Opinion Feature Extraction Using Class Sequential Rules

Minqing Hu and Bing Liu

Department of Computer Science
University of Illinois at Chicago
851 South Morgan Street
Chicago, IL 60607-7053
{mhu1, liub}@cs.uic.edu

Abstract

The paper studies the problem of analyzing user comments and reviews of products sold online. Analyzing such reviews and producing a summary of them is very useful to both potential customers and product manufacturers. By analyzing reviews, we mean to extract features of products (also called opinion features) that have been commented by reviewers and determine whether the opinions are positive or negative. This paper focuses on extracting opinion features from Pros and Cons, which typically consist of short phrases or incomplete sentences. We propose a language pattern based approach for this purpose. The language patterns are generated from Class Sequential Rules (CSR). A CSR is different from a classic sequential pattern because a CSR has a fixed class (or target). We propose an algorithm to mine CSR from a set of labeled training sequences. To perform extraction, the mined CSRs are transformed into language patterns, which are used to match Pros and Cons to extract opinion features. Experimental results show that the proposed approach is very effective.

Introduction

The Web has dramatically changed the way that consumers expressing their opinions. They can now post reviews of products at merchant sites (e.g., amazon.com and c|net.com), dedicated review sites (e.g., epininons.com), Internet forums and blogs. These reviews provide excellent sources of consumer opinions on products, which are very useful to both potential customers and product manufacturers. Techniques are now being developed to exploit these sources to help companies and individuals to gain such information effectively and easily (e.g., Hu and Liu 2004).

In this paper, we focus on consumer reviews of products, which are of very similar nature to blogs with ungrammatical sentences, incomplete sentences (sentence fragments), short phrases, and missing punctuations.

There are three main review formats on the Web.

Format (1) - Pros and Cons: The reviewer is asked to describe Pros and Cons separately. C|net.com uses this format.

Format (2) - Pros, Cons and detailed review: The reviewer is asked to describe Pros and Cons separately and also write a detailed review. epinions.com and MSN uses this format.

Format (3) - free format: The reviewer writes freely, i.e., no separation of Pros and Cons. Amazon.com uses this format.

In this paper, we propose to analyze and summarize reviews of format (2). We aim to identify product features that have been commented on by customers in a set of reviews and use them to summarize the reviews. To summarize reviews, we display the number of positive and negative reviews for each product feature, which show whether the customers like or dislike the feature. Note that for reviews of format (2), opinion orientations (positive or negative) of features are known as Pros and Cons are separated.

In (Hu and Liu 2004), several techniques were proposed to identify both product features and their opinion orientations from reviews of format (3). For format (3) (and also (1)), reviewers typically use full sentences. However, for format (2), Pros and Cons tend to be very brief. For example, under Cons, one may only write: "heavy, bad picture quality, battery life too short", which are elaborated in the detailed review. (Liu, Hu and Cheng, 2005) proposed an initial method to extract product features from Pros and Cons based on association rules. However, association rule mining is not suitable for this task because association rule mining is unable to consider the sequence of words, which is very important in natural language texts. Thus, many complex ad hoc post-processing methods are used in order to find patterns to extract features.

In this work, we propose a more principled mining method based on sequential pattern mining. In particular, we mine a special kind of sequential patterns called Class Sequential Rules (CSR). As its name suggests, the sequence of words is considered automatically in the mining process. Unlike standard sequential pattern mining, which is unsupervised, we mine sequential rules with some fixed targets or classes. Thus, the new method is supervised. To our knowledge, this is the first work that mines and uses such kind of rules.

The mined CSRs are used to extract product features from Pros and Cons in format (2). Note that we do not analyze detailed reviews in format (2) as they are elaborations of Pros and Cons. Analyzing short segments in Pros and Cons produce more accurate results. Our experimental results show that the proposed method is highly effective.

Related Work

In (Hu and Liu 2004), some methods are proposed to analyze customer reviews of format (3). However, since reviews of format (3) are usually complete sentences, the techniques in (Hu and Liu 2004) are thus not suitable for Pros and Cons of format (2). The work of [Popescu and Etzioni 2005] also

works on complete sentences and thus not suitable for sentence fragments and short phrase in Pros and Cons. In (Liu, Hu and Cheng, 2005), a method is also proposed to extract product feature from Pros and Cons of format (2). However, as we discussed in the Introduction section, the method is very complex and rather ad hoc because association rules cannot naturally capture word relations.

(Morinaga *et al*, 2002) compares different products in a category through search to find the products reputation. It does not analyze reviews, and does not identify product features. Below, we present some other related research.

Terminology Finding: There are basically two techniques for terminology finding: symbolic approaches that rely on noun phrases, and statistical approaches that exploit the fact that words composing a term tend to be found close to each other and reoccurring (Bourigault, 1995, Daille, 1996, Jacquemin, and Bourigault, 2001, Justeson and Katz, 1995). However, using noun phrases tends to produce too many non-terms, while using reoccurring phrases misses many low frequency terms, terms with variations, and terms with only one word. As shown in (Hu and Liu 2004) using the existing terminology finding system FASTR (FASTR) produces very poor results. Furthermore, using noun phrases is not sufficient for finding product features. We also need to consider other language components (e.g., verbs and adjectives).

Sentiment Classification: Sentiment classification classifies opinion texts or sentences as positive or negative. Work of (Hearst, 1992) on classification of entire documents uses models inspired by cognitive linguistics. (Das and Chen, 2001) uses a manually crafted lexicon in conjunction with several scoring methods to classify stock postings. (Tong, 2001) generates sentiment timelines as it tracks online discussions about movies.

(Turney, 2002) applies a unsupervised learning technique based on mutual information between document phrases and the words "excellent" and "poor" to find indicative words of opinions for classification. (Pang, Lee and Vaithyanathan, 2002) examines several supervised machine learning methods for sentiment classification of movie reviews. (Dave, Lawrence and Pennock, 2003) also experiments a number of learning methods for review classification. (Agrawal *et al*, 2003) finds that supervised sentiment classification is inaccurate. They proposed a method based on social network for the purpose. However, social networks are not applicable to customer reviews. (Hatzivassiloglou and Wiebe, 2000) investigates sentence subjectivity classification. Other related works include (Nasukawa and Yi, 2003, Nigam and Hurst 2004, Riloff and Wiebe, 2003, Wilson, Wiebe and Hwa, 2004, Yu, and Hatzivassiloglou, 2003).

Our work differs from sentiment and subjectivity classification as they do not identify features commented on by customers or what customers praise or complain about. Thus, we solve a related but different problem.

Problem Statement

We first describe the problem statement, and then discuss the new automatic technique for identifying product features from Pros and Cons in reviews of format (2).

Let P be a product and $R = \{r_1, r_2, \ldots, r_k\}$ be a set of reviews of P. Each review r_j consists of a list of Pros and Cons.

Definition (product feature): A *product feature f* in r_j is an attribute/component of the product that has been commented on in r_j. If f appears in r_j, it is called an *explicit feature* in r_j. If f does not appear in r_j but is implied, it is called an *implicit feature* in r_j.

For example, "battery life" in the following opinion segment is an explicit feature:

"Battery life too short"

"Size" is an implicit feature in the following opinion segment as it does not appear in each sentence but it is implied:

"Too small"

Figure 1 shows a review of format (2). Pros and Cons are separated and very brief. We do not study full reviews as they basically elaborate on Pros and Cons.

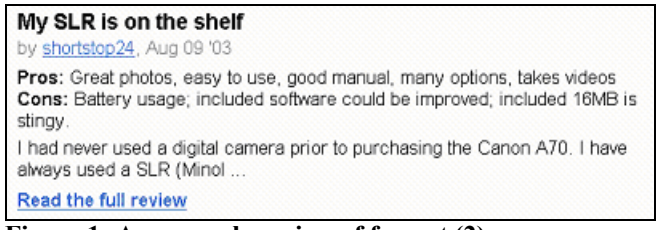
My SLR is on the shelf
by shortstop24, Aug 09 '03
Pros: Great photos, easy to use, good manual, many options, takes videos
Cons: Battery usage; included software could be improved; included 16MB is stingy.
I had never used a digital camera prior to purchasing the Canon A70. I have always used a SLR (Minol ...
Read the full review

Figure 1: An example review of format (2)

The task: Our objective in this paper is to find all the explicit and implicit product features on which reviewers have expressed their (positive or negative) opinions.

Class Sequential Rules Mining

We propose a supervised sequential pattern mining method to find language patterns to identify opinion (product) features from Pros and Cons.

Let $I = \{i_1, i_2, \ldots, i_n\}$ be a set of all items and $C = \{c_1, c_2, \ldots, c_m\}$ be a set of class items and $C \subset I$. A sequence is an ordered list of items, denoted by $<i_1 i_2 \ldots i_l>$. A sequence with length l is called an l-**sequence**. A sequence $s_1 = <a_1 a_2 \ldots a_n>$ is called a **subsequence** of another sequence $s_2 = <b_1 b_2 \ldots b_m>$ or s_2 a **supersequence** of s_1, if there exist integers $1 \leq j_1 < j_2 < \ldots < j_{n-1} \leq j_n$ such that $a_1 = b_{j1}$, $a_2 = b_{j2}$, ..., $a_1 = b_{jn}$.

A **sequence database** S is a set of tuples $<sid, s>$, where sid is a **sequence_id** and s a sequence. A tuple is said to **contain** a sequence s_i, if s_i is a subsequence of s. A **class sequential rule** (**CSR**) is an implication of the form

$$X \rightarrow Y,$$

where X is a sequence $<s_1 x_1 s_2 x_2 \ldots x_l s_{r+1}>$ ($s_i = <>$ or $<i_1 i_2 \ldots i_k>$ and $i_m \notin C$, and x_i *denotes a possible class at this position*, $x_i \in I$, for $1 \leq i \leq l$) and Y is a sequence $<s_1 c_{k1} s_2 c_{k2} \ldots c_{kr} s_{l+1}>$ ($c_{ki} \in C$, for $1 \leq i \leq r$). The *support* of a CSR csr_i in a sequence database S is the number of tuples in the database containing Y. The *confidence* of a CSR csr_i is the support of Y divided by the support of X and Y. A tuple $<sid, s>$ is said to **cover** a CSR csr_i, if X is a subsequence of s. A tuple is said to **contain** a CSR csr_i, if Y is a subsequence of s.

Table 1 gives an example sequence database which has 5 tuples, with $\boldsymbol{c_1}$ and $\boldsymbol{c_2}$ denoting the classes. We have a CSR

$<<ab>x<gh>> \rightarrow <<ab>\mathbf{c_1}<gh>>$ with support of 2 and confidence of 2/3, as sequence 10 and 50 contains the rule while sequence 10, 20 and 50 covers the rule.

Given a sequence database, minimum support and minimum confidence as thresholds, class sequential rule mining finds the complete set of class sequential rules in the database. In this paper, we mine the class sequential rules that have product features as class items.

sequence_id	sequence
10	$<abd\mathbf{c_1}gh>$
20	$<abeghk>$
30	$<\mathbf{c_2}kea>$
40	$<d\mathbf{c_2}kb>$
50	$<ab\mathbf{c_1}fgh>$

Table 1. An example of sequence database

ClassPrefix-Span: Class Sequential Rules Mining

We now present the algorithm *ClassPrefix-Span* to mine class sequential rules. Although there are several efficient sequential pattern mining algorithms, none of them addresses the specific problem of mining class sequential rules. Here we adapt the pattern growth method in (Pei *et al*, 2004) for the task, the general idea is outlined as follows: *as we are only interested in patterns that contain classes, we first find the patterns that have classes as suffix. Then taking the generated patterns as prefix, we can find all the class sequential rules by pattern growth.*

The algorithm recursively projects a sequence database into a set of smaller databases associated with the prefix pattern mined so far, and then mines locally frequent patterns in each projected database.

Let us examine the proposed approach for mining CSRs based on our running example.

Example. For the same sequence database *S* in Table 1 with minimum support = 2, CSRs in *S* can be mined in the following steps:

1. **Divide search space for each class and find length-1 patterns.** The complete set of patterns can be partitioned into the following three subsets according to the two classes: 1) the ones with class c_1 {sequence 10 and 50}, 2) the ones with class c_2 {sequence 30 and 40}, and 3) the ones without any class {sequence 20}. The length-1 patterns are: $<c_1>$:2, $<c_2>$:2 (2 is the support).
2. **Find subsets of suffix patterns for each class**. The subsets of patterns that have class as suffixes can be mined by constructing the corresponding sets of projected databases and mining each recursively. The projected databases as well as suffix patterns found in them are listed in Table 2.
   a. **Find patterns with suffix** $<c_1>$. Only subsequences ending with $<c_1>$ should be considered. For example, in $<abd\mathbf{c_1}gh>$, only the subsequence $<abd>$ should be considered for mining patterns with suffix $<c_1>$.

      The sequences in *S* are projected with regards to $<c_1>$ to form the projected database, which consists of two prefix sequences: $<abd>$ and $<ab>$.

      By scanning the projected database for $<c_1>$, its locally frequent items are *a*:2 and *b*:2. Thus, all the length-2 patterns suffixed with $<c_1>$ are found, and they are: $<ac_1>$ and $<bc_1>$.

      Recursively, all patterns with suffix $<c_1>$ can be partitioned into two subsets: 1) those suffixed with $<ac_1>$, and 2) those with $<bc_1>$. These subsets can be mined by constructing respective projected databases and mining each recursively as follows:

      i. The projected database for $<ac_1>$ consists of no subsequence. Thus the processing of this projected database terminates.

      ii. The projected database for $<bc_1>$ consists of subsequences ending with $<bc_1>$: $<a>$ and $<a>$. Recursively mining the projected database returns one pattern: $<abc_1>$. It forms the complete set of patterns suffixed with $<bc_1>$.

   b. **Find patterns with suffix** $<c_2>$. This can be done by constructing projected databases and mining them respectively. The projected databases and the patterns found are shown in Table 2.

Class	Sequence	projected db	Suffix patterns
c_1	$<abd\mathbf{c_1}gh>$, $<ab\mathbf{c_1}fgh>$	$<abd>$, $<ab>$	$<c_1>$, $<ac_1>$, $<bc_1>$, $<abc_1>$
c_2	$<\mathbf{c_2}kea>$, $<d\mathbf{c_2}kb>$	$<>$, $<d>$	$<c_2>$
no_class	$<abeghk>$	-	-

Table 2. Projected databases and patterns

3. **Find patterns with generated suffix patterns as prefixes**. The sequence database can be partitioned into five subsets according to the five generated prefixes: 1) the ones with prefix $<c_1>$, 2) the ones with prefix $<ac_1>$,..., and 5) the ones with prefix $<c_2>$. The projected databases and the patterns are listed in Table 3.

prefix	projected db	Prefix patterns
$<c_1>$	$<gh>$, $<fgh>$	$<c_1g>$, $<c_1h>$,$<c_1gh>$
$<ac_1>$	$<gh>$, $<fgh>$	$<ac_1g>$, $<ac_1h>$, $<ac_1gh>$
$<bc_1>$	$<gh>$, $<fgh>$	$<bc_1g>$, $<bc_1h>$,$<bc_1gh>$
$<abc_1>$	$<gh>$, $<fgh>$	$<abc_1g>$, $<abc_1h>$, $<abc_1gh>$
$<c_2>$	$<kea>$, $<kb>$	$<c_2k>$

Table 3. Projected databases and patterns

   a. **Find patterns with prefix** $<c_1>$. Only the subsequence prefixed with the occurrence of $<c_1>$ will be considered. The projected database for $<c_1>$ thus includes two suffix sequences: $<gh>$ and $<fgh>$. Note that the current projected database for $<c_1>$ is different from the database when projected for generating patterns in step 2.a. We are now projecting forwards but previously we projected backwards.

      By scanning the projected database for $<c_1>$, its locally frequent items are *g*:2 and *h*:2. Thus, all the length-2 patterns prefixed with $<c_1>$ are found, and they are: $<c_1g>$ and $<c_1h>$.

Recursively, all patterns with prefix $<c_1>$ can be partitioned into two subsets: 1) those prefixed with $<c_1g>$, and 2) those with $<c_1h>$. These subsets can be mined by constructing respective projected databases and mining each recursively, i.e., generating length-3 pattern $<c_1gh>$.

b. **Find patterns with prefix** $<ac_1>$, $<bc_1>$, $<abc_1>$, and $<c_2>$. This can be done by following the same procedure as finding patterns with prefix $<c_1>$.

4. **Compute the confidence of CSRs**. The sequence database *S* is scanned to compute the confidence of the rules.

The algorithm is presented as follows:

Algorithm: *ClassPrefix-Span*($\{c_1, c_2, ..., c_n\}$, *S*)
Input: $\{c_1, c_2, ..., c_n\}$ is the set of classes of interests, *S* is the sequence database.
Output: complete set of class sequential rules.
Method:

1. Let *csr_set* = {};
2. Scan *S* once, for each class *c* in $\{c_1, c_2, ..., c_n\}$
   (a) construct projected database $S|_c$ backwards;
   (b) call *Pre-Suf-fixSpan*(*c*, 1, $S|_c$, *"backward"*).
3. For each *csr* in *csr_set*
   (a) construct projected database $S|_{csr}$ forwards;
   (b) let *l* = length of *csr*;
   (c) call *Pre-Suf-fixSpan*(*csr, l,* $S|_{csr}$, *"forward"*).
4. Scan *S* once, for each *csr* in *csr_set*, compute confidence;
5. Return *csr_set*.

Subroutine *Pre-Suf-fixSpan*(*s, l,* $S|_s$, *d, csr_set*)
Input: *s* is a class sequential rule; *l* is the length of *s*; $S|_s$ is the projected database of *s*; *d* is the direction of construction of projected database: *csr_set* is for storing the generated *csr*s.
Method:

1. Scan $S|_s$ once, find each frequent item *a*, such that
   (a) *a* can be appended to the last element of *s* to form a class sequential rule if *d = "forward"*; or
   (b) *a* can be appended to the first element of *s* to form a class sequential rule if *d = "backward"*;
2. For each frequent item *a*, append to *s* to form a class sequential rule *s'*, and insert *s'* to *csr_set*.
3. For each *a'*
   (a) construct projected database $S|_{s'}$ backwards if *d = "backward"*, forwards if *d = "forward"*;
   (b) call *Pre-Suf-fixSpan*(*s', l*+1, $S|_{s'}$, *d, csr_set*).

ClassPrefix-Span is the main algorithm for generating class sequential rules. In step 2, we first construct the projected databases for each class backwards, and then use the projected databases to mine patterns of the format "i_1i_2 …*class*", that is, the patterns with class as suffix. The growth direction is backward as we first fix the last item (class item) in the pattern, then we grow the pattern by appending item in front of the pattern each time. In step 4, the mined patterns so far are taken as prefix, and we grow the patterns forward by appending an item at the end of the pattern each time. The steps will generate patterns of the form "i_1i_2…class $i_li_{l+1}...i_n$". In step 4, the database is scanned once again, to count the coverage of each csr and to calculate the confidence.

Pre-Suf-fixSpan is the function that grows patterns. Each time it appends one frequent item found from current projected database to the given pattern to form a new pattern (step 1 and 2). It then recursively constructs projected database for each frequent item and mines new patterns (step 3). Due to space limitations, we do not elaborate the process of construction of projected database, which is similar to that in (Pei *et al*, 2004). The major difference is that we can project the database in two directions.

Mining CSRs for Feature Extraction

In this work, we aim to find CSRs with the following target classes: <NN> [feature], <JJ> [feature], <VB> [feature] and <RB> [feature], which allow us to extract various types of product features (<NN>: nouns, <VB>: verbs, <JJ>: adjectives, and <RB>: adverbs).

Our approach is based on the following observation:

Each sentence segment in Pros and Cons contains at most one product feature. Sentence segments are separated by ',', '.', ';', '-', '&', 'and', and 'but'.

For example, "Pros" in Figure 1 can be separated into 5 segments.

great photos	<photo>
easy to use	<use>
good manual	<manual>
many options	<option>
takes videos	<video>

"Cons" in Figure 3 can be separated into 3 segments:

battery usage	<battery>
included software could be improved	<software>
included 16MB is stingy	<16MB>⇒ <memory>

We can see that each segment describes a product feature on which the reviewer has expressed an opinion (the last two can be seen as full sentences). The product feature for each segment is listed within <>. Notice that <16MB> is a value of feature <memory>, which is an implicit feature as it does not appear in the sentence segment.

Another important point to note is that a feature may not be a noun or noun phrase, which is used in (Hu and Liu 2004). Verbs may be features as well, e.g., "use" in "easy to use". Of course, we can also use its corresponding noun as the feature, e.g., "usage" or simply "use".

Given a manually labeled training review set, we perform the following preprocessing before mining CSRs:

1. Perform Part-Of-Speech (POS) tagging and remove digits and some punctuations: We use the NLProcessor linguistic parser (NLProcessor, 2000) to generate the POS tag of each word. POS tagging is crucial as it allows us to generate general language patterns.

   We remove digits in sentences, e.g., changing "16MB" to "MB". Digits often represent concepts that are too specific to be used in rule discovery, which aims to generalize. We use two examples from above to illustrate the results of this step:

   "<NN> Battery <NN> usage"

   "<VB> included <NN> MB <VB>is <JJ> stingy"

   <NN> indicates a noun, <VB> a verb, and <JJ> an ad-

jective.

2. Replace the actual feature words in a sentence with [feature]: This replacement is necessary because different products have different features. The replacement ensures that we can find general language patterns which can be used for any product feature. After replacement, the above two examples become:
   "<NN> [feature] <NN> usage"
   "<VB> included <NN> [feature] <VB> is <JJ> stingy"
   Note that "MB" is also replaced with [feature] as it indicates an implicit feature.
   It is possible that a feature may contain more than one word, e.g., "auto mode stinks", which will be changed to
   "<NN> [feature] <NN> [feature] <VB> stinks"
3. Use n-gram to produce shorter segments from long ones: For example, "<VB> included <NN> [feature] <VB> is <JJ> stingy" will generate 2 3-gram segments:
   "<JJ> included <NN> [feature] <VB> is"
   "<NN> [feature] <VB> is <JJ> stingy"
   We only use 3-grams (3 words with their POS tags) here, which works well. The reason for using n-gram rather than full sentences is because most product features can be found based on local information and POS tagging. Using long sentences tend to generate a large number of spurious rules.
4. Perform word stemming: This is performed as in information retrieval tasks to reduce a word to its stem.

After the four-step pre-processing and labeling (tagging), the resulting sentence (3-gram) segments are saved in a file (called a *transaction file*) for the generation of class sequential patterns. In this file, each line contains one processed (labeled) sentence segment. We then use class sequential pattern mining to find all language patterns. We use 1% as the minimum support, but do not set minimum confidence. As the patterns generated are small in number, further pattern pruning by setting a minimum confidence may cause some review segments not covered by any pattern. Experimental result also indicates that using minimum confidence decreases the recall and precision. Two example rules are given below (we omit supports and confidences).

(a) <NN> x <NN> x → <NN> [feature] <NN> [feature]
(b) <JJ> easy to <VB> x → <JJ> easy to <VB> [feature]

We observe that both POS tags and words may appear in rules. We also note that when using pattern (b) for feature extraction, it may cause ambiguity, e.g., is <JJ> the POS tag for "easy", or any word in front of "easy"? To tackle this problem, we do post-processing to reassemble the CSRs into the following four new rules (we use pattern (b) as example),

(1) <JJ> -1, -1 easy, -1 to, <VB> x → <JJ> -1, -1 easy, -1 to, <VB> [feature]
(2) <JJ> easy, -1 to, <VB> x → <JJ> easy, -1 to, <VB> [feature]
(3) <JJ> -1, -1 easy, -1 to, <VB> -1, -1 x → <JJ> -1, -1 easy, -1 to, <VB> -1, -1 [feature]
(4) <JJ> easy, -1 to, <VB> -1, -1 x → <JJ> easy, -1 to, <VB> -1, -1 [feature]

Note that in the new rules, we require each word to have its corresponding POS tag in front of it. In the case that there is no POS tag attached with the word, we use –1 to indicate the "do not care" situation. Similarly, –1 is used when we do not care about the word but only the word type. As shown in the above example, each POS tag together with the followed word/[feature] refers to one word in a sentence (separated by comma). We count support and confidence for each new rule by scanning the sequence database again. In this way, we can have a set of language rules for feature extraction without ambiguity.

Extraction of Product Features

The resulting language rules are used to identify product features from new reviews after POS tagging. A few situations need to be handled.

1. A generated rule does not necessarily require matching a part of a sentence segment with the same length as the rule. In other words, we allow 'gaps' for pattern matching. For example, rule (right-hand-side only) "<NN> [feature], <NN> -1" can match the segment "size of printout". This is achieved by allowing user to set a value for the maximum length that a pattern could expand. We also allow user to set the maximum length of review segment that a pattern should be applied. These two values enable a user expert to refine the patterns for better extracting product features. However, in our experiments reported below, we did set any of these values, i.e., no manual involvement.
2. If a sentence segment satisfies multiple rules, we search for a matching one in the following orders: rules of class <NN> [feature], then <JJ> [feature], <VB> [feature] and lastly <RB> [feature]. And for rules of each class, we select the rule that gives the highest confidence as higher confidence indicates higher predictive accuracy. The reason for this ordering is because as we observed that the noun features appear more frequently than other types.
3. For those sentence segments that no rule applies, we use nouns or noun phrases produced by NLProcessor as features if such nouns or noun phrases exist.

Note that our rule mining method does not apply to cases that a segment only has a single word, e.g., "heavy" and "big". In this case, we treat these single words as features.

Experiment Results

We now evaluate the proposed automatic technique to see how effective it is in identifying product features from Pros and Cons in reviews of format (2).

We use the same data as [Liu, Hu and Cheng, 2005) in our experiment. The data consists of a training set and a testing set. The training set has Pros and Cons of ten products. The test set has Pros and Cons of five (different) products. Using the rules discovered from the training set, we extract features from the test set.

We use recall (r) and precision (p) to evaluate the results,

$$r = \frac{\sum_{i=1}^{n} EC_i}{\sum_{i=1}^{n} C_i}, \text{ and } p = \frac{\sum_{i=1}^{n} EC_i}{\sum_{i=1}^{n} E_i},$$

where n is the total number of reviews of a particular product, EC_i is the number of extracted features from review i that are

correct, C_i is the number of actual features in review i, E_i is the number of extracted features from review i. This evaluation is based on the result of every review as it is crucial to extract features correctly from every review.

We generate language patterns and product features separately for Cons and Pros as this produces better results. Table 4 shows the results. With recall averages at 0.889 for Pros and 0.809 for Cons, it shows that the proposed extraction using CSRs is highly effective. Table 5 compares the proposed technique of using CSRs with the technique of using association rules in (Liu, Hu and Cheng, 2005). From the two tables, we can see that the proposed technique generates comparable results as the association rules. However, feature extraction using association rules needs a lot of extra post-processing and manual involvement as association rule mining is unable to consider the sequence of words, which is very important for natural language texts. The proposed feature extraction using sequential pattern mining is thus a more principled technique.

	Pros		Cons	
	recall	prec	Recall	prec
data1	0.862	0.857	0.865	0.794
data2	0.937	0.937	0.824	0.806
data3	0.817	0.817	0.730	0.741
data4	0.919	0.914	0.745	0.708
data5	0.911	0.904	0.883	0.900
Avg.	**0.889**	**0.886**	**0.809**	**0.790**

Table 4: Recall and precision results of CSRs

	Pros		Cons	
	recall	Prec.	recall	Prec.
data1	0.922	0.876	0.850	0.798
data2	0.894	0.902	0.860	0.833
data3	0.825	0.825	0.846	0.769
data4	0.942	0.922	0.681	0.657
data5	0.930	0.923	0.881	0.897
Avg.	**0.902**	**0.889**	**0.824**	**0.791**

Table 5: Recall and precision results of association rules

Conclusions and Future Work

Analyzing reviews on the Web has many applications. It is not only important for individual consumers, but also important for product manufacturers. In this paper, we focused on one type of product reviews, i.e., Pros and Cons expressed as short phrases or sentence segments. Our objective was to extract opinion (product) features that have been commented on by consumers. As the method in (Liu, Hu and Cheng, 2005) is very ad hoc, we proposed a more appropriate mining method called class sequential rule mining to perform the task which captures the sequential relationships of words in sentences. In our future works, we will further improve the results and also study how to use the proposed method to analyze reviews of full sentences.

References

Agrawal, R., Rajagopalan, S., Srikant, R., Xu, Y. Mining newsgroups using networks arising from social behavior. *WWW'*03.

Bourigault, D. Lexter: A terminology extraction software for knowledge acquisition from texts. *KAW'95*, 1995.

Bunescu, R., and Mooney, R. Collective Information Extraction with Relational Markov Networks. *ACL-2004*, 2004.

Daille, B. Study and Implementation of Combined Techniques for Automatic Extraction of Terminology. *The Balancing Act: Combining Symbolic and Statistical Approaches to Language*. MIT Press, 1996.

Das, S. and Chen, M., Yahoo! for Amazon: Extracting market sentiment from stock message boards. *APFA'01,* 2001.

Dave, K., Lawrence, S., and Pennock, D. Mining the Peanut Gallery: Opinion Extraction and Semantic Classification of Product Reviews. *WWW'03*, 2003.

FASTR. http://www.limsi.fr/Individu/jacquemi/FASTR/

Fellbaum, C. *WordNet: an Electronic Lexical Database*, MIT Press, 1998.

Freitag, D and McCallum, A. Information extraction with HMM structures learned by stochastic optimization. *AAAI-00*, 2000.

Hatzivassiloglou, V. and Wiebe, J. Effects of adjective orientation and gradability on sentence subjectivity. *COLING'00*, 2000.

Hearst, M, Direction-based Text Interpretation as an Information Access Refinement. In P. Jacobs, editor, *Text-Based Intelligent Systems*. Lawrence Erlbaum Associates, 1992.

Hu, M and Liu, B. "Mining and summarizing customer reviews". *KDD-04*, 2004.

Jacquemin, C., and Bourigault, D. Term extraction and automatic indexing. In R. Mitkov, editor, *Handbook of Computational Linguistics*. Oxford University Press, 2001.

Justeson, J. & Katz, S. Technical Terminology: some linguistic properties and an algorithm for identification in text. *Natural Language Engineering* 1(1):9-27, 1995.

Liu, B., Hu, M. and Cheng, J. 2005. Opinion Observer: Analyzing and comparing opinions on the Web. *WWW-2005*.

Morinaga, S., Yamanishi, K., Tateishi, K, and Fukushima, T. 2002. Mining Product Reputations on the Web. *KDD'02*, 2002.

Nasukawa, T. & Yi, J. 2003. Sentiment analysis: Capturing favorability using natural language processing. *Proceedings of the 2nd Intl Conf. on Knowledge Capture (K-CAP 2003)*.

Nigam, K. and Hurst, M. 2004. Towards a robust metric of opinion. *AAAI Spring Symp.on Exploring Attitude and Affect in Text*.

NLProcessor, 2000. http://www.infogistics.com/textanalysis.html

Pang, B., Lee, L., and Vaithyanathan, S. Thumbs up? Sentiment classification using machine learning techniques. *EMNLP-02*.

Popescu, A-M and Etzioni. O. 2005. “"Extracting Product Features and Opinions from Reviews. *EMNLP-05*.

Pei, J. Han, J., Mortazavi-Asl, B., Wang, J., Pinto, H., Chen, Q., Dayal, U., and Hsu, M.-C. Mining Sequential Patterns by Pattern-Growth: The PrefixSpan Approach. IEEE Transactions on Knowledge and Data Engineering, 16(10), 2004.

Riloff, E and Wiebe, J. 2003. Learning extraction patterns for subjective expressions. *EMNLP-03*. .

Tong, R. 2001. An Operational System for Detecting and Tracking Opinions in on-line discussion. *SIGIR 2001 Workshop on Operational Text Classification*, 2001.

Turney, P. Thumbs Up or Thumbs Down? semantic orientation applied to unsupervised classification of reviews. *ACL'2002*.

Wiebe, J., Bruce, R., O'Hara, T. Development and use of a gold standard data set for subjectivity classifications. *ACL'99*, 1999.

Wilson, T, Wiebe, J, & Hwa, R. Just how mad are you? Finding strong and weak opinion clauses. *AAAI-04,* 2004.

Yu, H and Hatzivassiloglou, V. Towards answering opinion questions: Separating facts from opinions and identifying the polarity of opinion sentences. *EMNLP-03*, 2003.