	+39.90**	+36.61**	+40.00**	+37.32**	+37.77**	WebNgram over **TfIdf**	+65.55**	+77.80**	+74.22**	+77.41**	+75.52**	+75.07**
WebNgram over **KEA**	-25.36	+22.38*	+22.74**	+29.31**	+27.33**	+25.37**	WebNgram over **KEA**	+39.33*	+56.59**	+52.95**	+58.18**	+55.66**	+52.34**
WebNgram over **Opinosis**	+7.48	+8.19	+3.41	+9.17	+5.02	+4.89	WebNgram over **Opinosis**	+27.73	+33.57**	+19.71*	+29.95**	+26.09**	+29.19**

mal setting. The user adjustable parameter for redundancy control (using Jaccard) is set to $\sigma_{sim} = 0.40$, for reasonable diversity in phrases.

Comparison of summarization strategies. First, we assess the effectiveness of our approach (WebNgram) in comparison with other representative approaches (KEA, Opinosis, TF-IDF). The performance comparison is shown in Table 2 for different σ_{ss} settings. Since the summary size is constrained, we are primarily interested in the gain in recall as the precision is proportional to recall when summary length is fixed. First, based on Table 2, we see that overall, WebNgram has the highest ROUGE-1 and ROUGE-2 scores, outperforming all baseline methods. The statistical significance of improvements (based on Wilcoxon test) is indicated in Table 2. As the summary size increases, WebNgram consistently gains both in terms of ROUGE-1 and ROUGE-2. This shows that representative phrases are being included in the summary and these phrases are also readable as shown by the consistent gain in ROUGE-2 (the other baseline methods do not gain as much with ROUGE-2).

Next, we see that the ROUGE-1 scores of Opinosis and WebNgram are quite similar, but the ROUGE-2 scores are very different. The similarity in ROUGE-1 scores indicates that Opinosis is able to capture similar topics and opinions as WebNgram. However, the ROUGE-2 score of Opinosis indicates that the generated phrases are less fluent. Through manual inspection, we found that Opinosis generates many short phrases (< 4 words), generally fragmented and thus less fluent. This explains the lower ROUGE-2 scores. This is further confirmed by the average number of phrases generated by Opinosis which is higher (i.e more fragmented) than that of WebNgram (shown in Table 3). One possible reason as to why Opinosis has problems generating longer and more complete phrases is lack of structural redundancies between sentences in the CNET reviews.

Amongst all the approaches, TF-IDF performs the worst as shown in Table 2. This is likely due to insufficient redundancies of meaningful n-grams. When there are subtle differences in common expressions, it is often difficult to discover redundant n-grams. This is where our method excels as we do not directly rely on the structure of the input text, but rather expand high frequency seed words into longer and meaningful phrases. Finally, notice that KEA does only slightly better than TF-IDF even though KEA is a supervised approach. While KEA is suitable for characterizing documents, such a supervised approach proves to be insufficient for the task of generating micropinions. It might be that the model needs a more sophisticated set of features for

Table 3: Average # of generated phrases in summary.

σ_{ss}	**Web Ngram**	**Opinosis**	**TF-IDF**	**Kea**
5	**1.00**	1.14	1.63	1.90
10	**2.91**	3.00	3.40	4.01
15	**4.53**	4.76	5.00	6.04
20	**5.85**	6.56	6.66	8.01
25	**7.36**	8.43	8.32	9.86
30	**8.74**	10.01	9.93	11.75

it to generate more meaningful summaries. Note that varying the size of training data had minimal effect on KEA.

Seen N-grams vs. System Generated N-grams. In our candidate generation approach, we form longer n-grams from shorter ones using the procedure described in Section 4. One may argue that with sufficiently redundant opinions, searching the restricted space of all *seen n-grams* may be sufficient for generating micropinion summaries and thus such a search procedure may not be necessary. We thus performed a run by forcing only seen n-grams to appear as candidate phrases. The results are shown in Table 2 under WebNgram$_{seen}$. From this, it is clear that this approach yields lower performance compared to using system generated n-grams (WebNgram), suggesting that our search algorithm actually helps discover useful new phrases. When opinions are not sufficiently redundant, observed n-grams tend to be less representative than our system generated n-grams which are constructed by combining related words and phrases (e.g. *big and clear screen* from *"..the phone has a big screen.."* and *"..the screen was clear.."*). This is one good example of why a more abstractive approach is suitable for generating such concise opinion summaries.

Well-formedness of phrases. Intuitively, a good summary phrase is one that is fairly well-formed and clearly conveys the intended meaning. Thus, a few readable phrases is more desirable than many fragmented phrases. Consider two micropinion summaries, $M = \{very\ clear, screen\ is\}$ and $M' = \{very\ clear\ screen\}$. In this example, it is obvious that M' is a more desirable summary than M. Thus, we now look into the average number of phrases generated for different summary sizes which is shown in Table 3. We can see that the WebNgram approach generates the fewest phrases for any given σ_{ss} constraint. This shows that the WebNgram phrases are longer on average (i.e. more well-formed) and also readable as previously shown by the ROUGE-2 scores and further validated by our manual evaluation. KEA seems

Table 4: Average # of opinion words in summary

σ_{ss}	WebNgram +Bias	Web Ngram	Opinosis	TF-IDF	KEA
10	2.80	2.11	2.08	2.13	1.16
20	5.48	4.19	4.56	4.01	2.33
30	7.81	6.10	6.77	6.07	3.39

Table 5: Recall scores with the use of biasing ($\sigma_{ss} = 30$).

	ROUGE-1	ROUGE-2	ROUGE-SU4
WebNgram	0.219	0.079	0.069
WebNgram+Bias	0.240	0.085	0.088
change	**9.3%**	**7.8%**	**27.9%**

Table 6: Performance of scoring components ($\sigma_{ss} = 30$).

Rouge	Recall			F-score		
	R-1	R-2	R-SU4	R-1	R-2	R-SU4
$+S_{rep}$-S_{read}	0.192	0.057	0.056	0.120	0.035	0.019
$-S_{rep}$+S_{read}	0.199	0.061	0.062	0.127	0.035	0.020
$+S_{rep}$+S_{read}	**0.219**	**0.080**	**0.073**	**0.140**	**0.046**	**0.024**

Table 7: Results of manual evaluation with $\sigma_{ss} = 25$. Score 1 (Very Poor) to 5 (Very Good).

Question	TF-IDF		WebNgram		Human	
	Avg.	Dev.	Avg.	Dev.	Avg.	Dev.
Grammaticality	2.01	0.63	4.16	0.86	4.71	0.65
Non-redundancy	2.34	0.87	3.92	0.69	4.53	0.71
Informativeness	1.67	0.71	3.22	0.76	3.61	1.01
Overall	**2.00**	**0.74**	**3.77**	**0.77**	**4.29**	**0.79**

to favor very short phrases and on average, KEA generates the most number of phrases.

Opinion coverage and effect of summary biasing. An important point to note is that each of the baselines used (KEA, Opinosis and TF-IDF) has some form of opinion biasing built-in; *KEA* with the training examples from human composed summaries, *Opinosis* through selection of phrases with certain POS tags and *TF-IDF* through selection of adjective containing n-grams. In our current model, we do not use any form of biasing to evaluate the effectiveness of our method as is. To estimate the actual coverage of opinions, we count the average number of opinion words in the summary using a general set of adjective words from General Inquirer[5] (1,614 positive and 1,982 negative). The results are reported in Table 4. Considering only the base WebNgram model and the other 3 baselines discussed earlier, Opinosis has the highest number of opinion words. However, notice that even without opinion specific refinements, WebNgram has comparable number of opinion words to that of Opinosis and TF-IDF (which have built-in biasing).

To gain insights into how opinion specific refinements can help our summaries, we explore a simple heuristics biasing. Specifically, in our final candidate selection step we only considered phrases that contain at least one adjective. This run is called **WebNgram+Bias**. From Table 4 and Table 5, it is clear that the addition of opinion biasing improves agreement with human summaries and the average number of opinion words in the summaries is also much higher than the base model. This suggests that our model can be further refined to generate task specific summaries.

Effectiveness of two-part scoring. So far, it has been assumed that both components of our scoring function, S_{rep} and S_{read} are required to generate reasonable summaries. To test this hypotheses, we generate summaries with different scoring components turned off at a given time. The ROUGE scores are reported in Table 6. Overall, the performance is lowest when S_{read} is not used (row 1). Without S_{read} it is likely that ungrammatical phrases with high representativeness scores appear as good candidates, thus resulting in poor performance. While S_{read} is important, by itself, it does not perform as well (row 2) as when used in conjunction with S_{rep} (row 3). This is because S_{read} favors highly readable phrases but these phrases may not be representative of the underlying opinions. Thus, it is clear that S_{read} and S_{rep} work in synergy to generate meaningful summaries.

Stability of parameters. It is critical to understand the effect of non-user dependent parameters in our model (namely

[5]http://www.wjh.harvard.edu/ inquirer/

σ_{read}, σ_{rep} and C), so that these parameters can be set correctly for a new dataset. The primary function of σ_{read} and σ_{rep} is to prune non-promising candidates, thus improving efficiency and as a result also ensuring minimum readability and representativeness. Without these two parameters, we will still arrive at a solution, but the time to convergence would be much longer and the results could be skewed (e.g. very representative but not readable). Figure 1 (a) and (b) show how different settings of σ_{read} and σ_{rep} affect the overall performance. The values in Figure 1 (a) are averaged across $\sigma_{rep} \in [1, 5]$ and $C \in [2, 6]$; The values in Figure 1 (b) are averaged across $\sigma_{read} \in [-4, -1]$ and $C \in [2, 6]$. Notice that these two parameters are actually stable across different values and do not affect performance except in extreme conditions (thresholds that are too high). When $\sigma_{read} \leq -1$, phrases are expected to have high readability scores from the start and this requirement is too restrictive in finding good candidates. Similarly, when $\sigma_{rep} \geq 5$, the candidates are expected to have extremely high representativeness scores at every point, and again restricting discovery of good candidates. The fact that σ_{read} and σ_{rep} do not affect performance (except when the thresholds are too high) suggests that the objective function already ensures phrases to be both representative and readable. It is thus safe to set these values to be small enough (between -2 and -4 for σ_{read}; between 1 and 4 for σ_{rep}) to ensure reasonable efficiency and meaningful summaries. Note that the low 'min' curves as seen in Figure 1 (a), (b) and (c) is caused by the extreme values, $\sigma_{rep} = 5$ and $\sigma_{read} = -1$.

The third parameter, C is a window size used to compute the representativeness score of a phrase. The requirement is that two words in a *candidate phrase* should occur within a context window of size C in the original text (see Section 3.1). Figure 1 (c) shows performance at different C values (averaged across $\sigma_{rep} \in [1, 5]$ and $\sigma_{read} \in [-4, -1]$). On average, the best performance is achieved when $C = 3$, which is quite reasonable. Intuitively, when C is large, certain important words that are spread out can now be seen in context (e.g. *the Nokia phone that I bought was cheap*). At the same time, wrong pairs of words may also be considered related and this is evident with lower performance when $C > 3$.

To further show that the suggested parameter settings would hold true on new datasets, we obtained the optimal parameter values tuned on the 100 review documents with-
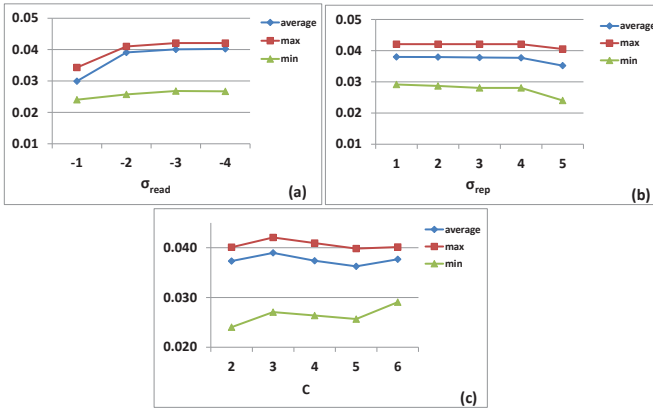
Figure 1: ROUGE-2 with varying σ_{read}, σ_{rep} and C. Labeled as (a), (b) and (c) respectively.

held to train KEA. The values are: $C = 3$, $\sigma_{read} = -4$ and $\sigma_{rep} = 4$. Note that all these values comply with the suggested settings. In fact, the σ_{rep} and C values are the same as our default settings. The only difference is that $\sigma_{read} = -4$. This new σ_{read} value on our evaluation dataset, did not show any significant difference in terms of performance (ROUGE-1 gain:-0.004, ROUGE-2 gain: +0.002). The only difference was in efficiency, which in this case was slower due to the more relaxed setting. It is thus clear, that the efficiency parameters have little effect on performance of summarization as long as the values are not too restrictive. **Manual evaluation**. To assess potential utility of the micropinion summaries to real users, a subset of the summaries were manually evaluated using the procedure described in Section 5.3. The average *grammaticality*, *non-redundancy* and *informativeness* scores (along with respective standard deviation scores) for three methods are reported in Table 7. The results on human summaries serves as an upper bound for comparison. A score *below* 3 is considered 'poor' and a score *above* 3 is considered 'good'.

Earlier, we showed that TF-IDF had the lowest ROUGE scores amongst all the approaches, indicating that the summaries may not be very useful (see Table 2). The scores assigned by the human assessors on the TF-IDF summaries agree with this conclusion. On average, TF-IDF summaries received poor scores (below 3) on all three dimensions compared to WebNgram and human summaries. WebNgram's average scores are above 3 on all dimensions and are quite close to human scores.

In terms of grammaticality, WebNgram summaries received an average score of 4.16, which means that the WebNgram phrases are mostly grammatically sound with a few exceptions. This number actually correlates well with our ROUGE-2 scores discussed earlier. Next, the average non-redundancy score of 3.92 tells us that the WebNgram phrases are fairly unique with a few cases of overlapping facts or opinions. Although our σ_{sim} setting requests a diverse set of phrases, some of the redundancies were caused by the different ways in which the same information can be conveyed. For example, the phrases *Excellent sound quality* and *Great audio* may seem different but actually mean the same thing. Such differences can be resolved with the use of opinion or phrase normalization or clustering of similar words and concepts prior to summarization. While the average informativeness scores of WebNgram and humans are quite close, notice that these scores are just slightly above

3. This suggests that the informativeness aspect is rather subjective and a score above 3 is actually quite encouraging. The WebNgram summaries that had informativeness scores between 3-4 mostly had meaningful and representative phrases along with some false positives that did not have any real information (e.g *I bought this for Christmas*). The informativeness aspect can be improved in different ways, one of which is through a much stricter selection of phrases. This is something we would like to study in detail in the future.

7. CONCLUSION

In this paper, we proposed an unsupervised summarization approach to generate ultra-concise summaries of opinions. We frame the problem as an optimization problem with an objective function capturing *representativeness* and *readability* along with constraints that ensures *compactness* of a generated summary. We use modified mutual information to capture representativeness and model readability with an n-gram language model. We propose a heuristic search algorithm to solve the optimization problem efficiently. Our evaluation using a set of user reviews shows that our summaries can convey essential information with minimal word usage and is more effective than other competing methods. Human assessors found the summaries to be very readable, fairly non-redundant and informative.

Compared with existing approaches, our approach is unsupervised, lightweight and practical with no reliance on any linguistic annotations (e.g. POS tagging or syntactic parsing) and is not designed or optimized for a specific domain. It only uses the existing text and a web scale n-gram model to generate meaningful summaries. Also, since we do not use any opinion specific refinements, our idea and approach can be easily used to generate ultra-concise summaries in other application areas such as Twitter comments, Facebook status updates, blogs, user comments and so on. In the future, we would like to evaluate our method in these other domains and study further refinements of the summarization framework.

8. ACKNOWLEDGMENTS

9. REFERENCES

[1] S. R. K. Branavan, H. Chen, J. Eisenstein, and R. Barzilay. Learning document-level semantic properties from free-text annotations. In *In Proceedings of ACL*, pages 263–271, 2008.

[2] G. Carenini and J. C. K. Cheung. Extractive vs. nlg-based abstractive summarization of evaluative text: the effect of corpus controversiality. In *Proceedings of the Fifth International Natural Language Generation Conference*, INLG '08, pages 33–41, Stroudsburg, PA, USA, 2008. Association for Computational Linguistics.

[3] G. Carenini, R. Ng, and A. Pauls. Multi-document summarization of evaluative text. In *Proceedings of EACL '06*, pages 305–312, 2006.

[4] H. T. Dang. Overview of DUC 2005. In *Document Understanding Conference*, 2005.

[5] K. Filippova. Multi-sentence compression: finding shortest paths in word graphs. In *Proceedings of the 23rd International Conference on Computational Linguistics*, COLING '10, pages 322–330, Stroudsburg, PA, USA, 2010. Association for Computational Linguistics.

[6] E. Frank, G. W. Paynter, I. H. Witten, and C. Gutwin. Domainspecific keyphrase extraction. In *Proc. Sixteenth International Joint Conference on Artificial Intelligence*, pages 668–673. Morgan Kaufmann Publishers, 1999.

[7] K. Ganesan, C. Zhai, and J. Han. Opinosis: A graph based approach to abstractive summarization of highly redundant opinions. In *Proceedings of the 23rd International Conference on Computational Linguistics (COLING)*, Beijing, China, 2010.

[8] N. Gupta, G. Di Fabbrizio, and P. Haffner. Capturing the stars: predicting ratings for service and product reviews. In *Proceedings of the NAACL HLT 2010 Workshop on Semantic Search*, SS '10, pages 36–43, Stroudsburg, PA, USA, 2010. Association for Computational Linguistics.

[9] M. Hu and B. Liu. Mining and summarizing customer reviews. In *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '04, pages 168–177, New York, NY, USA, 2004. ACM.

[10] H. D. Kim and C. Zhai. Generating comparative summaries of contradictory opinions in text. In *Proceeding of the 18th ACM conference on Information and knowledge management*, CIKM '09, pages 385–394, New York, NY, USA, 2009. ACM.

[11] C.-Y. Lin. Rouge: a package for automatic evaluation of summaries. In *Proceedings of the Workshop on Text Summarization Branches Out (WAS 2004), Barcelona, Spain*, 2004.

[12] B. Liu. *Web data mining; Exploring hyperlinks, contents, and usage data*. Springer, 2006.

[13] B. Liu, M. Hu, and J. Cheng. Opinion observer: analyzing and comparing opinions on the web. In *WWW '05: Proceedings of the 14th international conference on World Wide Web*, pages 342–351, 2005.

[14] Y. Lu, C. Zhai, and N. Sundaresan. Rated aspect summarization of short comments. In *18th International World Wide Web Conference (WWW2009)*, April 2009.

[15] O. Medelyan and I. H. Witten. Domain-independent automatic keyphrase indexing with small training sets. *J. Am. Soc. Inf. Sci. Technol.*, 59:1026–1040, May 2008.

[16] B. Pang and L. Lee. Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales. In *Proceedings of the ACL*, pages 115–124, 2005.

[17] B. Pang, L. Lee, and S. Vaithyanathan. Thumbs up? Sentiment classification using machine learning techniques.

[18] M. J. Paul, C. Zhai, and R. Girju. Summarizing contrastive viewpoints in opinionated text. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, EMNLP '10, pages 66–76, Stroudsburg, PA, USA, 2010. Association for Computational Linguistics.

[19] J. Pei, J. Han, B. Mortazavi-Asl, J. Wang, H. Pinto, Q. Chen, U. Dayal, and M.-C. Hsu. Mining sequential patterns by pattern-growth: The prefixspan approach. *IEEE Trans. on Knowl. and Data Eng.*, 16:1424–1440, November 2004.

[20] D. Radev, H. Jing, and M. Budzikowska. Centroid-based summarization of multiple documents: Sentence extraction, utility-based evaluation, and user studies. In *ANLP/NAACL Workshop on Summarization*, pages 21–29, 2000.

[21] R. Real and J. M. Vargas. The Probabilistic Basis of Jaccard's Index of Similarity. *Systematic Biology*, 45(3):380–385, 1996.

[22] B. Snyder and R. Barzilay. Multiple aspect ranking using the good grief algorithm. In *Proceedings of HLT-NAACL '07*, pages 300–307, 2007.

[23] E. Terra and C. L. A. Clarke. Frequency estimates for statistical word similarity measures. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology - Volume 1*, NAACL '03, pages 165–172, Stroudsburg, PA, USA, 2003. Association for Computational Linguistics.

[24] I. Titov and R. Mcdonald. A joint model of text and aspect ratings for sentiment summarization. In *Proceedings of ACL-08: HLT*, pages 308–316, Columbus, Ohio, June 2008. Association for Computational Linguistics.

[25] T. Tomokiyo and M. Hurst. A language model approach to keyphrase extraction. In *Proceedings of the ACL 2003 workshop on Multiword expressions: analysis, acquisition and treatment - Volume 18*, pages 33–40, Morristown, NJ, USA, 2003. Association for Computational Linguistics.

[26] P. D. Turney. Thumbs up or thumbs down?: semantic orientation applied to unsupervised classification of reviews. In *Proceedings of ACL '02*, pages 417–424, Stroudsburg, PA, USA, 2002. Association for Computational Linguistics.

[27] H. Wang, Y. Lu, and C. Zhai. Latent aspect rating analysis on review text data: a rating regression approach. In *Proceedings of KDD '10*, pages 783–792, New York, NY, USA, 2010. ACM.

[28] K. Wang, C. Thrasher, E. Viegas, X. Li, and B.-j. P. Hsu. An overview of microsoft web n-gram corpus and applications. In *Proceedings of the NAACL HLT 2010 Demonstration Session*, pages 45–48, Los Angeles, California, June 2010. Association for Computational Linguistics.

[29] T. Wilson, J. Wiebe, and P. Hoffmann. Recognizing contextual polarity in phrase-level sentiment analysis. In *Proceedings of the conference on Human Language Technology and Empirical Methods in Natural Language Processing*, HLT '05, pages 347–354, Stroudsburg, PA, USA, 2005. Association for Computational Linguistics.

[30] I. H. Witten, G. W. Paynter, E. Frank, C. Gutwin, and C. G. Nevill-Manning. Kea: practical automatic keyphrase extraction. In *Proceedings of the fourth ACM conference on Digital libraries*, DL '99, pages 254–255, New York, NY, USA, 1999. ACM.