Phrase Dependency Parsing for Opinion Mining

Yuanbin Wu, Qi Zhang, Xuanjing Huang, Lide Wu
Fudan University
School of Computer Science
{ybwu,qi_zhang,xjhuang,ldwu}@fudan.edu.cn

Abstract

In this paper, we present a novel approach for mining opinions from product reviews, where it converts opinion mining task to identify product features, expressions of opinions and relations between them. By taking advantage of the observation that a lot of product features are phrases, a concept of phrase dependency parsing is introduced, which extends traditional dependency parsing to phrase level. This concept is then implemented for extracting relations between product features and expressions of opinions. Experimental evaluations show that the mining task can benefit from phrase dependency parsing.

1 Introduction

As millions of users contribute rich information to the Internet everyday, an enormous number of product reviews are freely written in blog pages, Web forums and other consumer-generated mediums (CGMs). This vast richness of content becomes increasingly important information source for collecting and tracking customer opinions. Retrieving this information and analyzing this content are impossible tasks if they were to be manually done. However, advances in machine learning and natural language processing present us with a unique opportunity to automate the decoding of consumers' opinions from online reviews.

Previous works on mining opinions can be divided into two directions: sentiment classification and sentiment related information extraction. The former is a task of identifying positive and negative sentiments from a text which can be a passage, a sentence, a phrase and even a word (Somasundaran et al., 2008; Pang et al., 2002; Dave et al., 2003; Kim and Hovy, 2004; Takamura et al., 2005). The latter focuses on extracting the elements composing a sentiment text. The elements include source of opinions who expresses an opinion (Choi et al., 2005); target of opinions which is a receptor of an opinion (Popescu and Etzioni, 2005); opinion expression which delivers an opinion (Wilson et al., 2005b). Some researchers refer this information extraction task as opinion extraction or opinion mining. Comparing with the former one, opinion mining usually produces richer information.

In this paper, we define an opinion unit as a triple consisting of a product feature, an expression of opinion, and an emotional attitude(positive or negative). We use this definition as the basis for our opinion mining task. Since a product review may refer more than one product feature and express different opinions on each of them, the relation extraction is an important subtask of opinion mining. Consider the following sentences:

1. I highly [recommend]$^{(1)}$ the Canon SD500$^{(1)}$ to anybody looking for a compact camera that can take [good]$^{(2)}$ pictures$^{(2)}$.

2. This camera takes [amazing]$^{(3)}$ image qualities$^{(3)}$ and its size$^{(4)}$ [cannot be beat]$^{(4)}$.

The phrases underlined are the product features, marked with square brackets are opinion expressions. Product features and opinion expressions with identical superscript compose a relation. For the first sentence, an opinion relation exists between "*the Canon SD500*" and "*recommend*", but not between "*picture*" and "*recommend*". The example shows that more than one relation may appear in a sentence, and the correct relations are not simple Cartesian product of opinion expressions and product features.

Simple inspection of the data reveals that product features usually contain more than one word, such as "LCD screen", "image color", "Canon PowerShot SD500", and so on. An incomplete product feature will confuse the successive analysis. For example, in passage *"Image color is dis-*

Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing, pages 1533–1541,
Singapore, 6-7 August 2009.

appointed", the negative sentiment becomes obscure if only "*image*" or "*color*" is picked out.

Since a product feature could not be represented by a single word, dependency parsing might not be the best approach here unfortunately, which provides dependency relations only between words. Previous works on relation extraction usually use the head word to represent the whole phrase and extract features from the word level dependency tree. This solution is problematic because the information provided by the phrase itself can not be used by this kind of methods. And, experimental results show that relation extraction task can benefit from dependencies within a phrase.

To solve this issue, we introduce the concept of phrase dependency parsing and propose an approach to construct it. Phrase dependency parsing segments an input sentence into "phrases" and links segments with directed arcs. The parsing focuses on the "phrases" and the relations between them, rather than on the single words inside each phrase. Because phrase dependency parsing naturally divides the dependencies into local and global, a novel tree kernel method has also been proposed.

The remaining parts of this paper are organized as follows: In Section 2 we discuss our phrase dependency parsing and our approach. In Section 3, experiments are given to show the improvements. In Section 4, we present related work and Section 5 concludes the paper.

2 The Approach

Fig. 1 gives the architecture overview for our approach, which performs the opinion mining task in three main steps: (1) constructing phrase dependency tree from results of chunking and dependency parsing; (2) extracting candidate product features and candidate opinion expressions; (3) extracting relations between product features and opinion expressions.

2.1 Phrase Dependency Parsing

2.1.1 Overview of Dependency Grammar

Dependency grammar is a kind of syntactic theories presented by Lucien Tesnière(1959). In dependency grammar, structure is determined by the relation between a head and its dependents. In general, the dependent is a modifier or complement; the head plays a more important role in determining the behaviors of the pair. Therefore, criteria of how to establish dependency relations and how to distinguish the head and dependent in such relations is central problem for dependency grammar. Fig. 2(a) shows the dependency representation of an example sentence. The root of the sentence is "enjoyed". There are seven pairs of dependency relationships, depicted by seven arcs from heads to dependents.

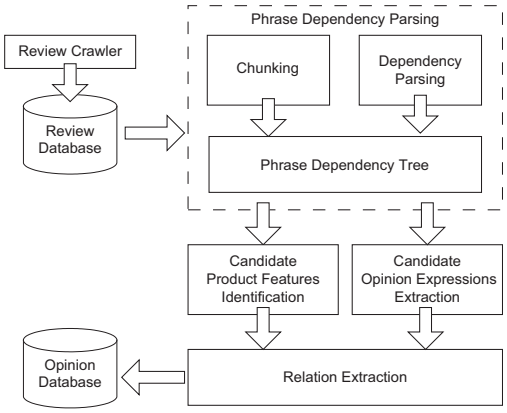

Figure 1: The architecture of our approach.

2.1.2 Phrase Dependency Parsing

Currently, the mainstream of dependency parsing is conducted on lexical elements: relations are built between single words. A major information loss of this word level dependency tree compared with constituent tree is that it doesn't explicitly provide local structures and syntactic categories (i.e. NP, VP labels) of phrases (Xia and Palmer, 2001). On the other hand, dependency tree provides connections between distant words, which are useful in extracting long distance relations. Therefore, compromising between the two, we extend the dependency tree node with phrases. That implies a noun phrase "Cannon SD500 PowerShot" can be a dependent that modifies a verb phrase head "really enjoy using" with relation type "dobj". The feasibility behind is that a phrase is a syntactic unit regardless of the length or syntactic category (Santorini and Kroch, 2007), and it is acceptable to substitute a single word by a phrase with same syntactic category in a sentence.

Formally, we define the dependency parsing with phrase nodes as *phrase dependency parsing*. A dependency relationship which is an asymmetric binary relationship holds between two phrases. One is called head, which is the central phrase in the relation. The other phrase is called dependent, which modifies the head. A label representing the

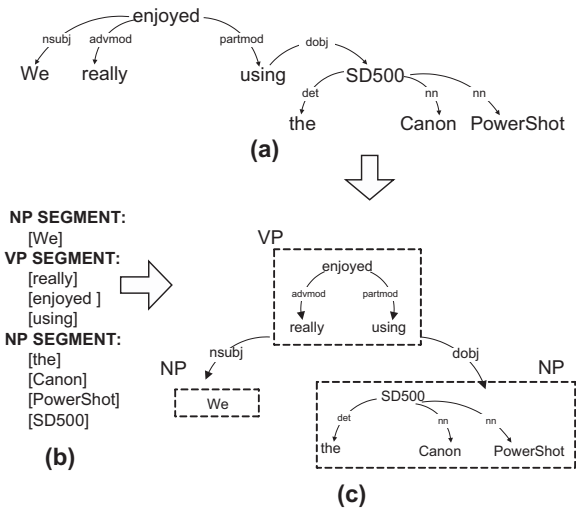

Figure 2: Example of Phrase Dependency Parsing.

relation type is assigned to each dependency relationship, such as subj (subject), obj (object), and so on. Fig.2(c) shows an example of phrase dependency parsing result.

By comparing the phrase dependency tree and the word level dependency tree in Fig.2, the former delivers a more succinct tree structure. Local words in same phrase are compacted into a single node. These words provide local syntactic and semantic effects which enrich the phrase they belong to. But they should have limited influences on the global tree topology, especially in applications which emphasis the whole tree structures, such as tree kernels. Pruning away local dependency relations by additional phrase structure information, phrase dependency parsing accelerates following processing of opinion relation extraction .

To construct phrase dependency tree, we propose a method which combines results from an existing shallow parser and a lexical dependency parser. A phrase dependency tree is defined as $\mathscr{T} = (\mathscr{V}, \mathscr{E})$, where $\mathscr{V}$ is the set of phrases, $\mathscr{E}$ is the dependency relations among the phrases in $\mathscr{V}$ representing by direct edges. To reserve the word level dependencies inside a phrase, we define a nested structure for a phrase T_i in $\mathscr{V}$: $T_i = (V_i, E_i)$. $V_i = \{v_1, v_2, \cdots, v_m\}$ is the internal words, E_i is the internal dependency relations.

We conduct the phrase dependency parsing in this way: traverses word level dependency tree in preorder (visits root node first, then traverses the children recursively). When visits a node R, searches in its children and finds the node set D which are in the same phrase with R according to the shallow parsing result. Compacts D and R into a single node. Then traverses all the remaining children in the same way. The algorithm is shown in Alg. 1.

Algorithm 1 Pseudo-Code for constructing the phrase dependency tree

INPUT:
$T' = (V', E')$ a word level dependency tree
P = phrases
OUTPUT:
phrase dependency tree $\mathscr{T} = (\mathscr{V}, \mathscr{E})$ where
$\mathscr{V} = \{T_1(V_1, E_1), T_2(V_2, E_2), \cdots, T_n(V_n, E_n)\}$
Initialize:
$\mathscr{V} \leftarrow \{(\{v'\}, \{\}) | v' \in V'\}$
$\mathscr{E} \leftarrow \{(T_i, T_j) | (v'_i, v'_j) \in E', v'_i \in V_i, v'_j \in V_j\}$
$R = (V_r, E_r)$ root of $\mathscr{T}$
PhraseDPTree(R, P)

```
Find p_i ∈ P where word[R] ∈ p_i
for each S = (V_s, E_s), (R, S) ∈ 𝓔 do
    if word[S] ∈ p_i then
        V_r ← V_r ∪ v_s; v_s ∈ V_s
        E_r ← E_r ∪ (v_r, root[S]); v_r ∈ V_r
        𝓥 ← 𝓥 − S
        𝓔 ← 𝓔 + (R, l); ∀(S, l) ∈ 𝓔
        𝓔 ← 𝓔 − (R, S)
    end if
end for
for each (R, S) ∈ 𝓔 do
    PhraseDPTree(S,P)
end for
return (𝓥, 𝓔)
```

The output of the algorithm is still a tree, for we only cut edges which are compacted into a phrase, the connectivity is keeped. Note that there will be inevitable disagrees between shallow parser and lexical dependency parser, the algorithm implies that we simply follow the result of the latter one: the phrases from shallow parser will not appear in the final result if they cannot be found in the procedure.

Consider the following example:

"*We really enjoyed using the Canon PowerShot SD500.*"

Fig.2 shows the procedure of phrase dependency parsing. Fig.2(a) is the result of the lexical dependency parser. Shallow parsers result is shown in Fig.2(b). Chunk phrases "NP(We)", "VP(really enjoyed using)" and "NP(the Canon PowerShot SD500)" are nodes in the output phrase dependency tree. When visiting node "enjoyed" in Fig.2(a), the shallow parser tells that "really" and "using" which are children of "enjoy" are in the same phrase with their parent, then the three nodes are packed. The final phrase dependency parsing tree is shown in the Fig. 2(c).

2.2 Candidate Product Features and Opinion Expressions Extraction

In this work, we define that *product features* are products, product parts, properties of products, properties of parts, company names and related objects. For example,in consumer electronic domain, "Canon PowerShot", "image quality","camera", "laptop" are all product features.

From analyzing the labeled corpus, we observe that more than 98% of product features are in a single phrase, which is either noun phrase (NP) or verb phrase (VP). Based on it, all NPs and VPs are selected as candidate product features. While prepositional phrases (PPs) and adjectival phrases (ADJPs) are excluded. Although it can cover nearly all the true product features, the precision is relatively low. The large amount of noise candidates may confuse the relation extraction classifier. To shrink the size of candidate set, we introduce language model by an intuition that the more likely a phrase to be a product feature, the more closely it related to the product review. In practice, for a certain domain of product reviews, a language model is build on easily acquired unlabeled data. Each candidate NP or VP chunk in the output of shallow parser is scored by the model, and cut off if its score is less than a threshold.

Opinion expressions are spans of text that express a comment or attitude of the opinion holder, which are usually evaluative or subjective phrases. We also analyze the labeled corpus for opinion expressions and observe that many opinion expressions are used in multiple domains, which is identical with the conclusion presented by Kobayashi et al. (2007). They collected 5,550 opinion expressions from various sources . The coverage of the dictionary is high in multiple domains. Motivated by those observations, we use a dictionary which contains 8221 opinion expressions to select candidates (Wilson et al., 2005b). An assumption we use to filter candidate opinion expressions is that opinion expressions tend to appear closely with product features, which is also used to extract product features by Hu and Liu (2004). In our experiments, the tree distance between product feature and opinion expression in a relation should be less than 5 in the phrase dependency parsing tree.

2.3 Relation Extraction

This section describes our method on extracting relations between opinion expressions and product features using phrase dependency tree. Manually built patterns were used in previous works which have an obvious drawback that those patterns can hardly cover all possible situations. By taking advantage of the kernel methods which can search a feature space much larger than that could be represented by a feature extraction-based approach, we define a new tree kernel over phrase dependency trees and incorporate this kernel within an SVM to extract relations between opinion expressions and product features.

The potential relation set consists of the all combinations between candidate product features and candidate opinion expressions in a sentence. Given a phrase dependency parsing tree, we choose the subtree rooted at the lowest common parent(LCP) of opinion expression and product feature to represent the relation.

Dependency tree kernels has been proposed by (Culotta and Sorensen, 2004). Their kernel is defined on lexical dependency tree by the convolution of similarities between all possible subtrees. However, if the convolution containing too many irrelevant subtrees, over-fitting may occur and decreases the performance of the classifier. In phrase dependency tree, local words in a same phrase are compacted, therefore it provides a way to treat "local dependencies" and "global dependencies" differently (Fig. 3). As a consequence, these two kinds of dependencies will not disturb each other in measuring similarity. Later experiments prove the validity of this statement.

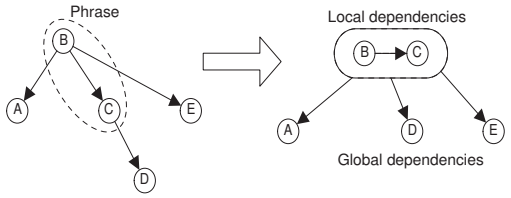

Figure 3: Example of "local dependencies" and "global dependencies".

We generalize the definition by (Culotta and Sorensen, 2004) to fit the phrase dependency tree. Use the symbols in Section 2.1.2, $\mathscr{T}^i$ and $\mathscr{T}^j$ are two trees with root R^i and R^j, $K(\mathscr{T}^i, \mathscr{T}^j)$ is the kernel function for them. Firstly, each tree node $T_k \in \mathscr{T}^i$ is augmented with a set of features F, and an instance of F for T_k is $F^k = \{f^k\}$. A match function $m(T_i, T_j)$ is defined on comparing a subset of nodes' features $M \subseteq F$. And in the same way, a similarity function $s(T_i, T_j)$ are de-

fined on $S \subseteq F$

$$m(T_i, T_j) = \begin{cases} 1 & if\ f_m^i = f_m^j \quad \forall f_m \in M \\ 0 & otherwise \end{cases} \quad (1)$$

and

$$s(T_i, T_j) = \sum_{f_s \in S} C(f_s^i, f_s^j) \quad (2)$$

where

$$C(f_s^i, f_s^j) = \begin{cases} 1 & if\ f_s^i = f_s^j \\ 0 & otherwise \end{cases} \quad (3)$$

For the given phrase dependency parsing trees, the kernel function $K(\mathscr{T}^i, \mathscr{T}^j)$ is defined as follow:

$$K(\mathscr{T}^i, \mathscr{T}^j) = \begin{cases} 0 & if\ m(R^i, R^j) = 0 \\ s(R^i, R^j) + K_{in}(R^i, R^j) & \\ + K_c(R^i.\mathbf{C}, R^j.\mathbf{C}) & otherwise \end{cases} \quad (4)$$

where $K_{in}(R^i, R^j)$ is a kernel function over $R^i = (V_r^i, E_r^i)$ and $R^j = (V_r^j, E_r^j)$'s internal phrase structures,

$$K_{in}(R^i, R^j) = K(R^i, R^j) \quad (5)$$

K_c is the kernel function over R^i and R^j's children. Denote **a** is a continuous subsequence of indices $a, a+1, \cdots a+l(\mathbf{a})$ for R^i's children where $l(\mathbf{a})$ is its length, $\mathbf{a}_s$ is the s-th element in **a**. And likewise **b** for R^j.

$$K_c(R^i.\mathbf{C}, R^j.\mathbf{C}) = \sum_{\mathbf{a},\mathbf{b},l(\mathbf{a})=l(\mathbf{b})} \lambda^{l(\mathbf{a})} K(R^i.[\mathbf{a}], R^j.[\mathbf{b}]) \times \prod_{s=1..l(\mathbf{a})} m(R^i.[\mathbf{a}_s], R^j.[\mathbf{b}_s]) \quad (6)$$

where the constant $0 < \lambda < 1$ normalizes the effects of children subsequences' length.

Compared with the definitions in (Culotta and Sorensen, 2004), we add term K_{in} to handle the internal nodes of a pharse, and make this extension still satisfy the kernel function requirements (composition of kernels is still a kernel (Joachims et al., 2001)). The consideration is that the local words should have limited effects on whole tree structures. So the kernel is defined on external children (K_c) and internal nodes (K_{in}) separately, as the result, the local words are not involved in subsequences of external children for K_c. After the kernel computing through training instances, support vector machine (SVM) is used for classification.

Table 1: Statistics for the annotated corpus

Category	# Products	# Sentences
Cell Phone	2	1100
Diaper	1	375
Digital Camera	4	1470
DVD Player	1	740
MP3 Player	3	3258

3 Experiments and Results

In this section, we describe the annotated corpus and experiment configurations including baseline methods and our results on in-domain and cross-domain.

3.1 Corpus

We conducted experiments with labeled corpus which are selected from Hu and Liu (2004), Jindal and Liu (2008) have built. Their documents are collected from Amazon.com and CNet.com, where products have a large number of reviews. They also manually labeled product features and polarity orientations. Our corpus is selected from them, which contains customer reviews of 11 products belong to 5 categories(Diaper, Cell Phone, Digital Camera, DVD Player, and MP3 Player). Table 1 gives the detail statistics.

Since we need to evaluate not only the product features but also the opinion expressions and relations between them, we asked two annotators to annotate them independently. The annotators started from identifying product features. Then for each product feature, they annotated the opinion expression which has relation with it. Finally, one annotator A_1 extracted 3595 relations, while the other annotator A_2 extracted 3745 relations, and 3217 cases of them matched. In order to measure the annotation quality, we use the following metric to measure the inter-annotator agreement, which is also used by Wiebe et al. (2005).

$$agr(a||b) = \frac{|A\ matches\ B|}{|A|}$$

Table 2: Results for extracting product features and opinion expressions

	P	R	F
Product Feature	42.8%	85.5%	57.0%
Opinion Expression	52.5%	75.2%	61.8%

Table 3: Features used in SVM-1: o denotes an opinion expression and t a product feature

1) Positions of o/t in sentence(start, end, other);
2) The distance between o and t (1, 2, 3, 4, other);
3) Whether o and t have direct dependency relation;
4) Whether o precedes t;
5) POS-Tags of o/t.

Table 4: Features used in SVM-PTree

Features for match function
1) The syntactic category of the tree node (e.g. NP, VP, PP, ADJP).
2) Whether it is an opinion expression node
3) Whether it is a product future node.
Features for similarity function
1) The syntactic category of the tree node (e.g. NP, VP, PP, ADJP).
2) POS-Tag of the head word of node's internal phrases.
3) The type of phrase dependency edge linking to node's parent.
4) Feature 2) for the node's parent
5) Feature 3) for the node's parent

where $agr(a||b)$ represents the inter-annotator agreement between annotator a and b, A and B are the sets of anchors annotated by annotators a and b. $agr(A_1||A_2)$ was 85.9% and $agr(A_2||A_1)$ was 89.5%. It indicates that the reliability of our annotated corpus is satisfactory.

3.2 Preprocessing Results

Results of extracting product features and opinion expressions are shown in Table 2. We use precision, recall and F-measure to evaluate performances. The candidate product features are extracted by the method described in Section 2.2, whose result is in the first row. 6760 of 24414 candidate product features remained after the filtering, which means we cut 72% of irrelevant candidates with a cost of 14.5%(1-85.5%) loss in true answers. Similar to the product feature extraction, the precision of extracting opinion expression is relatively low, while the recall is 75.2%. Since both product features and opinion expressions extractions are preprocessing steps, recall is more important.

3.3 Relation Extraction Experiments

3.3.1 Experiments Settings

In order to compare with state-of-the-art results, we also evaluated the following methods.

1. ***Adjacent*** method extracts relations between a product feature and its nearest opinion expression, which is also used in (Hu and Liu, 2004).

2. ***SVM-1***. To compare with tree kernel based approaches, we evaluated an SVM[1] result with a set of manually selected features(Table 3), which are also used in (Kobayashi et al., 2007).

3. ***SVM-2*** is designed to compare the effectiveness of cross-domain performances. The features used are simple bag of words and POS-Tags between opinion expressions and product features.

4. ***SVM-WTree*** uses head words of opinion expressions and product features in the word-level dependency tree, as the previous works in information extraction. Then conducts tree kernel proposed by Culotta and Sorensen (2004).

5. ***SVM-PTree*** denotes the results of our tree-kernel based SVM, which is described in the Section 2.3. Stanford parser (Klein and Manning, 2002) and Sundance (Riloff and Phillips, 2004) are used as lexical dependency parser and shallow parser. The features in match function and similarity function are shown in Table 4.

6. ***OERight*** is the result of SVM-PTree with correct opinion expressions.

7. ***PFRight*** is the result of SVM-PTree with correct product features.

Table 5 shows the performances of different relation extraction methods with in-domain data. For each domain, we conducted 5-fold cross validation. Table 6 shows the performances of the extraction methods on cross-domain data. We use the digital camera and cell phone domain as training set. The other domains are used as testing set.

[1]libsvm 2.88 is used in our experiments

Table 5: Results of different methods

	Cell Phone			MP3 Player			Digital Camera			DVD Player			Diaper		
Methods	P	R	F	P	R	F	P	R	F	P	R	F	P	R	F
Adjacent	40.3%	60.5%	48.4%	26.5%	59.3%	36.7%	32.7%	59.1%	42.1%	31.8%	68.4%	**43.4%**	23.4%	78.8%	**36.1%**
SVM-1	69.5%	42.3%	52.6%	60.7%	30.6%	40.7%	61.4%	32.4%	42.4%	56.0%	27.6%	37.0%	29.3%	14.1%	19.0%
SVM-2	60.7%	19.7%	29.7%	63.6%	23.8%	34.6%	66.9%	23.3%	34.6%	66.7%	13.2%	22.0%	79.2%	22.4%	34.9%
SVM-WTree	52.6%	52.7%	52.6%	46.4%	43.8%	45.1%	49.1%	46.0%	47.5%	35.9%	32.0%	33.8%	36.6%	31.7%	34.0%
SVM-PTree	55.6%	57.2%	**56.4%**	51.7%	50.7%	**51.2%**	54.0%	49.9%	**51.9%**	37.1%	35.4%	36.2%	37.3%	30.5%	33.6%
OERight	66.7%	69.5%	68.1%	65.6%	65.9%	65.7%	64.3%	61.0%	62.6%	59.9%	63.9%	61.8%	55.8%	58.5%	57.1%
PFRight	62.8%	62.1%	62.4%	61.3%	56.8%	59.0%	59.7%	56.2%	57.9%	46.9%	46.6%	46.7%	58.5%	51.3%	53.4%

Table 6: Results for total performance with cross domain training data

	Diaper			DVD Player			MP3 Player		
Methods	P	R	F	P	R	F	P	R	F
Adjacent	23.4%	78.8%	36.1%	31.8%	68.4%	43.4%	26.5%	59.3%	36.7%
SVM-1	22.4%	30.6%	25.9%	52.8%	30.9%	39.0%	55.9%	36.8%	44.4%
SVM-2	71.9%	15.1%	25.0%	51.2%	13.2%	21.0%	63.1%	22.0%	32.6%
SVM-WTree	38.7%	52.4%	**44.5%**	30.7%	59.2%	40.4%	38.1%	47.2%	42.2%
SVM-PTree	37.3%	53.7%	44.0%	59.2%	48.3%	**46.3%**	43.0%	48.9%	**45.8%**

3.3.2 Results Discussion

Table 5 presents different methods' results in five domains. We observe that the three learning based methods(SVM-1, SVM-WTree, SVM-PTree) perform better than the Adjacent baseline in the first three domains. However, in other domains, directly adjacent method is better than the learning based methods. The main difference between the first three domains and the last two domains is the size of data(Table 1). It implies that the simple Adjacent method is also competent when the training set is small.

A further inspection into the result of first 3 domains, we can also conclude that: 1) Tree kernels(SVM-WTree and SVM-PTree) are better than Adjacent, SVM-1 and SVM-2 in all domains. It proofs that the dependency tree is important in the opinion relation extraction. The reason for that is a connection between an opinion and its target can be discovered with various syntactic structures. 2) The kernel defined on phrase dependency tree (SVM-PTree) outperforms kernel defined on word level dependency tree(SVM-WTree) by 4.8% in average. We believe the main reason is that phrase dependency tree provides a more succinct tree structure, and the separative treatment of local dependencies and global dependencies in kernel computation can indeed improve the performance of relation extraction.

To analysis the results of preprocessing steps' influences on the following relation extraction, we provide 2 additional experiments which the product features and opinion expressions are all correctly extracted respectively: OERight and PFRight. These two results show that given an exactly extraction of opinion expression and product feature, the results of opinion relation extraction will be much better. Further, opinion expressions are more influential which naturally means the opinion expressions are crucial in opinion relation extraction.

For evaluations on cross domain, the Adjacent method doesn't need training data, its results are the same as the in-domain experiments. Note in Table 3 and Table 4, we don't use domain related features in SVM-1, SVM-WTree, SVM-PTree, but SVM-2's features are domain dependent. Since the cross-domain training set is larger than the original one in Diaper and DVD domain, the models are trained more sufficiently. The final results on cross-domain are even better than in-domain experiments on SVM-1, SVM-WTree, and SVM-PTree with percentage of 4.6%, 8.6%, 10.3% in average. And the cross-domain training set is smaller than in-domain in MP3, but it also achieve competitive performance with the

in-domain. On the other hand, SVM-2's result decreased compared with the in-domain experiments because the test domain changed. At the same time, SVM-PTree outperforms other methods which is similar in in-domain experiments.

4 Related Work

Opinion mining has recently received considerable attention. Amount of works have been done on sentimental classification in different levels (Zhang et al., 2009; Somasundaran et al., 2008; Pang et al., 2002; Dave et al., 2003; Kim and Hovy, 2004; Takamura et al., 2005). While we focus on extracting product features, opinion expressions and mining relations in this paper.

Kobayashi et al. (2007) presented their work on extracting opinion units including: opinion holder, subject, aspect and evaluation. Subject and aspect belong to product features, while evaluation is the opinion expression in our work. They converted the task to two kinds of relation extraction tasks and proposed a machine learning-based method which combines contextual clues and statistical clues. Their experimental results showed that the model using contextual clues improved the performance. However since the contextual information in a domain is specific, the model got by their approach can not easily converted to other domains.

Choi et al. (2006) used an integer linear programming approach to jointly extract entities and relations in the context of opinion oriented information extraction. They identified expressions of opinions, sources of opinions and the linking relation that exists between them. The sources of opinions denote to the person or entity that holds the opinion.

Another area related to our work is opinion expressions identification (Wilson et al., 2005a; Breck et al., 2007). They worked on identifying the words and phrases that express opinions in text. According to Wiebe et al. (2005), there are two types of opinion expressions, direct subjective expressions and expressive subjective elements.

5 Conclusions

In this paper, we described our work on mining opinions from unstructured documents. We focused on extracting relations between product features and opinion expressions. The novelties of our work included: 1) we defined the phrase dependency parsing and proposed an approach to construct the phrase dependency trees; 2) we proposed a new tree kernel function to model the phrase dependency trees. Experimental results show that our approach improved the performances of the mining task.

6 Acknowledgement

This work was (partially) funded by Chinese NSF 60673038, Doctoral Fund of Ministry of Education of China 200802460066, and Shanghai Science and Technology Development Funds 08511500302. The authors would like to thank the reviewers for their useful comments.

References

Eric Breck, Yejin Choi, and Claire Cardie. 2007. Identifying expressions of opinion in context. In *Proceedings of IJCAI-2007*.

Yejin Choi, Claire Cardie, Ellen Riloff, and Siddharth Patwardhan. 2005. Identifying sources of opinions with conditional random fields and extraction patterns. In *Proceedings of HLT/EMNLP*.

Yejin Choi, Eric Breck, and Claire Cardie. 2006. Joint extraction of entities and relations for opinion recognition. In *Proceedings EMNLP*.

Aron Culotta and Jeffrey Sorensen. 2004. Dependency tree kernels for relation extraction. In *In Proceedings of ACL 2004*.

Kushal Dave, Steve Lawrence, and David M. Pennock. 2003. Mining the peanut gallery: opinion extraction and semantic classification of product reviews. In *Proceedings of WWW 2003*.

Minqing Hu and Bing Liu. 2004. Mining and summarizing customer reviews. In *Proceedings of the ACM SIGKDD 2004*.

Nitin Jindal and Bing Liu. 2008. Opinion spam and analysis. In *Proceedings of WSDM '08*.

Thorsten Joachims, Nello Cristianini, and John Shawe-Taylor. 2001. Composite kernels for hypertext categorisation. In *Proceedings of ICML '01*.

Soo-Min Kim and Eduard Hovy. 2004. Determining the sentiment of opinions. In *Proceedings of Coling 2004*. COLING.

Dan Klein and Christopher D. Manning. 2002. Fast exact inference with a factored model for natural language parsing. In *In Advances in Neural Information Processing Systems*.

Nozomi Kobayashi, Kentaro Inui, and Yuji Matsumoto. 2007. Extracting aspect-evaluation and aspect-of relations in opinion mining. In *Proceedings of EMNLP-CoNLL 2007*.

Bo Pang, Lillian Lee, and Shivakumar Vaithyanathan. 2002. Thumbs up? Sentiment classification using machine learning techniques. In *Proc. of EMNLP 2002*.

Ana-Maria Popescu and Oren Etzioni. 2005. Extracting product features and opinions from reviews. In *Proceedings of HLT/EMNLP*.

E. Riloff and W. Phillips. 2004. An introduction to the sundance and autoslog systems. In *University of Utah School of Computing Technical Report UUCS-04-015*.

Beatrice Santorini and Anthony Kroch. 2007. *The syntax of natural language: An online introduction using the Trees program*. http://www.ling.upenn.edu/ beatrice/syntax-textbook.

Swapna Somasundaran, Janyce Wiebe, and Josef Ruppenhofer. 2008. Discourse level opinion interpretation. In *Proceedings of COLING 2008*.

Hiroya Takamura, Takashi Inui, and Manabu Okumura. 2005. Extracting semantic orientations of words using spin model. In *Proceedings of ACL'05*.

L. Tesnière. 1959. *Eléments de syntaxe structurale*. Editions Klincksieck.

Janyce Wiebe, Theresa Wilson, and Claire Cardie. 2005. Annotating expressions of opinions and emotions in language. *Language Resources and Evaluation*, 39(2/3).

Theresa Wilson, Paul Hoffmann, Swapna Somasundaran, Jason Kessler, Janyce Wiebe, Yejin Choi, Claire Cardie, Ellen Riloff, and Siddharth Patwardhan. 2005a. Opinionfinder: A system for subjectivity analysis. In *Demonstration Description in Conference on Empirical Methods in Natural Language Processing*.

Theresa Wilson, Janyce Wiebe, and Paul Hoffmann. 2005b. Recognizing contextual polarity in phrase-level sentiment analysis. In *Proceedings of HLT-EMNLP*.

Fei Xia and Martha Palmer. 2001. Converting dependency structures to phrase structures. In *HLT '01: Proceedings of the first international conference on Human language technology research*.

Qi Zhang, Yuanbin Wu, Tao Li, Mitsunori Ogihara, Joseph Johnson, and Xuanjing Huang. 2009. Mining product reviews based on shallow dependency parsing. In *Proceedings of SIGIR 2009*.