Comparison of Model-Based Learning Methods for Feature-Level Opinion Mining

Luole Qi
Department of Computer Science
Hong Kong Baptist University
Hong Kong, China
llqi@comp.hkbu.edu.hk

Li Chen
Department of Computer Science
Hong Kong Baptist University
Hong Kong, China
lichen@comp.hkbu.edu.hk

***Abstract*—The tasks of feature-level opinion mining usually include the extraction of product entities from product reviews, the identification of opinion words that are associated with the entities, and the determining of these opinions' polarities (e.g., positive, negative, or neutral). In recent years, several approaches have been proposed such as rule-based and statistical methods on this subject, but few attentions have been paid to applying more discriminative learning models to achieve the goal. On the other hand, little work has evaluated their algorithms' performance for identifying intensifiers, entity phrases and infrequent entities. In this paper, we in particular adopt the Conditional Random Fields (CRFs) model to perform the opinion mining tasks. Relative to related approaches, we have not only highlighted the algorithm's ability in mining intensifiers, phrases and infrequent entities, but also integrated more elements in the model so as to optimize its training and decoding process. Our method was compared to the lexicalized Hidden Markov Model (L-HMMs) based opinion mining method in the experiment, which proves its significantly better accuracy from several aspects.**

***Keywords*-User Reviews, Feature-Level Opinion Mining, Conditional Random Fields (CRFs), Lexicalized Hidden Markov Model (L-HMMs), e-Commerce**

I. Introduction

User-generated reviews have been increasingly regarded useful in business, education, and e-commerce, since they contain valuable opinions originated from the user's experiences. For instance, in e-commerce sites, customers can assess a product's quality by reading other customers' reviews to the product, which will help them to decide whether to purchase the product or not. Nowadays, many e-commerce websites, such as Amazon.com, Yahoo shopping, Epinions.com, allow users to post their reviews freely. The reviews' number has in fact reached to more than thousands in these large websites, which hence poses a challenge for a customer to go through all of them.

To resolve the problem, researchers have done some work on Web opinion mining with the aim to discover the essential information from reviews and then present to users. Most approaches can be classified into two major branches: *document-level* which focuses on producing an overall opinion for one document; and *feature-level* which aims to discover feature entities from sentences and identify opinion words as associated with each entity (e.g., a customer's opinion on the camera's image quality, weight). For the latter branch (which is the focus of our work), previous works have mainly adopted rule-based techniques [14] and statistic plus association rule mining methods such as the one by Hu and Liu [5]. Till recently, some researchers have attempted to adopt more precise machine learning models in order to increase the opinion mining efficacy. One typical work is OpinionMiner [6]. It was built based on lexicalized Hidden Markov Model (L-HMMs) to integrate multiple important linguistic properties into an automatic learning process. Compared to non-model based algorithms, the model-based opinion mining method requires a training process for determining the model's parameters. After the parameters were learnt, it is fast to decode the new data. In fact, [6] demonstrated that their method is more effective and accurate in mining entities and opinions, against related ones that do not have the training phase.

However, their algorithm is still limited at several aspects. First of all, L-HMMs model can not represent distributed hidden states and can not model arbitrary, non-independent entities of the input words' sequences. It can neither involve rich, overlapping model functions. Secondly, the algorithm was not tested in terms of whether it can be competent in identifying intensifiers, phrases, and infrequent entities. Concretely, ***intensifiers*** refer to the words that reviewers use to strengthen or weaken an opinion (e.g., very, highly, absolutely, extremely, greatly). ***Phrase*** is commonly a set of two or more continuous words (e.g., "image quality", "ease of use", "easy to navigate"), to represent an entity or an opinion. ***Infrequent entities*** refer to ones that were commented by few reviewers (so they appear infrequently in the dataset), which however can be used to differentiate products and answer users' specific questions [6]. As a matter of fact, related methods have mainly emphasized extracting pure opinions, single words, and/or frequent features, while ignoring the above elements which though are also important to be discovered from product reviews.

Therefore, to address limitations of related opinion mining techniques, in this paper, we have particularly studied the impact of Conditional Random Fields (CRFs) learning model on accomplishing the goals. Relative to L-HMMs, CRFs model is a discriminative, undirected graph model

[7], which naturally considers arbitrary, non-independent entities without conditional independence assumption, and can involve more rich and overlapping functions. Given these properties, we believe that it could be more qualified and flexible to extract various types of product entities and hence compensate for the inherent weakness of L-HMMs. It is also worth noting that although some attempts have also been lately done to apply CRFs model [9], they primarily used it for document-level segmentation and semantic labeling. Little emphasis has been on exploring its effect on identifying feature-level opinions, intensifiers, phrases and infrequent entities.

To be specific, our contributions to the area of model-based opinion mining can be summarized into the following three points. 1) For preparing the training set to build the model, we have not only specified different data tags, but also integrated the word expansion and self-tagging techniques so as to minimize manual labeling effort. 2) For defining the model, we involved different types of learning functions and identified their optimal combination through the experiment. 3) In the experiment, we mainly compared our CRFs-based opinion mining algorithm to the L-HMMs based method. The comparison was conducted in terms of several units: basic product entities (including components, functions, and features), opinions, intensifiers, phrases, and infrequent entities. The results show that the CRFs approach is more accurate to process and extract them. The differences even reach significant level regarding most comparisons.

The rest of this paper is therefore organized as follows: we first discuss related works in Section II and then describe in detail the CRFs based opinion mining method and algorithm steps in Section III. In Sections IV and V, we present experiment design and results. Finally, we discuss our work and indicate its future directions.

II. Related Work

As mentioned before, the opinion mining has been conducted either at the document level or at the feature level. At the document level (which is to obtain an overall opinion value for the whole document), Turney [14] used point-wise mutual information (PMI) to calculate an average semantic orientation score of extracted phrases for determining the document's polarity. Pang et al. [10] examined the effectiveness of applying machine learning techniques to address the sentiment classification problem for movie review data. Hatzivassiloglou and Wiebe [4] studied the effect of dynamic adjectives, semantically oriented adjectives, and gradable adjectives on a simple subjectivity classifier, and proposed a trainable method that statistically combines two indicators of grad ability. Wilson et al. [15] proposed a system called OpinionFinder that automatically identifies when opinions, sentiments, speculations, and other private states are present in text, via the subjectivity analysis. Das and Chen [3] studied sentimental classification for financial documents. However, although the above works are all related to sentiment classification, they mainly targeted to discover the sentiment to represent a reviewer's overall opinion and did not find which features the reviewer actually liked and disliked. Indeed, an overall negative sentiment on an object does not mean that the reviewer dislikes every aspect of the object.

To in-depth discover a reviewer's opinions on every aspect that s/he mentioned in the text, some researchers have tried to mine and extract opinions at the feature level. Hu and Liu [5] proposed a feature-based opinion summarization system that captures highly frequent feature words through finding association rules under a statistical framework. It extracts the features of a product that customers have expressed their opinions on, and then concludes with an opinion score for each frequent feature. Popescu and Etzioni [11] improved Hu and Liu's work by removing frequent noun phrases that may not be real features. Their method can identify part-of relationship and achieve a better precision, but a small drop in recall. Scaffidi et al. [13] presented a new search system called Red Opal that examined prior customer reviews, identified product features, and then scored each product on each feature. Red Opal used these scores to determine which products to be returned when a user specifies a desired product feature.

However, because above works are all non-model based mining mechanisms, it is unavoidable that their accuracy is limited and many of other types of entities (such as components, functions, non-independent features, etc.) can not be identified through them. Thus, some researchers have attempted to adopt more precise supervised learning models in order to increase the opinion mining efficacy. One typical work is OpinionMiner [6]. It was built based on lexicalized Hidden Markov Model (L-HMMs) which can integrate multiple important linguistic properties into an automatic learning process. However, its limitation is that it cannot represent distributed hidden states and complex interactions among labels. It can neither involve rich, overlapping function sets. That is why in our work we have employed another model, Conditional Random Field (CRFs) [7], because it naturally considers arbitrary, non-independent entities without conditional independence assumption. Although lately some investigators have also attempted to adopt CRFs to perform sentiment analysis [9], they did not use CRFs to identify feature-level polarity orientation, nor use it to extract intensifiers, phrases and infrequent entities. To the best of our knowledge, we are the first one who base this model to address these challenging opinion mining issues. Being different from our prior preliminary work on this subject [12], in this paper, we not only refine the opinion mining process with added new entity categories, learning functions and self-tagging process, but also emphasize its ability in extracting various types of entities.

III. CRFs based Approach for Opinion Mining

In this section, we first introduce the CRFs model and our problem statement. We then in detail describe how we have applied the CRFs in conducting the opinion mining process, including the mining of basic product entities, opinions, intensifiers and phrases.

A. Our Problem Statement

CRFs are conditional probability distributions that factorize according to an undirected model [7]. It is formally defined as follows: considering a graph $G = (V, E)$, let $Y = (Y_v)_{v \in V}$, and (X, Y) is a CRF. X is the set of variables over the observation sequence to be labeled. For example, it can be a sequence of natural language words that form a sentence. Y is the set of random variables over the corresponding labeling sequence which obeys the Markov property with respect to the graph. For example, in our case, Y denotes the tags that are assigned to the words. $p(y|x)$ is globally conditioned on the observation X:

$$p(y|x) = \frac{1}{Z(x)} \prod_{i \in N} \phi_i(y_i, x_i) \qquad (1)$$

where $Z(x) = \sum_y \prod_{i \in N} \phi_i(y_i, x_i)$ is a normalization factor over all states for the sequence X. The potentials are normally factorized according to a set of functions f_k:

$$\phi_i(y_i, x_i) = \exp(\sum_k \lambda_k f_k(y_i, x_i)) \qquad (2)$$

Given the model defined in Equation (1), the most probable labeling sequence for an input X is hence

$$\widehat{Y} = \arg\max_y p(y|x) \qquad (3)$$

Problem statement. When we build the model, we take all the nodes V of the graph as states, including observed states and hidden states. We use W and S to respectively represent the observed states and part-of-speech (POS) tags, and use T to represent the hidden states. The edges E of the graph are relationships among all the states, which are formally defined by the learning functions (see details in Section III.E). Our objective was thus to extract different types of entities from product reviews and identify opinions and intensifiers that are associated with the entities. A review document is thus denoted as W, and if we can give each word a pre-defined tag which is denoted in T, this objective can be achieved. Therefore, the task of opining mining can be regarded as an automatic labeling process, and the challenge is how to define appropriate tags and furthermore precise learning functions so as to maximize the conditional likelihood.

Here we first give an overview to our opinion mining process. It is divided into four major steps: (1) pre-processing, which includes crawling raw review data and cleaning them; (2) preparing the training set to learn the model; (3) defining learning functions for CRFs and training it to maximize the conditional likelihood; and (4) applying the model to label new review data, including their product entities, opinions, intensifiers, and phrases. In the following, we in detail describe each step.

Table I
Five types of entities and their descriptions

Entity	Description and examples
Component	Physical objects of a product, such as the camera itself, the camera's battery, LCD
Function	Capabilities provided by a product, e.g., optical zoom, auto focus, movie playback
Feature	Properties of components or functions, e.g., color, speed, size, weight
Opinion	Ideas and thoughts expressed by reviewers on the product's components, functions, or features, e.g., satisfying, good, easy to use
Intensifier	Words expressed by reviewers to indicate the strength of opinions, e.g., extremely, highly, slightly

B. Pre-Processing

During the pre-processing stage, we first crawled raw and real product reviews from popular e-commerce sites (e.g., Yahoo Shopping and Amazon.com). The raw reviews usually contain some meaningless characters such as '&', '+' and some html tags such as <br>. So we first removed these characters. We also processed word-variants and miss-spelling words, and converted noun words to their stems.

C. Defining Tags and Preparing Training Set

At the second stage, our focus is on defining product entities the algorithm could target to extract, and then labeling the training data with the defined tags. According to [6], the entities that appear in a product review can be classified into four categories : Components, Functions, Features and Opinions. Based on this classification, we add a new category of entities called Intensifiers (e.g., "very", "extremely"), which reviewers have often used to strengthen or weaken the polarities of corresponding opinions. Table I lists these five categories and gives their descriptions and examples.

To tag these entities for forming the model's training set, we designed three types of tags: entity tags, opinion tags and intensifier tags. As for basic product entities, we simply use the category names as tags (e.g., tag "Component" used for component entity, "Function" for function entity, "Feature" for feature entity).

As for opinion entity, besides the "Opinion" tag, we use characters 'P' and 'N' to respectively represent Positive opinion polarity and Negative opinion polarity, and use 'Exp' and 'Imp' to respectively indicate explicit opinion and implicit opinion. Here, explicit opinion means that the user expresses opinion in the review explicitly and implicit opinion means that the opinion needs to be inferred from the review.

As for intensifier entity, besides "Intensifier" tag, we use characters 'S' and 'W' to respectively denote that it strengthens the corresponding opinion (e.g., extremely, highly) and weakens (e.g., slightly, somewhat, sort of).

For a phrase entity, we assign a position to each word of this phrase, which is similar to the method used by [6]. Any word of a phrase has three possible positions: the beginning of the phrase, the middle of the phrase and the end of phrase. We hence use characters 'B', 'M' and 'E' as position tags to respectively indicate these three positions.

For a word that does not belong to any of the above categories, we use "BG" to represent it.

Thus, we have total 15 distinct tags. With these tags, we can label any word and its role in a sentence. For example, the following sentence from a camera's review is labeled as:

The(BG) battery(Feature-B) life(Feature-E) is(BG) really(Intensifier-S) great(Opinion-P-Exp) compared(BG) to(BG) most(BG) cameras(Component) .(BG)

Therefore, for a new un-trained review sentence, if we get its words' tag sequence, we can know which product entities it mentioned, and discover these entities' associated opinions and intensifiers if any. We can also identify whether the entities are expressed in the form of phrases or not. By this way, the task of opining mining can be transformed to an automatic labeling task. The problem is then formalized as follows: given a sequence of words $W = w_1 w_2 w_3 ... w_N$, and its corresponding part-of-speech tags $S = s_1 s_2 s_3 ... s_N$, the objective is to find an appropriate sequence of tags which can maximize the conditional likelihood of Equation (3). The resulting equation is then:

$$\widehat{T} = \arg\max_{T} p(T|W,S) = \arg\max_{T} \prod_{i=1}^{N} p(t_i|W,S,T^{(-i)}) \tag{4}$$

In Equation (4), $T^{(-i)} = \{t_1 t_2 ... t_{i-1} t_{i+1} ... t_N\}$ (being exclusive of the current word's tag t_i). From this equation, we can see that the tag of a word at position i depends on all the words $W = w_{1:N}$ in that sentence, part-of-speech $S = s_{1:N}$ and tags on other words. Unfortunately, it is very hard to compute with this equation as it involves too many variables. To reduce the complexity, we employ linear-chain CRFs as an approximation to restrict the relationship among tags. In the linear-chain CRFs (see Figure 1), all the hidden states T in the graph form a linear chain and only two consecutive states are connected. If we further assume that the current state is only dependent on the previous one, Equation (4) can be rewritten as:

$$\widehat{T} = \arg\max_{T} p(T|W,S) = \arg\max_{T} \prod_{i=1}^{N} p(t_i|W,S,t_{i-1}) \tag{5}$$

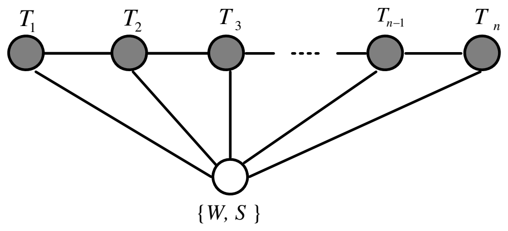

Figure 1. Linear-chain CRFs graph structure.

D. Word Expansion and Self-Tagging Process

In order to minimize the human effort in labeling and preparing the training data, we conducted two processes to automatically enlarge the labeled set. First of all, for a single entity word or an opinion that appears in a sentence, we substitute it with its synonymous word and remain the tag unchanged. The synonymous words are from the Microsoft Word's Thesaurus. For example, for the sentence "it shoots good image", we first obtained the synonym set of "good", which is "great", "nice", "cracking", and the synonym set of "image", which is "picture", "photo". We then list all the combinations of these words such as "nice photo", "great picture", etc., and add them in the training dataset.

In addition, we propose a self-tagging (ST) technique. That is, we only manually label three sets of training data and the other training data will be automatically tagged by the system. The process concretely works as follows:

1) Evenly divide the manually labeled data into three groups, and give each group a number (with numbers 1, 2, 3);
2) Conduct training process on the three groups separately so as to get 3 different training models. We then use the three models to self-tag sentences from the un-labeled training data. After this process, we can get three tagging results for a sentence;
3) Compare the three tagging results and select the tags based on this criterion: if at least two of the three tagging results hold agreement on the same sentence (that is, they produced the same tags to the concerned words of the sentence);
4) Randomly assign the newly tagged sentences into the three groups and then go back to step 2. Repeat the process until all the other training data are self-tagged.

Actually, we relaxed the strict constraints of bootstrapping method that was proposed in [6]: 1) they divided the manually labeled data into two groups and required that the agreement must be achieved between two results. In our algorithm, we divided it into three groups and accepted if the agreement is only between two of the three groups. 2) To compute the agreement degree, we emphasized the concerned entities (e.g., component, function, feature, opinion, intensifier), and did not require that all (including background tags) should be matched. With these refinements, we expected that the data augmentation process could be more effectively performed.

E. Defining Learning Functions for the CRFs Model

The next step is then to define learning functions for the CRFs model. As indicated before, the functions define the types of relationships between observation states $W = w_{1:N}$, $S = s_{1:N}$ (i.e., the words' sequence in a sentence and their corresponding part-of-speech tags) and hidden states $T = t_{1:N}$ (the tags' sequence). Therefore, the functions are very crucial for determining the model's final accuracy. In our case with the linear-chain CRFs (see Equation (5)), the general form of a function is $f_i(t_{j-1}, t_j, w_{1:N}, s_{1:N}, j)$, which looks at a pair of adjacent states t_{j-1}, t_j, the whole input sequence $w_{1:N}$ and $s_{1:N}$, and the current word's position j. For example, we can define a function that produces the binary value: 1 if the current word w_j is "image", the corresponding part-of-speech s_j is "NN" (i.e., single noun word), the previous state t_{j-1} is "Opinion", and the current state t_j is "Feature"; 0 otherwise.

$$f_i(t_{j-1}, t_j, w_{1:N}, s_{1:N}, j) = \begin{cases} 1 & if\ w_j = image, \\ & s_j = NN, t_j = Feature, \\ & t_{j-1} = Opinion; \\ 0 & Otherwise \end{cases} \tag{6}$$

Combining this function with Equation (1) and Equation (2), we have:

$$\begin{aligned} & p(t_{1:N} | w_{1:N}, s_{1:N}) \\ & = \frac{1}{Z} \exp\left(\sum_{j=1}^{N} \sum_{i=1}^{F} \lambda_i f_i(t_{j-1}, t_j, w_{1:N}, s_{1:N}, j)\right) \end{aligned} \tag{7}$$

According to Equation (7), each function f_i is associated with a weight λ_i. That is, if $\lambda_i > 0$, and f_i is active (i.e., $f_i = 1$), it will increase the probability of a potentially related tag. On the contrary, if $\lambda_i < 0$, or f_i is inactive (i.e., $f_i = 0$), it will decrease the probability of the tag.

Another function example is:

$$f_i(t_{j-1}, t_j, w_{1:N}, s_{1:N}, j) = \begin{cases} 1 & if\ w_{j-1} = good, \\ & s_{j-1} = JJ, t_j = Feature; \\ 0 & Otherwise \end{cases} \tag{8}$$

With these two functions (6) and (8), if the current word is "image" such as in "good image", they will both be active, so the probability of assigning tag "Feature" to the word "image" will be increased. This is an example of overlapping functions that L-HMMs cannot address.

Specifically, we have defined several types of functions to specify the state-transition structures between W/S and T, according to the Markov orders. For instance, the first-order functions are defined as: 1) the assignment of current tag t_j only depends on the current word. The function is hence represented as $f(t_j, w_j)$. 2) The assignment of current tag t_j only depends on the current part-of-speech tag. The function is represented as $f(t_j, s_j)$. 3) The assignment of current tag t_j depends on both the current word and the current part-of-speech tag. The function is represented as $f(t_j, s_j, w_j)$. These three functions are first-order, by which the inputs are examined in the context of the current state only. In addition, we defined first-order plus transitions, second-order functions and third-order functions (see Table II), by which the inputs are examined in the context of both the current state and previous one or two states. In order to evaluate the impact of these functions on opinion mining results, we tested various combinations in the experiment and were able to identify the optimal function set (see the results in Section V).

Table II
VARIOUS FUNCTION TYPES AND EXPRESSIONS

Type	Function Expression
1. First-Order	$f(t_i, w_i)$, $f(t_i, s_i)$, $f(t_i, w_i, s_i)$
2. First-order+ Transition	$f(t_i, w_i)f(t_i, t_{i-1})$, $f(t_i, s_i)f(t_i, t_{i-1})$, $f(t_i, w_i, s_i)f(t_i, t_{i-1})$
3. First-order+ Transition	$f(t_i, w_i)f(t_i, t_{i-2})$, $f(t_i, s_i)f(t_i, t_{i-2})$, $f(t_i, w_i, s_i)f(t_i, t_{i-2})$
4. First-order+ Transition	$f(t_i, w_{i-1})f(t_i, t_{i-1})$, $f(t_i, s_{i-1})f(t_i, t_{i-1})$, $f(t_i, w_{i-1}, s_{i-1})f(t_i, t_{i-1})$
5. First-order+ Transition	$f(t_i, w_{i-1})f(t_i, t_{i-2})$, $f(t_i, s_{i-1})f(t_i, t_{i-2})$, $f(t_i, w_{i-1}, s_{i-1})f(t_i, t_{i-2})$
6. Second-Order	$f(t_i, t_{i-1}, w_i)$, $f(t_i, t_{i-1}, s_i)$, $f(t_i, t_{i-1}, w_i, s_i)$
7. Second-Order	$f(t_i, t_{i-2}, w_i)$, $f(t_i, t_{i-2}, s_i)$, $f(t_i, t_{i-2}, w_i, s_i)$
8. Third-Order	$f(t_i, t_{i-1}, t_{i-2}, w_i)$, $f(t_i, t_{i-1}, t_{i-2}, s_i)$, $f(t_i, t_{i-1}, t_{i-2}, w_i, s_i)$

F. Training CRFs Model

After the graph and functions are defined, we can start training the CRFs model. The purpose of training is in essence to determine the parameter values for $\lambda_{1:F}$ (i.e., the weights of learning functions such as in Equation (7)). Formally, the fully labeled review data is denoted as $\{(w^{(1)}, s^{(1)}, t^{(1)}), ..., (w^{(M)}, s^{(M)}, t^{(M)})\}$, where $w^{(j)} = w^{(j)}_{1:N_j}$ (the j-th sentence), $s^{(j)} = s^{(j)}_{1:N_j}$ (the j-th part-of-speech sequence), and $t^{(j)} = t^{(j)}_{1:N_j}$ (the j-th tags' sequence). Given that in CRFs we defined the conditional probability $p(t|w, s)$, the aim of parameter learning is to maximize the conditional likelihood according to this Equation:

$$\sum_{j=1}^{M} \log p(\mathbf{t}^{(j)} | \mathbf{w}^{(j)}, \mathbf{s}^{(j)}) \tag{9}$$

To avoid over-fitting, the likelihood can be penalized by some prior distributions over the parameters. A commonly used distribution is zero-mean Gaussian: if $\lambda \sim N(0, \sigma^2)$, Equation (9) will become

$$\sum_{j=1}^{M} \log p(t^{(j)}|w^{(j)}, s^{(j)}) - \sum_{i}^{F} \frac{\lambda_i^2}{2\sigma^2} \tag{10}$$

This equation is concave, so λ has a unique set of global optimal values. We have thus learnt parameters by computing the gradient of the objective function, and applied the gradient in an optimization algorithm called Limited memory BFGS (L-BFGS) [8]. Specifically, the gradient of the objective function is computed as:

$$\begin{aligned}
&\tfrac{\partial}{\partial \lambda_k} \sum_{j=1}^{m} \log p(t^{(j)}|w^{(j)}, s^{(j)}) - \sum_{i}^{F} \tfrac{\lambda_i^2}{2\sigma^2} \\
&= \tfrac{\partial}{\partial \lambda_k} \sum_{j=1}^{m} (\sum_n \sum_i \lambda_i f_i(t_{n-1}, t_n, w_{1:N}, s_{1:N}, n) \\
&- \log Z^{(j)}) - \sum_{i}^{F} \tfrac{\lambda_i^2}{2\sigma^2} \\
&= \sum_{j=1}^{m} \sum_n f_k(t_{n-1}, t_n, w_{1:N}, s_{1:N}, n) \\
&- \sum_{j-1}^{m} \sum_n E_{t'_{n-1}, t'_n}[f_k(t'_{n-1}, t'_n, w_{1:N}, s_{1:N}, n)] - \tfrac{\lambda_k}{\sigma^2}
\end{aligned} \tag{11}$$

In Equation (11), the first term is the actual times that a function f_i is active (i.e., $f_i = 1$) in the training data. The second term is the predicted activation times of this function under the current trained model. The third term is generated by the prior distribution. Hence, this derivative measures the difference between the actual frequency and the predicted frequency. Suppose that in the training data, a function f_k is active at A times, while under the current model, the predicted activations are B times: when $|A| = |B|$, the derivative is zero. Therefore, the training process is to find λs that can minimize the derivative.

G. Decoding and Determining Orientations for Opinions and Intensifiers

After the parameters were learnt, the model is fixed. We can then apply it to process new review data. As we discussed above, our objective is to apply the model to automatically label the words with the most probable tags. This requires that the conditional likelihood (as defined in Equation (4)) can be maximized at each step. More specifically, suppose that the current word's position is j and there are M different tags that can be candidates for this word, we can get the Viterbi variables $\alpha_j(m) = p(W, S, t_j = m)$ $(m \in M)$. The Viterbi recursion is obtained with:

$$\alpha_j(m) = \max_{m \in M} \varphi_j(W, S, m', m)\alpha_{j-1}(m') \tag{12}$$

where $\varphi_j(W, S, m', m)$ is the function on the transition from state m' to state m when the observations are W, S. In our work, the transition function is formally defined as:

$$\varphi_j(W, S, m', m) = \exp(\sum_k \lambda_k f_k(W, S, t_{j-1} = m', t_j = m, j)) \tag{13}$$

Here we give some examples that were resulted from the decoding process:

I(BG) love(Opinion-P-Exp) the(BG) image(Feature-B) quality(Feature-E) of(BG) this(BG) camera(Component) .(BG)

I(BG) am(BG) absolutely(Intensifier-S) in(Opinion-B-P-Exp) love(Opinion-M-P-Exp) of(Opinion-E-P-Exp) this(BG) camera(Component) .(BG)

Buttons(Component) on(BG) the(BG) menu(Function) are(BG) easy(Opinion-B-P-Exp) to(Opinion-M-P-Exp) use(Opinion-E-P-Exp) .(BG)

From these examples we can see that when the labeling tags and learning functions are defined, it is feasible to extract product entities, and the entity-associated opinions and intensifiers, from a review.

As the final step of the process, we checked the appearance of negations (e.g., not, don't, no, didn't). If they are within five-word distance in front of an opinion/intensifier and their total number is odd, the final orientation for the corresponding opinion/intensifer will be reversed.

IV. Experiment Setup

We conducted an experiment that systematically compared the CRFs based opinion mining algorithm with the most related one: the L-HMMs based method [6] as it is also a supervised model-based learning technique. We also used the rule-based method as the baseline. The reason that we did not compare our method with the statistical methods (such as [5]) was because they treated all product entities (e.g., component, function) as features, rather than handling them separately. Moreover, they did not target to extract phrases and intensifiers.

Concretely, we evaluated these approaches' performance with respect to three metrics: recall, precision and F-measure. Recall is $\frac{|C \cap P|}{|C|}$ and Precision is $\frac{|C \cap P|}{|P|}$, where C and P are the sets of true and predicted tags respectively. F-measure is the harmonic mean of precision and recall, i.e., $\frac{2RP}{R+P}$ (where R is the recall value and P is the precision value). We did not use label accuracy to test each tag, because for some tags (such as "BG" used to denote background words), it is not very meaningful to evaluate them. Instead, we emphasize the precision, recall and F-score results on the five types of entities, e.g., Component, Function, Feature, Opinion, Intensifier, when comparing the CRFs based opinion mining method with the L-HMMs based one.

Our dataset was concretely crawled from Yahoo shopping and Amazon. It also contains the corpus shared from [5]. In total, the dataset has 821 reviews (1775 sentences) for 9 digital cameras. After pre-processing and cleaning the raw data, two researchers were involved in labeling them. They first independently labeled the data, and then met together to aggregate their results. We applied the LBJPOS tool [1] to produce the part-of-speech tag for each word.

All labeled data were then divided into 10 sets to perform the 10-fold cross-validation: one set was used as the validation data for testing, and the other 9 sets were used as training data (three training sets were automatically labeled through our self-tagging technique as described in Section III.D). The cross-validation process was repeated for ten times, and every time one set was randomly selected as the testing data. Afterwards, the results were averaged to calculate the final precision, recall and F-measure scores.

A. Compared Methods in the Experiment

Before showing the experimental results, we first describe the methods that were compared to our CRFs based approach.

1) Rule-based Method as Baseline: Motivated by [14], we designed a rule-based method as the baseline approach for comparison. The first step of this method was performing Part-of-Speech (POS) task. Each word can get a tag of POS such as NN (noun word), JJ (adjective word), etc. We then applied several classic rules to extract objective product entities [5][6]. One rule is that a single noun that follows an adjective word or consecutive adjective words will be regarded as a product entity, i.e., NN in JJ + NN or JJ + JJ + NN. Any single noun word that connects an adjective word to a verb will also be taken as a product entity, i.e., NN in NN + VBZ +JJ. Any consecutive noun words that appear at above positions will be taken as an entity phrase.

As for opinion words, the adjective words that appear in above rules will be opinion entities, and their sentimental orientation was determined by a lexicon that contains polarities for over 8000 adjective words [2].

We also defined a rule to extract intensifier: the adverb that is within the 3-word distance to the opinion entity is regarded as an intensifier.

2) L-HMMs Based Opinion Mining Method: As noted before, [6] integrated linguistic properties, including part-of-speech results and lexical patterns, into Hidden Markov Model (HMMs) model. Their aim was also to maximize the conditional probability defined in Equation (4). According to the Bayes law, they rewrote the equation as:

$$\widehat{T} = \arg\max_{T} \frac{p(W,S|T)p(T)}{P(W,S)}$$

Because the value of $P(W,S)$ remains constant for all candidate tag sequences, they produced a general statistical model:

Table III
WINNING COMBINATION OF FUNCTIONS DURING EACH ROUND (P: PRECISION, R: RECALL, F: F-MEASURE)

Round	Function combination (the number refers to Table II)	Average accuracy		
		P	R	F
The 1st	1	0.894	0.789	0.837
The 2nd	**1 + 3 + 5**	0.906	0.815	0.858
The 3rd	1 + 3 + 5 + 6 + 7	0.905	0.814	0.857
The 4th	1 + 3 + 5 + 6 + 7 + 8	0.893	0.801	0.844

$$\widehat{T} = \arg\max_{T} p(W,S|T)p(T) = \arg\max_{T} p(S|T)p(W|T,S)p(T)$$
$$= \arg\max_{T} \prod_{i=1}^{N} \left\{ \begin{array}{l} p(s_i|w_1...w_{i-1}, s_1...s_{i-1}, t_1...t_{i-1}t_i)\times \\ p(w_i|w_1...w_{i-1}, s_1...s_{i-1}s_i, t_1...t_{i-1}t_i)\times \\ p(t_i|w_1...w_{i-1}, s_1...s_{i-1}, t_1...t_{i-1}) \end{array} \right\}$$

In order to further simplify the computation, they made several assumptions. For example, the assignment of the current tag depends only on its previous tag and the previous word. The appearance of the current word depends on the current tag, the current POS, and the previous word. Their final objective was then to maximize:

$$\arg\max_{T} \prod_{i=1}^{N} \left\{ \begin{array}{l} p(s_i|w_{i-1}, t_i)\times \\ p(w_i|w_{i-1}, s_i, t_i)\times \\ p(t_i|w_{i-1}, t_{i-1}) \end{array} \right\}$$

Maximum Likelihood Estimation (MLE) was used to estimate the parameters. Other techniques were also used in their approach, including information propagation (similar to our word expansion process) and bootstrapping phase.

In our experiment, in order to make the CRFs-based and L-HMMs based learning methods vary only in terms of the model itself, we used the same definitions of tags (see Section III.C) and conducted similar word expansion and self-tagging processes (see Section III.D) in both approaches.

V. EXPERIMENTAL RESULTS AND DISCUSSION

A. Testing on CRFs Functions

As defined in Table II, we have eight types of CRFs learning functions in total: one first-order, four first-order plus transition, two second-order, and one third-order. We assigned each type a number from 1 to 8 (see Table II). We varied their combinations and used them in the training process. Specifically, in the first round, we only used the first-order functions. Then, in the second round, various combinations of first-order plus transition functions were respectively added with the first-order functions (i.e., 1+2, 1+3, 1+4, 1+5, 1+2+3, 1+2+4, 1+2+5, 1+3+4, ..., 1+2+3+4+5). As a result, 1+3+5 was found with higher precision, recall and F-measure scores, against most of other combinations respecting all types of entities (see Table III). In fact, another combination 1+2+3+4+5 reached at similar accuracy level (and for some values, it is even slightly better), but considering that it demands higher time complexity, we chose the less complex one, 1+3+5.

Table IV
COMPARISON RESULTS REGARDING COMPONENT, FUNCTION, FEATURE, OPINION AND INTENSIFIER (P: PRECISION, R: RECALL, F: F-MEASURE)

Method	Component			Function			Feature			Overall			Opinion			Intensifier		
	P	R	F	P	R	F	P	R	F	P	R	F	P	R	F	P	R	F
CRFs	0.911	0.827	0.867	0.843	0.675	0.749	0.907	0.828	0.862	0.887	0.777	0.826	0.895	0.822	0.859	0.912	0.832	0.870
L-HMMs	0.786	0.647	0.709	0.661	0.475	0.551	0.878	0.762	0.815	0.775	0.628	0.692	0.895	0.790	0.839	0.897	0.792	0.841
Rule-Based	-	-	-	-	-	-	-	-	-	0.285	0.258	0.256	0.296	0.214	0.248	0.200	0.141	0.166
p value (by ANOVA)	$<$.001	$<$.001	$<$.001	$<$.001	$<$.001	$<$.001	$<$.05	$<$.05	$<$.05	$<$.001	$<$.001	$<$.001	$<$.001	$<$.001	$<$.001	$<$.001	$<$.001	$<$.001

The combination 1+3+5 was further integrated with second-order functions (i.e., 1+3+5+6, 1+3+5+7, 1+3+5+6+7). The winning combination from the third round was then integrated with the third-order function (i.e., 1+3+5+6+7+8).

Table III lists the best combination resulted from each round. Comparison of their accuracy shows that the involvement of second-order and third-order functions did not help much on increasing the accuracy (but with the cost of higher time complexity). We have thus eventually fixed the functions' set to be 1+3+5, which contains the first-order functions and two types of first-order plus transition.

B. CRFs based vs. L-HMMs based Opinion Mining Methods

After determining the optimal learning functions, as the next experiment step, we compared the CRFs-based approach (henceforth CRFs) with L-HMMs based method (henceforth L-HMMs) from several aspects: basic product entities (i.e., component, function, feature), opinions, intensifiers, phrases, and infrequent entities.

1) Basic Product Entities: The Table IV lists the comparison results after 10-fold cross validation, from which we can see that CRFs achieve better accuracy results on all the basic product entities (i.e., Component, Function and Feature), than L-HMMs. In fact, most of its values are above 80% accuracy level. The ANOVA analysis further indicated that the differences are significant (the *p* values are less than 0.05) regarding all precisions, recalls and F-measure scores.

Because rule-based method did not allow to handle these entities individually, we calculated its overall performance on all entities and compared to the average values by CRFs and L-HMMs. Still, the comparison among them shows that the differences are significant. Rule-based is in fact very less accurate than the model-based approaches.

2) Opinions and Intensifiers: As for opinion and intensifier, CRFs also outperform L-HMMs and rule-based method. Table IV gives the significant *p* values (by ANOVA) from comparing these three methods. We further did a pair-wise comparison just between CRFs and L-HMMs. It indicates that CRFs is significantly better than L-HMMs, in respect of opinion's recall (i.e., increasing the accuracy from 79.0% to 82.2%, $p < 0.05$), and intensifier's recall (from 79.2% to 83.2%, $p < 0.05$) and F-measure score (from 84.1% to 87.0%, $p < 0.1$).

Combining with results from the last section thus implies that CRFs can be competent and flexible in extracting various kinds of entities. It even reaches at significantly better accuracy level than the L-HMMs based method, regarding all the basic entities, opinion and intensifier. The findings hence well demonstrate our hypothesis that the CRFs model is more effective in accomplishing the opinion mining tasks.

Table V
COMPARISON RESULTS REGARDING PHRASE AND INFREQUENT ENTITY (P: PRECISION, R: RECALL, F: F-MEASURE)

Method	Phrase			Infrequent Entity	
	P	R	F	Hit Ratio	Average Ratio
CRFs	0.825	0.700	0.757	0.908	0.781
L-HMMs	0.800	0.667	0.726	0.902	0.766
p value (by ANOVA)	.18	.11	.088	.67	.23

3) Phrases and Infrequent Entities: We further compared CRFs and L-HMMs regarding their abilities in determining phrases and infrequent entities. Because it is implausible to define appropriate rules for these items, the rule-based method was not involved in the comparison. Table V reports the results.

Specifically, a phrase was defined as a set of two or more consecutive words that include the beginning, (middle) and end position tags. It can not only be a product entity (e.g., "image quality"), but also be an opinion (e.g., "easy to use", "easy to navigate"). The results show that the precision, recall and F-measure have higher values with CRFs, than with L-HMMs. The difference regarding F-measure is moderately significant ($p < 0.1$).

To define infrequent entities, we took entities with the appearing percentage below or equal to 1% over the whole dataset as infrequent entities. We used the hit-ratio R_{hit} and the average ratio R_{ave} to assess the algorithm's accuracy in discovering them. That is, assume there are N_{all} infrequent entities, and N_{find} of them were correctly extracted by the evaluated algorithm, $R_{hit} = N_{find}/N_{all}$. Average ratio refers that for the discovered infrequent entities, the average times that the algorithm is able to correctly identify them: $R_{ave} = \frac{1}{N_{all}} \sum_{N_{all}} \frac{F_i}{C_i}$, where C_i indicates the actual appearance times of i-th entity, and F_i is the times that the algorithm identified it.

From Table V, we can see that CRFs performs slightly better than L-HMMs in terms of both hit ratio and average ratio, though the differences are not significant.

4) Discussion: We believe that there are two major reasons that enable CRFs-based approach to achieve better performance in respect of all measures. The first is that the CRFs model can naturally incorporate arbitrary, non-independent entities of the input reviews, without making conditional independence assumptions among them. The second is that it can involve rich and overlapping learning functions, which should help to discover phrases and infrequent entities. For example, although the entity "ISO" only appears once in our data, more than one functions might be active in finding it (so its exposure probability is increased).

VI. Conclusion and Future Work

To conclude our work, in this paper, we described how the CRFs model was applied and enhanced for making feature-level Web opinion mining. The experiment demonstrated its outperforming effectiveness against another model-based opinion mining approach. Specifically, contrasting to L-HMMs which holds conditional independence assumption, CRFs can handle arbitrary, non-independent input entities and integrate rich learning functions to discover relationships among tags. Moreover, in this paper, we introduced how we have attempted to include self-tagging process to reduce the manual labeling effort, so that the supervised model-based approach could be applicable in the Web environment to process large amount of reviews.

Encouraged by these findings, in the future, we will conduct more experiments to compare CRFs-based approach with others. For instance, it will be meaningful to compare the model-based opinion mining technique with non-model based ones such as the statistical methods proposed in [5] and [11]. We will conduct detailed comparison to investigate the two types of approaches in the condition that the scale of data varies. The hypothesis we will be engaged in testing is: model-based learning approaches would be more accurate because it involves the model's optimization process (though the training effort is required). We will try to test how much human effort would be optimally saved in making the training data, so that the tradeoff between accuracy and effort could achieve an ideal balance.

References

[1] http://cogcomp.cs.illinois.edu/page/software_view/3.

[2] http://www.cs.cornell.edu/People/pabo/movie-review-data/.

[3] S. Das and M. Chen. Yahoo! for Amazon: Sentiment extraction from small talk on the web. *Management Science*, 53(9):1375–1388, 2007.

[4] V. Hatzivassiloglou and J. Wiebe. Effects of adjective orientation and gradability on sentence subjectivity. In *Proceedings of the 18th Conference on Computational linguistics-Volume 1*, pages 299–305. Association for Computational Linguistics, 2000.

[5] M. Hu and B. Liu. Mining and summarizing customer reviews. In *Proceedings of Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 168–177. ACM, 2004.

[6] W. Jin, H. Ho, and R. Srihari. OpinionMiner: a novel machine learning system for web opinion mining and extraction. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1195–1204. ACM, 2009.

[7] J. Lafferty, A. McCallum, and F. Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the Eighteenth International Conference on Machine Learning*, ICML '01, pages 282–289, San Francisco, CA, USA, 2001. Morgan Kaufmann Publishers Inc.

[8] D. Liu and J. Nocedal. On the limited memory bfgs method for large scale optimization. *Mathematical programming*, 45(1):503–528, 1989.

[9] Q. Miao, Q. Li, and D. Zeng. Mining fine grained opinions by using probabilistic models and domain knowledge. In *Proceedings of the 2010 IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology - Volume 01*, WI-IAT '10, pages 358–365, Washington, DC, USA, 2010. IEEE Computer Society.

[10] B. Pang, L. Lee, and S. Vaithyanathan. Thumbs up?: sentiment classification using machine learning techniques. In *Proceedings of the ACL-02 Conference on Empirical Methods in Natural Language Processing-Volume 10*, pages 79–86. Association for Computational Linguistics, 2002.

[11] A. Popescu and O. Etzioni. Extracting product features and opinions from reviews. In *Proceedings of the conference on Human Language Technology and Empirical Methods in Natural Language Processing*, pages 339–346. Association for Computational Linguistics, 2005.

[12] L. Qi and L. Chen. A linear-chain CRF-Based learning approach for Web opinion mining. In *Proceedings of 11th International Conference on Web Information System Engineering (WISE?0)*, pages 128–141. Springer, 2010.

[13] C. Scaffidi, K. Bierhoff, E. Chang, M. Felker, H. Ng, and C. Jin. Red Opal: product-feature scoring from reviews. In *Proceedings of the 8th ACM Conference on Electronic commerce*, pages 182–191. ACM, 2007.

[14] P. Turney. Thumbs up or thumbs down?: semantic orientation applied to unsupervised classification of reviews. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, pages 417–424. Association for Computational Linguistics, 2002.

[15] T. Wilson, P. Hoffmann, S. Somasundaran, J. Kessler, J. Wiebe, Y. Choi, C. Cardie, E. Riloff, and S. Patwardhan. OpinionFinder: A system for subjectivity analysis. In *Proceedings of HLT/EMNLP on Interactive Demonstrations*, pages 34–35. Association for Computational Linguistics, 2005.