Extracting Product Features and Opinions from Reviews

Ana-Maria Popescu and Oren Etzioni
Department of Computer Science and Engineering
University of Washington
Seattle, WA 98195-2350
{amp, etzioni}@cs.washington.edu

Abstract

Consumers are often forced to wade through many on-line reviews in order to make an informed product choice. This paper introduces OPINE, an unsupervised information-extraction system which mines reviews in order to build a model of important product features, their evaluation by reviewers, and their relative quality across products.

Compared to previous work, OPINE achieves 22% higher precision (with only 3% lower recall) on the feature extraction task. OPINE's novel use of *relaxation labeling* for finding the semantic orientation of words in context leads to strong performance on the tasks of finding opinion phrases and their polarity.

1 Introduction

The Web contains a wealth of opinions about products, politicians, and more, which are expressed in newsgroup posts, review sites, and elsewhere. As a result, the problem of "opinion mining" has seen increasing attention over the last three years from (Turney, 2002; Hu and Liu, 2004) and many others. This paper focuses on product reviews, though our methods apply to a broader range of opinions.

Product reviews on Web sites such as `amazon.com` and elsewhere often associate meta-data with each review indicating how positive (or negative) it is using a 5-star scale, and also rank products by how they fare in the reviews at the site. However, the reader's taste may differ from the reviewers'. For example, the reader may feel strongly about the quality of the gym in a hotel, whereas many reviewers may focus on other aspects of the hotel, such as the decor or the location. Thus, the reader is forced to wade through a large number of reviews looking for information about particular features of interest.

We decompose the problem of review mining into the following main subtasks:

I. Identify product features.

II. Identify opinions regarding product features.

III. Determine the polarity of opinions.

IV. Rank opinions based on their strength.

This paper introduces OPINE, an unsupervised information extraction system that embodies a solution to each of the above subtasks. OPINE is built on top of the KnowItAll Web information-extraction system (Etzioni et al., 2005) as detailed in Section 3.

Given a particular product and a corresponding set of reviews, OPINE solves the opinion mining tasks outlined above and outputs a set of *product features*, each accompanied by a list of *associated opinions* which are ranked based on strength (*e.g.*, "abominable" is stronger than "bad). This output information can then be used to generate various types of opinion summaries.

This paper focuses on the first 3 review mining subtasks and our contributions are as follows:

1. We introduce OPINE, a review-mining system whose novel components include the use of *relaxation labeling* to find the semantic orientation of words in the context of given product features and sentences.

2. We compare OPINE with the most relevant previous review-mining system (Hu and Liu, 2004) and find that OPINE's precision on the *feature extraction* task is 22% better though its recall is 3% lower on Hu's data sets. We show that 1/3 of this increase in precision comes from using OPINE's *feature assessment* mechanism on review data while the rest is due to Web PMI statistics.

3. While many other systems have used extracted opinion phrases in order to determine the polarity of sentences or documents, OPINE is the first to report its precision and recall on the tasks of *opinion phrase extraction* and *opinion phrase polarity determination* in the context of known product features and sentences. On the first task, OPINE has a precision of 79% and a recall of 76%. On the second task, OPINE has a precision of 86% and a recall of 89%.

Input: product class C, reviews R.
Output: set of [feature, ranked opinion list] tuples
R' ← parseReviews(R);
E ← findExplicitFeatures(R', C);
O ← findOpinions(R', E);
CO ← clusterOpinions(O);
I ← findImplicitFeatures(CO, E);
RO ← rankOpinions(CO);
$\{(f, o_i, ...o_j)...\}$←outputTuples(RO, I ∪ E);

Figure 1: **OPINE Overview.**

The remainder of this paper is organized as follows: Section 2 introduces the basic terminology, Section 3 gives an overview of OPINE, describes and evaluates its main components, Section 4 describes related work and Section 5 presents our conclusion.

2 Terminology

A *product class* (*e.g.*, Scanner) is a set of *products* (*e.g.*, Epson1200). OPINE extracts the following types of *product features*: *properties*, *parts*, *features of product parts*, *related concepts*, *parts* and *properties of related concepts* (see Table 1 for examples of such features in the Scanner domains). *Related concepts* are concepts relevant to the customers' experience with the main product (*e.g.*, the company that manufactures a scanner). The relationships between the main product and related concepts are typically expressed as verbs (*e.g.*, "Epson *manufactures* scanners") or prepositions ("scanners *from* Epson"). Features can be *explicit* ("good `scan quality`") or *implicit* ("good scans" implies good `ScanQuality`).

OPINE also extracts *opinion phrases*, which are adjective, noun, verb or adverb phrases representing customer opinions. Opinions can be *positive* or *negative* and vary in *strength* (*e.g.*, "fantastic" is stronger than "good").

3 OPINE Overview

This section gives an overview of OPINE (see Figure 1) and describes its components and their experimental evaluation.

Goal Given product class C with instances I and reviews R, OPINE's goal is to find a set of (feature, opinions) tuples $\{(f, o_i, ...o_j)\}$ s.t. $f \in F$ and $o_i, ...o_j \in O$, where:

a) F is the set of product class features in R.

b) O is the set of opinion phrases in R.

c) f is a feature of a particular product instance.

d) o is an opinion about f in a particular sentence.

d) the opinions associated with each feature f are ranked based on their strength.

Solution The steps of our solution are outlined in Figure 1 above. OPINE parses the reviews using MINIPAR (Lin, 1998) and applies a simple pronoun-resolution module to parsed review data. OPINE then uses the data to find *explicit* product features (E). OPINE's *Feature Assessor* and its use of Web PMI statistics are vital for the extraction of high-quality features (see 3.2). OPINE then identifies *opinion phrases* associated with features in E and finds their polarity. OPINE's novel use of relaxation-labeling techniques for determining the semantic orientation of potential opinion words in the context of given features and sentences leads to high precision and recall on the tasks of *opinion phrase extraction* and *opinion phrase polarity extraction* (see 3.3).

In this paper, we only focus on the extraction of explicit features, identifying corresponding customer opinions about these features and determining their polarity. We omit the descriptions of the opinion clustering, implicit feature generation and opinion ranking algorithms.

3.0.1 The KnowItAll System.

OPINE is built on top of KnowItAll, a Web-based, domain-independent information extraction system (Etzioni et al., 2005). Given a set of relations of interest, KnowItAll instantiates relation-specific generic extraction patterns into extraction rules which find candidate facts. KnowItAll's Assessor then assigns a probability to each candidate. The Assessor uses a form of *Point-wise Mutual Information* (PMI) between phrases that is estimated from Web search engine hit counts (Turney, 2001). It computes the PMI between each fact and *automatically generated discriminator phrases* (*e.g.*, "is a scanner" for the `isA()` relationship in the context of the `Scanner` class). Given fact f and discriminator d, the computed PMI score is:

$$\text{PMI}(f, d) = \frac{\text{Hits}(d + f)}{\text{Hits}(d) * \text{Hits}(f)}$$

The PMI scores are converted to binary features for a Naive Bayes Classifier, which outputs a probability associated with each fact (Etzioni et al., 2005).

3.1 Finding Explicit Features

OPINE extracts *explicit* features for the given product class from parsed review data. First, the system recursively identifies both the *parts* and the *properties* of the given product class and their parts and properties, in turn, continuing until no candidates are found. Then, the system finds *related concepts* as described in (Popescu et al., 2004) and extracts their parts and properties. Table 1 shows that each feature type contributes to the set of final features (averaged over 7 product classes).

Explicit Features	Examples	% Total
Properties	ScannerSize	7%
Parts	ScannerCover	52%
Features of Parts	BatteryLife	24%
Related Concepts	ScannerImage	9%
Related Concepts' Features	ScannerImageSize	8%

Table 1: **Explicit Feature Information**

In order to find parts and properties, OPINE first extracts the noun phrases from reviews and retains those with frequency greater than an experimentally set threshold. OPINE's *Feature Assessor*, which is an instantiation of KnowItAll's Assessor, evaluates each noun phrase by computing the PMI scores between the phrase and *meronymy discriminators* associated with the product class (*e.g.*, "of scanner", "scanner has", "scanner comes with", etc. for the `Scanner` class). OPINE distinguishes parts from properties using WordNet's IS-A hierarchy (which enumerates different kinds of properties) and morphological cues (*e.g.*, "-iness", "-ity" suffixes).

3.2 Experiments: Explicit Feature Extraction

In our experiments we use sets of reviews for 7 product classes (1621 total reviews) which include the publicly available data sets for 5 product classes from (Hu and Liu, 2004). Hu's system is the review mining system most relevant to our work. It uses association rule mining to extract *frequent* review noun phrases as features. Frequent features are used to find *potential opinion* words (only adjectives) and the system uses WordNet synonyms/antonyms in conjunction with a set of seed words in order to find actual *opinion* words. Finally, opinion words are used to extract associated *infrequent* features. The system only extracts *explicit* features.

On the 5 datasets in (Hu and Liu, 2004), OPINE's precision is 22% higher than Hu's at the cost of a 3% recall drop. There are two important differences between OPINE and Hu's system: a) OPINE's Feature Assessor uses PMI assessment to evaluate each candidate feature and b) OPINE incorporates Web PMI statistics in addition to review data in its assessment. In the following, we quantify the performance gains from a) and b).

a) In order to quantify the benefits of OPINE's Feature Assessor, we use it to evaluate the features extracted by Hu's algorithm on review data (**Hu+A/R**). The Feature Assessor improves Hu's precision by 6%.

b) In order to evaluate the impact of using Web PMI statistics, we assess OPINE's features first on reviews (**OP/R**) and then on reviews in conjunction with the Web (the corresponding methods are **Hu+A/R+W** and **OPINE**). Web PMI statistics increase precision by an average of 14.5%.

Overall, 1/3 of **OPINE**'s precision increase over Hu's system comes from using PMI assessment on reviews and the other 2/3 from the use of the Web PMI statistics.

In order to show that OPINE's performance is robust across multiple product classes, we used two sets of reviews downloaded from `tripadvisor.com` for Hotels and `amazon.com` for Scanners. Two annotators labeled a set of unique 450 OPINE extractions as *correct* or *incorrect*. The inter-annotator agreement was 86%. The extractions on which the annotators agreed were used to compute OPINE's precision, which was 89%. Furthermore, the annotators extracted explicit features from 800 review sentences (400 for each domain). The inter-annotator agreement was 82%. OPINE's recall on the set of 179 features on which both annotators agreed was 73%.

Data	Explicit Feature Extraction: Precision				
	Hu	**Hu+A/R**	**Hu+A/R+W**	**OP/R**	**OPINE**
D_1	*0.75*	+0.05	+0.17	+0.07	**+0.19**
D_2	*0.71*	+0.03	+0.19	+0.08	**+0.22**
D_3	*0.72*	+0.03	+0.25	+0.09	**+0.23**
D_4	*0.69*	+0.06	+0.22	+0.08	**+0.25**
D_5	*0.74*	+0.08	+0.19	+0.04	**+0.21**
Avg	*0.72*	+0.06	+ 0.20	+0.07	**+0.22**

Table 2: **Precision Comparison on the Explicit Feature-Extraction Task.** OPINE's precision is 22% better than Hu's precision; Web PMI statistics are responsible for 2/3 of the precision increase. All results are reported with respect to Hu's.

Data	Explicit Feature Extraction: Recall				
	Hu	**Hu+A/R**	**Hu+A/R+W**	**OP/R**	**OPINE**
D_1	*0.82*	-0.16	-0.08	-0.14	**-0.02**
D_2	*0.79*	-0.17	-0.09	-0.13	**-0.06**
D_3	*0.76*	-0.12	-0.08	-0.15	**-0.03**
D_4	*0.82*	-0.19	-0.04	-0.17	**-0.03**
D_5	*0.80*	-0.16	-0.06	-0.12	**-0.02**
Avg	*0.80*	-0.16	-0.07	-0.14	**-0.03**

Table 3: **Recall Comparison on the Explicit Feature-Extraction Task.** OPINE's recall is 3% lower than the recall of Hu's original system (precision level = 0.8). All results are reported with respect to Hu's.

3.3 Finding Opinion Phrases and Their Polarity

This subsection describes how OPINE extracts potential opinion phrases, distinguishes between opinions and non-opinions, and finds the *polarity* of each opinion in the context of its associated feature in a particular review sentence.

3.3.1 Extracting Potential Opinion Phrases

OPINE uses explicit features to identify potential opinion phrases. Our intuition is that an opinion phrase associated with a product feature will occur in its vicinity. This idea is similar to that of (Kim and Hovy, 2004) and (Hu and Liu, 2004), but instead of using a window of size k or the output of a noun phrase chunker, OPINE takes advantage of the syntactic dependencies computed by the MINIPAR parser. Our intuition is embodied by 10 *extraction rules*, some of which are shown in Table 4. If an explicit feature is found in a sentence, OPINE applies the extraction rules in order to find the heads of potential opinion phrases. Each head word together with its modi-

fiers is returned as a potential opinion phrase[1].

Extraction Rules	Examples
if $\exists(M, NP = f) \rightarrow po = M$	(expensive) `scanner`
if $\exists(S = f, P, O) \rightarrow po = O$	`lamp` has (problems)
if $\exists(S, P, O = f) \rightarrow po = P$	I (hate) this `scanner`
if $\exists(S = f, P, O) \rightarrow po = P$	`program` (crashed)

Table 4: **Examples of Domain-independent Rules for the Extraction of Potential Opinion Phrases.** Notation: po=potential opinion, M=modifier, NP=noun phrase, S=subject, P=predicate, O=object. Extracted phrases are enclosed in parentheses. Features are indicated by the typewriter font. The equality conditions on the left-hand side use po's head.

Rule Templates	Rules
$dep(w, w')$	$m(w, w')$
$\exists v$ s.t. $dep(w, v), dep(v, w')$	$\exists v$ s.t. $m(w, v), o(v, w')$
$\exists v$ s.t. $dep(w, v), dep(w', v)$	$\exists v$ s.t. $m(w, v), o(w', v)$

Table 5: **Dependency Rule Templates For Finding Words w, w' with Related SO Labels** . OPINE instantiates these templates in order to obtain extraction rules. Notation: dep=dependent, m=modifier, o=object, v,w,w'=words.

OPINE examines the potential opinion phrases in order to identify the actual opinions. First, the system finds the semantic orientation for the lexical head of each potential opinion phrase. Every phrase whose head word has a *positive* or *negative* semantic orientation is then retained as an *opinion phrase*. In the following, we describe how OPINE finds the semantic orientation of words.

3.3.2 Word Semantic Orientation

OPINE finds the semantic orientation of a word w in the context of an associated feature f and sentence s. We restate this task as follows:

Task Given a set of *semantic orientation (SO) labels* ($\{positive, negative, neutral\}$), a set of reviews and a set of tuples (w, f, s), where w is a potential opinion word associated with feature f in sentence s, assign a SO label to each tuple (w, f, s).

For example, the tuple (*sluggish*, *driver*, "I am not happy with this sluggish driver") would be assigned a *negative* SO label.

Note: We use "word" to refer to a potential opinion word w and "feature" to refer to the word or phrase which represents the explicit feature f.

Solution OPINE uses the 3-step approach below:

1. Given the set of reviews, OPINE finds a SO label for each word w.

2. Given the set of reviews and the set of SO labels for words w, OPINE finds a SO label for each (w, f) pair.

[1]The (S,P,O) tuples in Table 4 are automatically generated from MINIPAR's output.

3. Given the set of SO labels for (w, f) pairs, OPINE finds a SO label for each (w, f, s) input tuple.

Each of these subtasks is cast as an *unsupervised collective classification* problem and solved using the same mechanism. In each case, OPINE is given a set of *objects* (words, pairs or tuples) and a set of *labels* (SO labels); OPINE then searches for a *global* assignment of labels to objects. In each case, OPINE makes use of *local constraints* on label assignments (*e.g.*, conjunctions and disjunctions constraining the assignment of SO labels to words (Hatzivassiloglou and McKeown, 1997)).

A key insight in OPINE is that the problem of searching for a *global* SO label assignment to words, pairs or tuples while trying to satisfy as many *local* constraints on assignments as possible is analogous to labeling problems in computer vision (*e.g.*, model-based matching). OPINE uses a well-known computer vision technique, *relaxation labeling* (Hummel and Zucker, 1983), in order to solve the three subtasks described above.

3.3.3 Relaxation Labeling Overview

Relaxation labeling is an unsupervised classification technique which takes as input:
a) a set of *objects* (*e.g.*, words)
b) a set of *labels* (*e.g.*, SO labels)
c) initial probabilities for each object's possible labels
d) the definition of an object o's *neighborhood* (a set of other objects which influence the choice of o's label)
e) the definition of *neighborhood features*
f) the definition of a *support function* for an object label

The influence of an object o's neighborhood on its label L is quantified using the *support function*. The support function computes the probability of the label L being assigned to o as a function of o's *neighborhood features*. Examples of features include the fact that a certain *local constraint* is satisfied (*e.g.*, the word $nice$ participates in the conjunction *and* together with some other word whose SO label is estimated to be *positive*).

Relaxation labeling is an iterative procedure whose output is an assignment of labels to objects. At each iteration, the algorithm uses an *update equation* to reestimate the probability of an object label based on its previous probability estimate and the features of its neighborhood. The algorithm stops when the global label assignment stays constant over multiple consecutive iterations.

We employ relaxation labeling for the following reasons: a) it has been extensively used in computer-vision with good results b) its formalism allows for many types of constraints on label assignments to be used simultaneously. As mentioned before, constraints are integrated into the algorithm as neighborhood features which influence the assignment of a particular label to a particular object.

OPINE uses the following sources of constraints:

a) *conjunctions* and *disjunctions* in the review text

b) manually-supplied *syntactic dependency rule templates* (see Table 5). The templates are automatically instantiated by our system with different dependency relationships (premodifier, postmodifier, subject, etc.) in order to obtain syntactic dependency rules which find words with related SO labels.

c) automatically derived *morphological relationships* (*e.g.*, "wonderful" and "wonderfully" are likely to have similar SO labels).

d) WordNet-supplied *synonymy, antonymy, IS-A* and *morphological* relationships between words. For example, *clean* and *neat* are synonyms and so they are likely to have similar SO labels.

Each of the SO label assignment subtasks previously identified is solved using a relaxation labeling step. In the following, we describe in detail how relaxation labeling is used to find SO labels for words in the given review sets.

3.3.4 Finding SO Labels for Words

For many words, a word sense or set of senses is used throughout the review corpus with a consistently positive, negative or neutral connotation (*e.g.*, "great", "awful", etc.). Thus, in many cases, a word w's SO label in the context of a feature f and sentence s will be the same as its SO label in the context of other features and sentences. In the following, we describe how OPINE's relaxation labeling mechanism is used to find a word's dominant SO label in a set of reviews.

For this task, a word's *neighborhood* is defined as the set of words connected to it through conjunctions, disjunctions and all other relationships previously introduced as sources of constraints.

RL uses an *update equation* to re-estimate the probability of a word label based on its previous probability estimate and the features of its neighborhood (see **Neighborhood Features**). At iteration m, let $q(w, L)_{(m)}$ denote the support function for label L of w and let $P(l(w) = L)_{(m)}$ denote the probability that L is the label of w. $P(l(w) = L)_{(m+1)}$ is computed as follows:

RL Update Equation (Rangarajan, 2000)

$$P(l(w) = L)_{(m+1)} = \frac{P(l(w) = L)_{(m)}(1 + \alpha q(w, L)_{(m)})}{\sum_{L'} P(l(w) = L')_{(m)}(1 + \alpha q(w, L')_{(m)})}$$

where $L' \in \{pos, neg, neutral\}$ and $\alpha > 0$ is an experimentally set constant keeping the numerator and probabilities positive. RL's output is an assignment of dominant SO labels to words.

In the following, we describe in detail the initialization step, the derivation of the support function formula and the use of neighborhood features.

RL Initialization Step OPINE uses a version of Turney's PMI-based approach (Turney, 2003) in order to derive the initial probability estimates ($P(l(w) = L)_{(0)}$) for a subset S of the words. OPINE computes a *SO score so(w)* for each w in S as the difference between the PMI of w with positive keywords (*e.g.*, "excellent") and the PMI of w with negative keywords (*e.g.*, "awful"). When $so(w)$ is small, or w rarely co-occurs with the keywords, w is classified as *neutral*. If $so(w) > 0$, then w is *positive*, otherwise w is negative. OPINE then uses the labeled S set in order to compute prior probabilities $P(l(w) = L)$, $L \in \{pos, neg, neutral\}$ by computing the ratio between the number of words in S labeled L and $|S|$. Such probabilities are used as initial probability estimates associated with the labels of the remaining words.

Support Function The support function computes the probability of each label for word w based on the labels of objects in w's neighborhood N.

Let $A_k = \{(w_j, L_j)|w_j \in N\}$, $0 < k \leq 3^{|N|}$ represent one of the potential assignments of labels to the words in N. Let $P(A_k)_{(m)}$ denote the probability of this particular assignment at iteration m. The *support* for label L of word w at iteration m is :

$$q(w, L)_{(m)} = \sum_{k=1}^{3^{|N|}} P(l(w) = L|A_k)_{(m)} * P(A_k)_{(m)}$$

We assume that the labels of w's neighbors are independent of each other and so the formula becomes:

$$q(w, L)_{(m)} = \sum_{k=1}^{3^{|N|}} P(l(w) = L|A_k)_{(m)} * \prod_{j=1}^{|N|} P(l(w_j) = L_j)_{(m)}$$

Every $P(l(w_j) = L_j)_{(m)}$ term is the estimate for the probability that $l(w_j) = L_j$ (which was computed at iteration m using the RL update equation).

The $P(l(w) = L|A_k)_{(m)}$ term quantifies the influence of a particular label assignment to w's neighborhood over w's label. In the following, we describe how we estimate this term.

Neighborhood Features

Each type of word relationship which constrains the assignment of SO labels to words (synonymy, antonymy, etc.) is mapped by OPINE to a neighborhood feature. This mapping allows OPINE to use simultaneously use multiple independent sources of constraints on the label of a particular word. In the following, we formalize this mapping.

Let T denote the type of a word relationship in R (synonym, antonym, etc.) and let $A_{k,T}$ represent the labels assigned by A_k to neighbors of a word w which are connected to w through a relationship of type T . We have $A_k = \bigcup_T A_{k,T}$ and

$$P(l(w) = L|A_k)_{(m)} = P(l(w) = L|\bigcup_T A_{k,T})_{(m)}$$

For each relationship type T, OPINE defines a *neighborhood feature* $f_T(w, L, A_{k,T})$ which computes $P(l(w) = L|A_{k,T})$, the probability that w's label is L given $A_{k,T}$ (see below). $P(l(w) = L|\bigcup_T A_{k,T})_{(m)}$ is estimated combining the information from various features about w's label using the sigmoid function $\sigma()$:

$$P(l(w) = L|A_k)_{(m)} = \sigma(\sum_{i=1}^{j} f_i(w, L, A_{k,i})_{(m)} * c_i)$$

where $c_0, ...c_j$ are weights whose sum is 1 and which reflect OPINE 's confidence in each type of feature.

Given word w, label L, relationship type T and neighborhood label assignment A_k, let N_T represent the subset of w's neighbors connected to w through a type T relationship. The feature f_T computes the probability that w's label is L given the labels assigned by A_k to words in N_T. Using Bayes's Law and assuming that these labels are independent given $l(w)$, we have the following formula for f_T at iteration m:

$$f_T(w, L, A_{k,T})_{(m)} = P(l(w) = L)_{(m)} * \prod_{j=1}^{|N_T|} P(L_j|l(w) = L)$$

$P(L_j|l(w) = L)$ is the probability that word w_j has label L_j if w_j and w are linked by a relationship of type T and w has label L. We make the simplifying assumption that this probability is constant and depends only of T, L and L', not of the particular words w_j and w. For each tuple (T, L, L_j), $L, L_j \in \{pos, neg, neutral\}$, OPINE builds a probability table using a small set of bootstrapped positive, negative and neutral words.

3.3.5 Finding (Word, Feature) SO Labels

This subtask is motivated by the existence of frequent words which change their SO label based on associated features, but whose SO labels in the context of the respective features are consistent throughout the reviews (*e.g.*, in the Hotel domain, "hot water" has a consistently positive connotation, whereas "hot room" has a negative one).

In order to solve this task, OPINE first assigns each (w, f) pair an initial SO label which is w's SO label. The system then executes a relaxation labeling step during which syntactic relationships between words and, respectively, between features, are used to update the default SO labels whenever necessary. For example, *(hot, room)* appears in the proximity of *(broken, fan)*. If "room"and "fan" are conjoined by *and*, this suggests that "hot" and "broken" have similar SO labels in the context of their respective features. If "broken" has a strongly negative semantic orientation, this fact contributes to OPINE's belief that "hot" may also be negative in this context. Since *(hot, room)* occurs in the vicinity of other such phrases (*e.g., stifling kitchen*), "hot" acquires a negative SO label in the context of "room".

3.3.6 Finding (Word, Feature, Sentence) SO Labels

This subtask is motivated by the existence of (w,f) pairs (*e.g., (big, room)*) for which w's orientation changes based on the sentence in which the pair appears (*e.g.*, " I hated the big, drafty room because I ended up freezing." vs. "We had a big, luxurious room".)

In order to solve this subtask, OPINE first assigns each (w, f, s) tuple an initial label which is simply the SO label for the (w, f) pair. The system then uses syntactic relationships between words and, respectively, features in order to update the SO labels when necessary. For example, in the sentence "I hated the big, drafty room because I ended up freezing.", "big" and "hate" satisfy condition 2 in Table 5 and therefore OPINE expects them to have similar SO labels. Since "hate" has a strong negative connotation, "big" acquires a negative SO label in this context.

In order to correctly update SO labels in this last step, OPINE takes into consideration the presence of *negation modifiers*. For example, in the sentence "I don't like a large scanner either", OPINE first replaces the *positive* (w, f) pair *(like, scanner)* with the *negative* labeled pair *(not like, scanner)* and then infers that "large" is likely to have a negative SO label in this context.

3.3.7 Identifying Opinion Phrases

After OPINE has computed the most likely SO labels for the head words of each potential opinion phrase in the context of given features and sentences, OPINE can extract opinion phrases and establish their polarity. Phrases whose head words have been assigned *positive* or *negative* labels are retained as *opinion phrases*. Furthermore, the polarity of an opinion phrase o in the context of a feature f and sentence s is given by the SO label assigned to the tuple $(head(o), f, s)$ (3.3.6 shows how OPINE takes into account negation modifiers).

3.4 Experiments

In this section we evaluate OPINE's performance on the following tasks: finding SO labels of words in the context of known features and sentences (*SO label extraction*); distinguishing between opinion and non-opinion phrases in the context of known features and sentences (*opinion phrase extraction*); finding the correct polarity of extracted opinion phrases in the context of known features and sentences (*opinion phrase polarity extraction*).

While other systems, such as (Hu and Liu, 2004; Turney, 2002), have addressed these tasks to some degree, OPINE is the first to report results. We first ran OPINE on 13841 sentences and 538 previously extracted features. OPINE searched for a SO label assignment for 1756 different words in the context of the given features and sentences. We compared OPINE against two baseline methods, **PMI++** and **Hu++**.

PMI++ is an extended version of (Turney, 2002)'s method for finding the SO label of a phrase (as an attempt to deal with context-sensitive words). For a given (word, feature, sentence) tuple, **PMI++** ignores the sentence, generates a phrase based on the word and the feature (*e.g., (clean, room)*: "clean room") and finds its SO label using PMI statistics. If unsure of the label, **PMI++** tries to find the orientation of the potential opinion word instead. The search engine queries use domain-specific keywords (*e.g.*, "scanner"), which are dropped if they

lead to low counts.

Hu++ is a WordNet-based method for finding a word's context-independent semantic orientation. It extends Hu's adjective labeling method in a number of ways in order to handle nouns, verbs and adverbs in addition to adjectives and in order to improve coverage. Hu's method starts with two sets of positive and negative words and iteratively grows each one by including synonyms and antonyms from WordNet. The final sets are used to predict the orientation of an incoming word.

Type	PMI++		Hu++		OPINE	
	P	R	P	R	P	R
adj	0.73	0.91	+0.02	-0.17	**+0.07**	**-0.03**
nn	0.63	0.92	+0.04	-0.24	**+0.11**	**-0.08**
vb	0.71	0.88	+0.03	-0.12	**+0.01**	**-0.01**
adv	0.82	0.92	+0.02	-0.01	**+0.06**	**+0.01**
Avg	0.72	0.91	+0.03	-0.14	**+0.06**	**-0.03**

Table 6: **Finding SO Labels of Potential Opinion Words in the Context of Given Product Features and Sentences.** OPINE's precision is higher than that of **PMI++** and **Hu++**. All results are reported with respect to **PMI++** . Notation: adj=adjectives, nn=nouns, vb=verbs, adv=adverbs

3.4.1 Experiments: SO Labels

On the task of *finding SO labels for words in the context of given features and review sentences*, OPINE obtains higher precision than both baseline methods at a small loss in recall with respect to **PMI++**. As described below, this result is due in large part to OPINE's ability to handle context-sensitive opinion words.

We randomly selected 200 (word, feature, sentence) tuples for each word type (adjective, adverb, etc.) and obtained a test set containing 800 tuples. Two annotators assigned positive, negative and neutral labels to each tuple (the inter-annotator agreement was 78%). We retained the tuples on which the annotators agreed as the gold standard. We ran **PMI++** and **Hu++** on the test data and compared the results against OPINE's results on the same data.

In order to quantify the benefits of each of the three steps of our method for finding SO labels, we also compared OPINE with a version which only finds SO labels for words and a version which finds SO labels for words in the context of given features, but doesn't take into account given sentences. We have learned from this comparison that OPINE's precision gain over **PMI++** and **Hu++** is mostly due to to its ability to handle context-sensitive words in a large number of cases.

Although **Hu++** does not handle context-sensitive SO label assignment, its average precision was reasonable (75%) and better than that of **PMI++**. Finding a word's SO label is good enough in the case of strongly positive or negative opinion words, which account for the majority of opinion instances. The method's loss in recall is due to not recognizing words absent from WordNet (*e.g.*, "depth-adjustable") or not having enough information to classify some words in WordNet.

PMI++ typically does well in the presence of strongly positive or strongly negative words. Its high recall is correlated with decreased precision, but overall this simple approach does well. **PMI++**'s main shortcoming is misclassifying terms such as "basic" or "visible" which change orientation based on context.

3.4.2 Experiments: Opinion Phrases

In order to evaluate OPINE on the tasks of *opinion phrase extraction* and *opinion phrase polarity extraction in the context of known features and sentences*, we used a set of 550 sentences containing previously extracted features. The sentences were annotated with the opinion phrases corresponding to the known features and with the opinion polarity. We compared OPINE with **PMI++** and **Hu++** on the tasks of interest. We found that OPINE had the highest precision on both tasks at a small loss in recall with respect to **PMI++**. OPINE's ability to identify a word's SO label in the context of a given feature and sentence allows the system to correctly extract opinions expressed by words such as "big" or "small", whose semantic orientation varies based on context.

Measure	PMI++	Hu++	OPINE
OP Extraction: Precision	0.71	+0.06	**+0.08**
OP Extraction: Recall	0.78	-0.08	**-0.02**
OP Polarity: Precision	0.80	-0.04	**+0.06**
OP Polarity: Recall	0.93	+0.07	**-0.04**

Table 7: **Extracting Opinion Phrases and Opinion Phrase Polarity Corresponding to Known Features and Sentences.** OPINE's precision is higher than that of **PMI++** and of **Hu++**. All results are reported with respect to **PMI++**.

4 Related Work

The key components of OPINE described in this paper are the PMI feature assessment which leads to high-precision feature extraction and the use of relaxation-labeling in order to find the semantic orientation of potential opinion words. The review-mining work most relevant to our research is that of (Hu and Liu, 2004) and (Kobayashi et al., 2004). Both identify product features from reviews, but OPINE significantly improves on both. (Hu and Liu, 2004) doesn't assess candidate features, so its precision is lower than OPINE's. (Kobayashi et al., 2004) employs an iterative semi-automatic approach which requires human input at every iteration. Neither model explicitly addresses *composite* (feature of feature) or *implicit* features. Other systems (Morinaga et al., 2002; Kushal et al., 2003) also look at Web product reviews but they do not extract

opinions about particular product features. OPINE's use of meronymy lexico-syntactic patterns is similar to that of many others, from (Berland and Charniak, 1999) to (Almuhareb and Poesio, 2004).

Recognizing the subjective character and polarity of words, phrases or sentences has been addressed by many authors, including (Turney, 2003; Riloff et al., 2003; Wiebe, 2000; Hatzivassiloglou and McKeown, 1997). Most recently, (Takamura et al., 2005) reports on the use of spin models to infer the semantic orientation of words. The paper's global optimization approach and use of multiple sources of constraints on a word's semantic orientation is similar to ours, but the mechanism differs and they currently omit the use of syntactic information. Subjective phrases are used by (Turney, 2002; Pang and Vaithyanathan, 2002; Kushal et al., 2003; Kim and Hovy, 2004) and others in order to classify reviews or sentences as positive or negative. So far, OPINE's focus has been on extracting and analyzing opinion phrases corresponding to specific features in specific sentences, rather than on determining sentence or review polarity.

5 Conclusion

OPINE is an unsupervised information extraction system which extracts fine-grained features, and associated opinions, from reviews. OPINE's use of the Web as a corpus helps identify product features with improved precision compared with previous work. OPINE uses a novel relaxation-labeling technique to determine the semantic orientation of potential opinion words in the context of the extracted product features and specific review sentences; this technique allows the system to identify customer opinions and their polarity with high precision and recall.

6 Acknowledgments

We would like to thank the KnowItAll project and the anonymous reviewers for their comments. Michael Gamon, Costas Boulis and Adam Carlson have also provided valuable feedback. We thank Minquing Hu and Bing Liu for providing their data sets and for their comments. Finally, we are grateful to Bernadette Minton and Fetch Technologies for their help in collecting additional reviews. This research was supported in part by NSF grant IIS-0312988, DARPA contract NBCHD030010, ONR grant N00014-02-1-0324 as well as gifts from Google and the Turing Center.

References

A. Almuhareb and M. Poesio. 2004. Attribute-based and value-based clustering: An evaluation. In *EMNLP*, pages 158–165.

M. Berland and E. Charniak. 1999. Finding parts in very large corpora. In *ACL*, pages 57–64.

O. Etzioni, M. Cafarella, D. Downey, S. Kok, A. Popescu, T. Shaked, S. Soderland, D. Weld, and A. Yates. 2005. Unsupervised named-entity extraction from the web: An experimental study. *Artificial Intelligence*, 165(1):91–134.

V. Hatzivassiloglou and K. McKeown. 1997. Predicting the semantic orientation of adjectives. In *ACL/EACL*, pages 174–181.

M. Hu and B. Liu. 2004. Mining and Summarizing Customer Reviews. In *KDD*, pages 168–177, Seattle, WA.

R.A. Hummel and S.W. Zucker. 1983. On the foundations of relaxation labeling processes. In *PAMI*, pages 267–287.

S. Kim and E. Hovy. 2004. Determining the sentiment of opinions. In *COLING*.

N. Kobayashi, K. Inui, K. Tateishi, and T. Fukushima. 2004. Collecting Evaluative Expressions for Opinion Extraction. In *IJCNLP*, pages 596–605.

D. Kushal, S. Lawrence, and D. Pennock. 2003. Mining the peanut gallery: Opinion extraction and semantic classification of product reviews. In *WWW*.

D. Lin. 1998. Dependency-based evaluation of MINIPAR. In *Workshop on Evaluation of Parsing Systems at ICLRE*.

S. Morinaga, K. Yamanishi, K. Tateishi, and T. Fukushima. 2002. Mining product reputations on the web. In *KDD*.

Lee L. Pang, B and S. Vaithyanathan. 2002. Thumbs up? sentiment classification using machine learning techniques. In *EMNLP*, pages 79–86.

A. Popescu, A. Yates, and O. Etzioni. 2004. Class extraction from the World Wide Web. In *AAAI-04 Workshop on Adaptive Text Extraction and Mining*, pages 68–73.

A. Rangarajan. 2000. Self annealing and self annihilation: unifying deterministic annealing and relaxation labeling. In *Pattern Recognition, 33:635-649*.

E. Riloff, J. Wiebe, and T. Wilson. 2003. Learning Subjective Nouns Using Extraction Pattern Bootstrapping. In *CoNLL*, pages 25–32s.

H. Takamura, T. Inui, and M. Okumura. 2005. Extracting Semantic Orientations of Words using Spin Model. In *ACL*, pages 133–141.

P. D. Turney. 2001. Mining the Web for Synonyms: PMI-IR versus LSA on TOEFL. In *Procs. of the Twelfth European Conference on Machine Learning (ECML-2001)*, pages 491–502, Freiburg, Germany.

P. D. Turney. 2002. Thumbs up or thumbs down? semantic orientation applied to unsupervised classification of reviews. In *Procs. of the 40th Annual Meeting of the Association for Computational Linguistics (ACL'02)*, pages 417–424.

P. Turney. 2003. Inference of Semantic Orientation from Association. In *CoRR cs. CL/0309034*.

J. Wiebe. 2000. Learning subjective adjectives from corpora. In *AAAI/IAAI*, pages 735–740.