

# Clustering Product Features for Opinion Mining

Zhongwu Zhai<sup>†</sup> Bing Liu<sup>‡</sup> Hua Xu<sup>†</sup> Peifa Jia<sup>†</sup>

<sup>†</sup>State Key Lab of Intelligent Tech. & Sys., Tsinghua National Lab for Info. Sci. and Tech., Dept. of Comp. Sci. & Tech., Tsinghua University  
zhaizhongwu@gmail.com

<sup>‡</sup>Dept. of Comp. Sci., University of Illinois at Chicago  
liub@cs.uic.edu

## ABSTRACT

In sentiment analysis of product reviews, one important problem is to produce a summary of opinions based on product features/attributes (also called aspects). However, for the same feature, people can express it with many different words or phrases. To produce a useful summary, these words and phrases, which are domain synonyms, need to be grouped under the same feature group. Although several methods have been proposed to extract product features from reviews, limited work has been done on clustering or grouping of synonym features. This paper focuses on this task. Classic methods for solving this problem are based on unsupervised learning using some forms of distributional similarity. However, we found that these methods do not do well. We then model it as a semi-supervised learning problem. Lexical characteristics of the problem are exploited to automatically identify some labeled examples. Empirical evaluation shows that the proposed method outperforms existing state-of-the-art methods by a large margin.

## Categories and Subject Descriptors

H.3.3 [Information Storage and Retrieval]: Information Search and Retrieval – *Clustering*. I.2.7 [Artificial Intelligence]: Natural Language Processing – *Text analysis*.

## General Terms

Algorithms, Experimentation

## Keywords

Opinion Mining, Product Feature Grouping

## 1. INTRODUCTION

One form of opinion mining in product reviews is to produce a feature-based summary [14, 31]. In this model, product features are first identified, and positive and negative opinions on them are aggregated to produce a summary on the features. Features of a product are attributes, components and other aspects of the product, e.g., “picture quality”, “battery life” and “zoom” of a digital camera.

In reviews (or any writings), people often use different words and phrases to describe the same product feature. We call the actual

words or phrases that express the same feature, feature expressions. For example, “picture” and “photo” are feature expressions referring to the same feature of cameras. In this paper, we assume that all feature expressions have been identified by an existing algorithm. There are many such algorithms [17-20, 29, 35, 38].

Grouping feature expressions, which are domain synonyms, is critical for effective opinion summary [26]. Since there are typically hundreds of feature expressions that can be discovered from text for an opinion mining application, it’s very time-consuming and tedious for human users to group them into feature categories. Some automated assistance is needed. Unsupervised learning or clustering is the natural technique for solving the problem. The similarity measures used in clustering are usually based on some form of distributional similarity [6, 10, 22, 24, 32, 34, 37]. Recent work also used topic modeling [12, 40]. However, we show that these methods do not perform well. Even the latest topic modeling method that consider pre-existing knowledge [3] does not do well.

Obviously, thesaurus dictionaries can be helpful for finding synonyms [9, 26], but they are far from sufficient due to a few reasons. First, many words and phrases that are not synonyms in a dictionary may refer to the same feature in an application domain. For example, “appearance” and “design” are not synonymous, but they can indicate the same feature, design. Second, many synonyms are domain dependent. For example, “movie” and “picture” are synonyms in movie reviews, but they are not synonyms in camera reviews as “picture” is more likely to be synonymous to “photo” while “movie” to “video”.

Due to the poor performance of the unsupervised methods, we formulate the problem as a supervised learning problem, or more precisely a semi-supervised learning problem but without asking the user to manually label any training examples. This new formulation produces much better results as we will see in Section 5. However, for semi-supervised learning, a small set of labeled examples and a set of unlabeled examples are required. The problem then is how to partition the feature expressions into a labeled set and an unlabeled set automatically. We exploit two pieces of natural language knowledge to achieve this:

- Sharing words: Feature expressions sharing some common words are likely to belong to the same group, e.g., “battery life”, “battery”, and “battery power”.
- Lexical similarity: Feature expressions that are similar lexically based on WordNet [11, 16, 23, 33, 36] are likely to belong to the same group, e.g., “movie” and “picture”. Note that synonyms are covered by lexical similarity as they will have very high lexical similarity.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

WSDM’11, February 9–12, 2011, Hong Kong, China.

Copyright 2011 ACM 978-1-4503-0493-1/11/02...\$10.00.

We call these two pieces of knowledge *soft constraints* as they constrain certain feature expressions to be in the same feature group to some extent. They are called soft (rather than hard) constraints because they can be relaxed in the learning process. This relaxation is important because the above two constraints can result in wrong groupings. The semi-supervised learning method is allowed to re-assign them to other groups in the learning process. Note that sharing words can also be handled by lexical similarity, but we do not use lexical similarity due to two reasons. First, some words may not exist in WordNet. Second, sharing words is much easier to compute and there is no need to refer to WordNet.

Clearly, lexical similarity can also be used in clustering. However, we again found it quite inaccurate because only those words or phrases with very high similarities are reliable. We thus propose to combine it with semi-supervised learning, i.e., using limited lexical similarity to find some labeled examples for semi-supervised learning. This combination produces much better results.

For semi-supervised learning, we use the EM algorithm formulated in [30], which is based on naïve Bayesian classification. Although other supervised methods can also be applied, the EM algorithm based on naïve Bayesian is probably the most efficient method and it already performs much better than the state-of-the-art current methods as we will see later. In the future work, we will experiment with other semi-supervised learning methods. Instead of using the algorithm in [30] directly, we need to augment it to allow EM to re-assign classes of the labeled set in order to correct some errors in the automated labeling process. Note that, for our problem, each input document for the augmented EM is actually the surrounding words of each feature expression  $v_i$ . Here, we further exploit another piece of natural language knowledge to help our algorithm to extract more discriminative distributional context:

- Positive and negative correlation: Feature expressions that ever co-occur in the same sentence are unlikely to belong to the same group, e.g., “*I like the picture quality, the battery life, and zoom of this camera*” and “*The picture quality is great, the battery life is also long, but the zoom is not good*”. From either of the sentences, we can infer that “picture quality”, “battery life”, and “zoom” are probably not synonyms because people are unlikely to repeat the same thing in the same sentence.

Based on this knowledge, when extracting the surrounding words context  $D_i$  for each feature expression  $v_i$ , any feature expression  $v_k$ , that co-occur with  $v_i$  in review sentences, are removed from  $D_i$ . Thus,  $D_i$  and  $D_k$  will not compete with each other in the learning process, which results in better performances.

Our evaluation was conducted using a large number of reviews from 5 different domains. The results show that the proposed method outperforms different variations of topic modeling, clustering methods based on distributional similarity and lexical similarity, and also the recent unsupervised feature grouping method *mLSA*.

In summary, this paper makes the following contributions:

1. The problem solved in this paper is an unsupervised task. However, due to the poor performances of existing unsupervised methods we formulate it as a semi-supervised

learning task but without asking the user to label any training examples. To our knowledge, this is first such attempt. An EM algorithm based on naïve Bayesian classification is adapted to solve the problem, which allows EM to re-assign classes of the labeled examples to different classes.

2. Since there are no labeled examples for learning, we propose to use two soft constraints to help label some examples and one piece of pre-existing natural language knowledge to extract more discriminative distributional context for the augmented EM.
3. It is shown experimentally that the new method outperforms the main existing state-of-the-art methods based on clustering and other techniques that can be applied to the task.

## 2. RELATED WORK

The key to clustering is the similarity measure. There are two main kinds of similarity measures for our task [1]: those relying on *pre-existing knowledge resources* (e.g., thesauri, and semantic networks) [2, 15, 42], and those relying on *distributional properties* of words in corpora [6, 10, 22, 32, 34, 37].

In the category that relies on pre-existing knowledge sources, the work of Carenini *et al.* [9] is most related to ours. The authors proposed a method to map discovered feature expressions to a given domain product feature taxonomy, using several word similarity metrics. This kind of similarities is often called *lexical similarity*. A related approach based on WordNet was also used by Liu *et al.* [26]. Both these works do not use the word distribution information, which is their main weakness because many expressions of the same feature are not synonyms or even similar in WordNet because they are domain dependent. Dictionaries do not contain domain specific knowledge, for which a domain corpus is needed.

Distributional similarity is based on the hypothesis that words with similar meaning tend to appear in similar contexts [13]. As such, it fetches the surrounding words as context for each term (e.g., feature expression). Similarity measures such as *Cosine*, *Jaccard*, *Dice*, etc [21], can then be employed to compute the similarities between words or phrases. [8, 24, 28] also calculate PMI (Pointwise Mutual Information) similarity between words, and [28] groups words using a graph-based algorithm based on PMI or Chi-squared ( $\chi^2$ ) test. [24] clusters words with the cosine similarity based on PMI weighting. We will show that these methods do not perform well in the experiment section. In [43], a semi-supervised learning method is used. However, it requires the user to provide labeled examples, whereas this study does not need any pre-labeled examples. It thus solves a different problem. There are also two detailed differences: First, this proposed algorithm is further enhanced by lexical (or WordNet) similarity (see Section 3.1). Second, another piece of pre-existing natural language knowledge is used to extract more discriminative context documents for the proposed algorithm (see Section 4).

Recent work also applied topic modeling (e.g., Latent Dirichlet Allocation (LDA)) to solve the problem based on a domain corpus. Branavan *et al.*[7] extracted and clustered semantic properties of *reviews* based on pros/cons annotations using modified LDA, which performs a different task from ours. Guo *et al.* [12] proposed a multilevel latent semantic association technique (called *mLSA*) to group product feature expressions. At the first level, all the words in product feature expressions are

grouped into a set of concepts using LDA. The results are used to build latent topic structures for product feature expressions. At the second level, feature expressions are grouped by LDA again according to their latent topic structures produced from level 1 and context snippets in reviews.

Recently, [3] reported a new formulation of LDA, which is able to consider pre-existing knowledge in the form of must-link and cannot-link constraints borrowed from constrained clustering [41]. The new method is called *DF-LDA*. Must-links state that some data points must be in the same cluster, and cannot-links state that some data points cannot be in the same cluster. Our soft constraints are similar to must-links. However, *DF-LDA* does not perform as well as the proposed algorithm, which is mainly based on semi-supervised learning.

### 3. THE PROPOSED ALGORITHM

Since our original problem is unsupervised, we assume that the user will specify the number of clusters  $k$ , which is also the number of classes used in semi-supervised learning. The input to the proposed algorithm consists of: a set of reviews  $R$ , and a set of discovered feature expressions  $F$  from  $R$ . Then the proposed algorithm assigns the discovered features  $F$  to  $k$  groups.

Since unsupervised methods do not perform well, we re-formulate the problem a semi-supervised learning problem. However, semi-supervised learning needs some labeled examples. Since no labeled data exist, the proposed algorithm has to first automatically label some such data. After that, a semi-supervised learning method can be applied. In this work, we adapted the naïve Bayesian based EM formulation in [30] for our purpose, which will be discussed in Section 3.2.

In the following, we use a graph to introduce the step for finding labeled examples (feature expressions). The given feature expressions  $\{v_1, v_2, \dots, v_n\}$  are vertices in the graph, and different constraints or similarities are considered as edges. Some connected components will be used as the labeled data.

#### 3.1 Generating Labeled Data L

The labeled set  $L$  is generated using two types of pre-existing knowledge/constraints in three steps.

**Step 1 (connect feature expressions using sharing words):** For the task of opinion mining, many feature expressions are phrases consisting of multiple words, e.g., “customer service”, “customer support”, “service”. Sharing words is an important clue (pre-existing knowledge) that can be exploited for our purpose, i.e., feature expressions sharing some words are likely to belong to the same group or cluster. Since this constraint can be violated in some occasions, we call this piece of existing knowledge a soft-constraint  $SC$ .

Then, we add all such constrained pairs into the graph  $G$  as a set of edges, which gives us a new graph  $G_{sc}$ . That is to say, all sharing-words feature expressions are connected in  $G_{sc}$ . Stopwords are removed before sharing of words among feature expressions is computed.

Although the sharing-word constraint is useful, the number of such pairs is small and does not help us that much. For example, in our five experimental real-world data sets, their corresponding graphs  $G_{sc}$  are very sparse, which means that there are too many disconnected components or sub-graphs. Figure 1 shows the graph

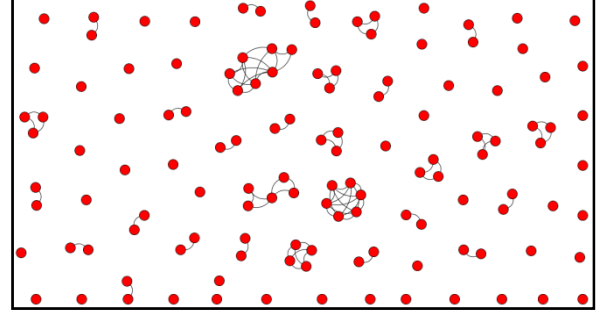


Figure 1. The  $G_{sc}$  of the Insurance data set

---

**Input:** components of  $G_{sc}$   $\{c_1, c_2, \dots, c_n\}$ ;  
number of merges  $K$ ;  
**Output:** merged components  $\{C_1, C_2, \dots, C_p\}$

- 1 //Calculate pairwise similarities of  $\{c_1, c_2, \dots, c_n\}$
- 2 **for**  $c_i$  **in**  $\{c_1, c_2, \dots, c_n\}$ :
- 3     **for**  $c_j$  **in**  $\{c_{i+1}, c_{i+2}, \dots, c_n\}$ :
- 4          $sim(c_i, c_j) = \text{Avg}_{v_r \in c_i, v_t \in c_j} (\text{PhraseSim}(v_r, v_t))$
- 5 **Sort**  $\text{pair}(i, j)$  as *SortedPairs* **by**  $sim(c_i, c_j)$  in descending order, where  $i \neq j, i \in \{1, 2, \dots, n\}, j \in \{1, 2, \dots, n\}$
- 6 **for**  $\text{pair}(i, j)$  **in**  $\{\text{top } K \text{ SortedPairs}\}$ :
- 7     **merge**  $c_i$  and  $c_j$  in  $G_{sc}$
- 8 Output components in  $G_{sc}$  as  $\{C_1, C_2, \dots, C_p\}$
- 9
- 10 //Subfunction for calculating similarity between phrases
- 11 **PhraseSim** ( $prs1, prs2$ ):
- 12     **return**  $\text{Max}_{w_k \in prs1, w_q \in prs2} Jcn(w_k, w_q)$

---

Figure 2. Merging components by lexical similarity

of one of our experimental data sets. It is difficult to decide which connected components should be used as the labeled examples. We need another piece of existing knowledge, lexical similarity to merge some components of  $G_{sc}$ .

**Step 2 (merge components using lexical similarity):** Lexical similarity based on WordNet is widely-used in the NLP area to measure the similarity of two words [11, 16, 23, 33]. It is another piece of knowledge that can be utilized for our grouping task. For example, “picture” and “image” has very high similarity in WordNet.

Note that although we say that many domain synonyms may be quite dissimilar in WordNet, those words with high similarities in WordNet are likely to be synonyms in most (if not all) domains. It is also important to note that here we do not use a thesaurus to find synonyms because the WordNet lexical similarity can quantify the similarity strength, and thus subsumes the synonym look-up. This step uses such similarities to merge similar components in  $G_{sc}$ . The algorithm is given in Figure 2.

First, we calculate pair-wise similarities between the components  $\{c_1, c_2, \dots, c_n\}$  (lines 1 to 4). Take components  $c_i$  and  $c_j$  as examples. Their pair-wise similarity  $sim(c_i, c_j)$  is the average value of all the cross similarities of their members, computed using  $\text{PhraseSim}(v_r, v_t)$ , where  $v_r \in c_i, v_t \in c_j$ .  $\text{PhraseSim}(v_r, v_t)$  is the function for calculating the WordNet similarity between two phrases (lines 11 and 12). In line 12,  $Jcn(w_k, w_q)$  is the algorithm

$$Res(w_1, w_2) = IC(LCS(w_1, w_2)) \quad (1)$$

$$IC(w) = -\log \Pr(w) \quad (2)$$

$$Lin(w_1, w_2) = \frac{2 \times Res(w_1, w_2)}{IC(w_1) + IC(w_2)} \quad (3)$$

$$Jcn(w_1, w_2) = \frac{1}{IC(w_1) + IC(w_2) - 2 \times Res(w_1, w_2)} \quad (4)$$

for calculating the similarity between two words  $w_k$  and  $w_q$  given in [16]. We also tried some other similarity calculation algorithms *Res* [36] and *Lin* [23], but *Jcn* performs the best for our task. These measures all rely on varying degrees of least common subsumer (*LCS*), which is the most specific concept that is a shared ancestor of the two concepts represented by the words [33]. For example, the *LCS* of *automobile* and *scooter* is *vehicle*. *Res* simply uses the information content of *LCS* as the similarity value (Equations 1 and 2).  $\Pr(w)$  is the probability of the concept word  $w$  (based on the observed frequency counts in the WordNet corpus). Both *Lin* and *Jcn* try to refine *Res* by augmenting it with the information content of the individual concepts being measured in two different ways using Equations 3 and 4, respectively.

*Second*, the components are selected to be merged based on ranking according to the pair-wise similarities (lines 5 to 7). Clearly, one can use these pair-wise similarities for hierarchical clustering. However, its results are poorer as we will see in Section 5 (see the results of *CHC* and *SHC*). We believe that the main reason is that only those components with very high similarities are reliably. Thus, we only trust those high similarities. In the algorithm, we only perform the top  $k$  merges (lines 6 and 7), where  $k$  is the number of clusters required by the user. In the experimental section, we will show the effect of the number of merges on the final results.

**Step 3 (select the leader components as labeled data  $L$ ):** After the above two steps, we obtain a new set of components  $\{C_1, C_2, \dots, C_p\}$ . Although the number  $p$  of remaining components is much smaller than the number of original feature expressions,  $p$  is usually still much larger than the required number of clusters  $k$  because the graph  $G_{sc}$  is highly disconnected (see step 1), and only limited number of merges ( $k$ ) are performed.

This step selects  $k$  leader components from the  $p$  components  $\{C_1, C_2, \dots, C_p\}$  to form the labeled data with  $k$  classes or clusters. Since in semi-supervised learning, the number of labeled examples has a major and positive impact on the final result, i.e., the more the better, we thus first rank  $\{C_1, C_2, \dots, C_p\}$  according to their sizes (the number of vertices in each component), and then select the top  $k$  components as the labeled data  $L$ . Naturally, the vertices or feature expressions in the *unselected* components in  $\{C_1, C_2, \dots, C_p\}$  form the unlabeled set  $U$ .

## 3.2 Semi-Supervised Learning using EM

With the labeled ( $L$ ) and unlabeled ( $U$ ) examples, we can run the EM based semi-supervised learning algorithm to assign a class or cluster to each unlabeled feature expression. For learning, each example or feature expression is represented with a document, which consists of the surrounding words of the feature expression (see Section 4). Hence, EM uses the distributional information in

---

**Input:** Labeled examples  $L$   
Unlabeled examples  $U$

- 1 Learn an initial naïve Bayesian classifier  $f_0$  using  $L$  and Equations 5 and 6;
- 2 **repeat**
- 3   // E-Step
- 4   **for** each example  $d_i$  in  $U \cup L$  :
- 5     Using the current classifier  $f_x$  to compute  $P(c_j|d_i)$  using Equation 7.
- 6   **end**
- 7   // M-Step
- 8   Learn a new naïve Bayesian classifier  $f_x$  from  $L$  and  $U$  by computing  $P(w_t|c_j)$  and  $P(c_j)$  using Equations 5 and 6.
- 9 **until** the classifier parameters stabilize

---

**Output:** the classifier  $f_x$  from the last iteration.

---

**Figure 3. The augmented EM algorithm**

$$P(w_t|c_j) = \frac{1 + \sum_{i=1}^{|D|} N_{ti} P(c_j|d_i)}{|V| + \sum_{m=1}^{|V|} \sum_{i=1}^{|D|} N_{mi} P(c_j|d_i)} \quad (5^1)$$

$$P(c_j) = \frac{1 + \sum_{i=1}^{|D|} P(c_j|d_i)}{|C| + |D|} \quad (6^1)$$

$$P(c_j|d_i) = \frac{P(c_j) \prod_{k=1}^{|d_i|} P(w_{d_i,k}|c_j)}{\sum_{r=1}^{|C|} P(c_r) \prod_{k=1}^{|d_i|} P(w_{d_i,k}|c_r)} \quad (7)$$

its classification model building process. However, we should note that here we use only the *distributional information*, not *distributional similarity*, which we will show is less effective. Distributional information is critical for finding domain synonyms because it gives the domain context. Recall in generating the labeled data in Section 3.1, no distributional information was used. Only domain independent information, i.e., sharing of words and lexical similarity, were employed. Thus, our approach is able to exploit both domain independent and domain dependent information.

Due to the way that we generated the labeled data, we need to modify the EM algorithm in [30]. In the original algorithm, the class labels of the labeled examples do not change in the learning process as they are considered correct (manually labeled). However, in our case, the labeled data may not be reliable. Sharing of words does not work in some occasions. For example, “picture quality” and “build quality” share the word “quality”, but they should not be grouped to the same class or cluster. For a similar reason, lexical similarity may result in errors too.

To solve this problem, we modify the EM algorithm in [30] to allow class labels in the labeled set  $L$  to change. That is, EM can reassign classes to the labeled examples. The augmented EM algorithm for our purpose is given in Figure 3. The algorithm uses three equations, i.e., (5), (6) and (7).

First, the algorithm learns a classifier  $f_0$  using only the labeled data  $L$  and Equations 5 and 6 (line 1). Then,  $f_0$  is applied to assign probabilistic labels to both the labeled data  $L$  and unlabeled data  $U$  (lines 4 to 6) using Equation 7 (the Expectation step). Next, a new classifier is learned using the newly probabilistically labeled examples in both  $L$  and  $U$ , again using Equations 5 and 6 (the

---

<sup>1</sup> Laplace smoothing is used to prevent zero probabilities for infrequently occurring words.

---

```

1 for each feature expression  $v_i$  in  $L$  (or  $U$ ) :
2    $S_i \leftarrow$  all sentences containing  $v_i$  in  $R$ 
3   for each sentence  $s_{ij} \in S_i$  :
4     for each word  $v$  in window  $[-t, t]$ : //including the words in  $v_i$ 
5       if  $v$  does not co-occur with  $v_i$  in any review sentences:
6          $d_{ij} \leftarrow v$ 
7    $D_i \leftarrow$  words from all  $d_{ij}, j = 1, 2, \dots, |S_i|$ 
8     //duplicates are kept as it is not union

```

---

**Figure 4. Distributional context extraction**

Maximization step). These last two steps iterate until convergence. When the algorithm ends, each labeled and unlabeled example is assigned a posterior probability of belonging to each class or cluster.

We now explain the notations in the Equations. Given a set of training documents  $D$ , each document  $D_i$  in  $D$  is considered an ordered list of words.  $w_{d_i,k}$  denotes the  $k^{\text{th}}$  word in  $D_i$ , where each word is from the vocabulary  $V = \{w_1, w_2, \dots, w_{|V|}\}$ .  $C = \{c_1, c_2, \dots, c_{|C|}\}$  is the set of pre-defined classes or groups. In our case,  $|C| = k$ .  $N_{d_i}$  is the number of times the word  $w_i$  occurs in document  $d_i$ .

For our problem, the surrounding words contexts of the labeled feature expressions form  $L$ , while the surrounding words of the non-labeled feature expressions form  $U$ . When EM converges, the classification labels of all the feature expressions give us the final grouping. Surrounding words contexts will be discussed in Section 4.

## 4. CONTEXT EXTRACTION

To apply the proposed algorithm, a document  $D_i$  needs to be prepared for each feature expression  $v_i$  for naïve Bayesian learning.  $D_i$  is formed by aggregating the distributional context of every sentence  $s_{ij}$  in our corpus that contains the expression  $v_i$ . The context of a sentence for  $v_i$  is the surrounding words of  $v_i$  in a text window of  $[-t, t]$ , including the words in  $v_i$ . In our experiments,  $t$  is empirically set to 15. Given a relevant corpus  $R$ , the document  $D_i$  for each feature expression  $v_i$  in  $L$  (or  $U$ ) is generated using the algorithm in Figure 4. Stopwords are removed.

For example, two feature expression from  $L$  (or  $U$ ) are  $v_i = \text{“screen”}$  and  $v_k = \text{“picture”}$ , and there are two sentences in our corpus  $R$  that contain “screen” or “picture”.

$s_{i1} = \text{“The LCD screen gives clear picture”}$ .

Here we use the window size of  $[-3, 3]$ . For feature expression  $v_i = \text{“screen”}$ ,  $s_{i1}$  gives us  $d_{i1} = \langle \text{LCD, screen, gives, clear} \rangle$  as a bag of words, “the” is removed as stopwords. For feature expression  $v_k = \text{“picture”}$ ,  $s_{i1}$  gives us  $d_{k1} = \langle \text{gives, clear} \rangle$  as a bag of words. Note that “picture” is removed in  $d_{i1}$  and “screen” is removed in  $d_{k1}$ , because they co-occur in review sentences and they are unlikely to belong to the same topic. Finally, we obtain the document  $D_i$  for feature expression  $v_i$  and  $D_k$  for feature expression  $v_k$ :

$D_i = \langle \text{LCD, screen, gives, clear} \rangle$   
 $D_k = \langle \text{gives, clear} \rangle$

## 5. EMPIRICAL EVALUATION

This section evaluates the proposed algorithm and compares it with the main existing methods that can be applied to solve the

**Table 1. Data sets and gold standards**

	H	I	M	C	V
#Sentences	6355	12446	12107	9731	8785
#Reviews	587	2802	933	1486	551
#Expressions	237	148	333	317	266
#Groups	15	8	15	16	28

problem.

### 5.1 Review Data Sets and Gold Standards

To show the generality of the proposed method, experiments were conducted using reviews from 5 diverse domains: *Hometheater* (H), *Insurance* (I), *Mattress* (M), *Car* (C) and *Vacuum* (V). All the data sets were obtained from a commercial company that provides sentiment analysis services.

All the *gold standard* feature expressions and groups were also obtained from the company, which were annotated by their customers, and have been used to produce sentiment analysis reports based on features. The details of the data sets and the gold standards are given in Table 1.

### 5.2 Evaluation Measures

Since the problem of grouping feature expressions is a clustering task, two common measures for evaluating clustering are used in this study, *Entropy* and *Purity* [25]. Below, we briefly describe entropy and purity. Given a data set  $DS$ , its gold partition is  $G = \{g_1, \dots, g_j, \dots, g_k\}$ , where  $k$  is the given number of clusters. The groups partition  $DS$  into  $k$  disjoint subsets,  $DS_1, \dots, DS_i, \dots, DS_k$ .

**Entropy:** For each resulting cluster, we can measure its entropy using Equation (8), where  $P_i(g_j)$  is the proportion of  $g_j$  data points in  $DS_i$ . The total entropy of the whole clustering (which considers all clusters) is calculated by Equation (9).

**Purity:** Purity measures the extent that a cluster contains only data from one gold-partition. The cluster purity is computed with Equation (10). The total purity of the whole clustering (all clusters) is computed with Equation (11).

$$entropy(DS_i) = - \sum_{j=1}^k P_i(g_j) \log_2 P_i(g_j) \quad (8)$$

$$entropy_{total} = \sum_{i=1}^k \frac{|DS_i|}{|DS|} entropy(DS_i) \quad (9)$$

$$purity(DS_i) = \max_j P_i(g_j) \quad (10)$$

$$purity_{total} = \sum_{i=1}^k \frac{|DS_i|}{|DS|} purity(DS_i) \quad (11)$$

### 5.3 Baseline Methods and Settings

The proposed *L-EM* algorithm is compared with a number of existing methods, which can be categorized into  $k$ -means (denoted by *Kmeans*) clustering series, topic modeling series, correlation series, lexical similarity series, and EM classification series.

In the *Kmeans* clustering series, including *Kmeans(TF)*, *Kmeans(PMI)* and *L-Kmeans(TF)*, the distributional similarity or distributional information is used.

**Table 2. Experimental results on 5 data sets, i.e., H, I, M, C, and V.**

Method	Entropy						Purity					
	H	I	M	C	V	avg	H	I	M	C	V	avg
<b>Kmeans(TF)</b>	<b>2.45</b>	2.32	<b>2.61</b>	2.67	<b>2.15</b>	<b>2.44</b>	<b>0.35</b>	0.31	<b>0.38</b>	0.32	<b>0.41</b>	<b>0.35</b>
<b>Kmeans(PMI)</b>	2.87	2.37	2.80	2.99	2.66	2.74	0.26	0.35	0.29	0.24	0.27	0.28
<b>LDA</b>	2.63	2.32	2.75	2.63	2.43	2.55	0.31	0.35	0.31	0.32	0.35	0.33
<b>mLSA</b>	2.62	<b>2.31</b>	2.74	<b>2.46</b>	2.38	2.50	0.32	<b>0.35</b>	0.31	<b>0.38</b>	0.37	0.35
<b>Newman(<math>\chi^2</math>)</b>	2.88	2.48	2.75	2.89	2.54	2.71	0.27	0.32	0.32	0.26	0.34	0.30
<b>Newman(PMI)</b>	3.06	2.42	2.98	3.31	3.64	3.08	0.26	0.30	0.26	0.21	0.23	0.25
<b>CHC</b>	<b>2.73</b>	<b>2.20</b>	<b>2.74</b>	<b>2.82</b>	<b>2.32</b>	<b>2.56</b>	<b>0.28</b>	<b>0.41</b>	<b>0.28</b>	<b>0.30</b>	<b>0.37</b>	<b>0.33</b>
<b>SHC</b>	3.35	2.55	3.15	3.41	3.76	3.24	0.25	0.34	0.23	0.21	0.23	0.25
<b>L-Rand</b>	2.18	2.19	2.52	2.58	1.84	2.26	0.47	0.41	0.42	0.38	0.52	0.44
<b>L-Kmeans(TF)</b>	1.90	2.04	2.24	2.18	1.56	1.98	0.52	0.43	0.46	0.47	0.57	0.49
<b>L-Kmeans'(TF)</b>	1.95	1.91	2.39	2.29	1.85	2.08	0.51	0.46	0.44	0.45	0.50	0.47
<b>L-LDA</b>	2.12	2.11	2.24	2.25	1.70	2.08	0.46	0.43	0.48	0.45	0.55	0.47
<b>DF-LDA</b>	2.19	1.91	2.11	2.14	1.64	2.00	0.41	0.49	0.46	0.47	0.50	0.47
<b>L-EM</b>	<b>1.89</b>	<b>1.59</b>	<b>2.14</b>	<b>2.04</b>	<b>1.58</b>	<b>1.84</b>	<b>0.55</b>	<b>0.59</b>	<b>0.51</b>	<b>0.53</b>	<b>0.59</b>	<b>0.55</b>

**Kmeans(TF)**<sup>2</sup>: This is the *k-means* clustering method [27] based on distributional similarity with cosine as the similarity measure and *TF* (Term Frequency) as the term weight.

**Kmeans(PMI)**<sup>2</sup>: PMI (Pointwise Mutual Information) values are used in place of terms' weights. This method is proposed in [24].

**L-Kmeans(TF)**: This is also based on *k-means*, but the clusters of the labeled data *L* are fixed at the initiation and remain unchanged afterward.

**L-Kmeans'(TF)**: This is similar to **L-Kmeans(TF)**, but the clusters of the labeled data *L* can change after the initiation.

In the topic modeling series, the topic-word distributions are used for clustering. These methods are listed as follows.

**LDA**<sup>2</sup>: LDA is a popular topic modeling method [5]. Given a set of documents, it outputs groups of terms and each group is said to belong to a topic. In our case, each feature expression is processed as a term. In LDA, a term may belong to more than one topic/group, but we take the topic/group with the maximum probability.

**mLSA**<sup>2</sup>: This is a state-of-the-art unsupervised method for solving the problem. It is also based on LDA, and has been discussed in related work.

**DF-LDA**<sup>2</sup>: This is the method proposed by [3]. It accepts the same inputs as **LDA**, and in additions the must-links (edges) extracted in Section 4 are also inputted. We used the code from the author's website<sup>3</sup>.

**L-LDA**<sup>2</sup>: This method is also based on **LDA**, but the labeled examples are used as seeds for each group/topic in topic modeling.

In the correlation series, correlations between feature expressions are used for clustering [28]. The representative methods are:

**Newman(PMI)**: Newman clustering method is employed to group feature expressions based on the PMI values between them, and the PMI threshold is set to its default value as in their paper<sup>4</sup>.

**Newman( $\chi^2$ )**: This is similar to **Newman(PMI)**, but it is based on  $\chi^2$  values among feature expressions. The  $\chi^2$  threshold is also set to its default value<sup>4</sup>.

In the lexical similarity series, the feature expressions are hierarchically clustered based on WordNet similarity (see lines 11~12 in Figure 2).

**CHC**: This method applies complete-link hierarchical clustering to cluster feature expressions into *k* clusters based on the WordNet similarity.

**SHC**: Compared with **CHC**, single-link hierarchical clustering is employed.

The proposed algorithm **L-EM** belongs to the EM classification series. In order to illustrate the impact of EM, **L-Rand** is implemented as follows.

**L-Rand**<sup>2</sup>: This method is based on the labeled data *L* generated by the method in Section 3.1, with the unlabeled data *U* randomly labeled. This baseline is used to show whether EM is making any improvement beyond the two soft constraints in Section 3.1.

Note that all these methods are compared in a variety of settings. For all the **LDA** based methods, i.e., **LDA**, **mLSA**, **L-LDA** and **DF-LDA**, the topic modeling parameters are set to their default values:  $\alpha=50/k$ ,  $\beta=0.01$ , where *k* is the number of groups/topics in the gold standard for each data set. The number of iterations is 1000. We used the LDA system in MALLE<sup>5</sup>. We modified it to suit different LDA-based methods, e.g., topic constraining. We implemented **mLSA**, **Kmeans** and changed the EM<sup>6</sup> implementation to take soft constraints. For all the **Kmeans** based methods, the distance function is the cosine similarity.

<sup>2</sup> This method's result depends on the random initiation, so we use the average value of 10 runs as the final result.

<sup>3</sup> [http://pages.cs.wisc.edu/~andrzej/research/df\\_lda.html](http://pages.cs.wisc.edu/~andrzej/research/df_lda.html)

<sup>4</sup> See the footnote 6 of the cited paper.

<sup>5</sup> <http://mallet.cs.umass.edu/>

<sup>6</sup> <http://alias-i.com/lingpipe/>

## 5.4 Evaluation Results

We now present and compare the results of *L-EM* and the 13 baseline methods based on 5 data sets. Since all these 13 methods require the number of clusters/topics as the input, in order to fairly compare the results, the number of clusters/topics is set to the number of gold partitions of each data set (see Section 5.1). The proposed algorithm *L-EM* also has a parameter, the number of merges, which is set to the number of clusters. Section 5.5 will study the effects of the number of merges.

All the results are shown in Table 2, where H, I, M, C and V are the abbreviations of the names of the data sets, and **avg** represents the average result of the 5 data sets. For *Entropy*, the smaller the value is the better, but for *Purity* the larger the better.

Table 2 clearly shows that the proposed algorithm (*L-EM*) outperforms all 13 baseline methods by a large margin on every dataset. In addition, we make the following observations:

- Methods without using any pre-existing knowledge all performed poorly, i.e., *Kmeans(TF)*, *Kmeans(PMI)*, *LDA*, *mLSA*, *Newman( $\chi^2$ )* and *Newman(PMI)*, which illustrate that only the distributional information of feature expressions is far from sufficient.
- Methods using only lexical similarity also do not work well, i.e., *CHC* and *SHC*. They are even worse than the 6 distributional information based methods. On the one hand, this observation shows that lexical similarity is also unreliable. On the other hand, it illustrates the importance of domain dependences of our task.

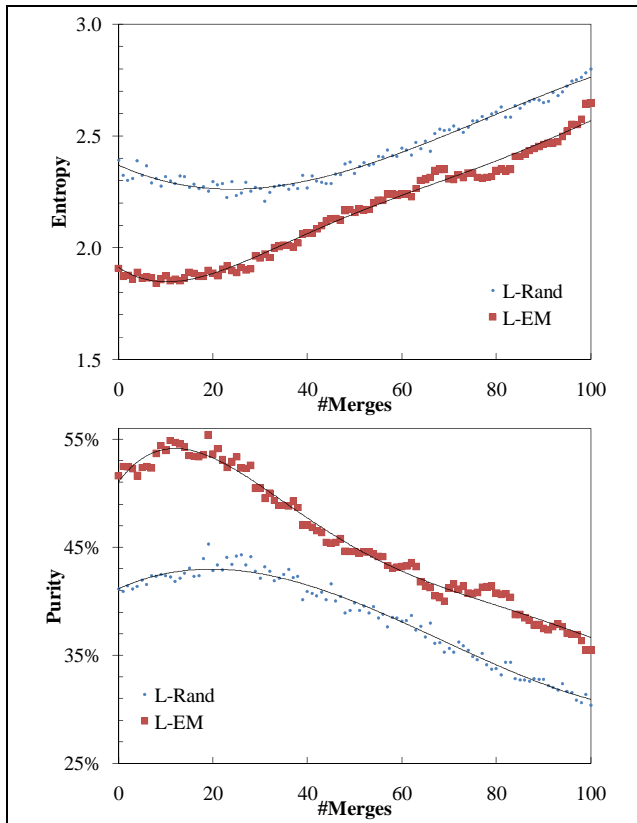


Figure 5. The influence of the number of merges to the proposed algorithm *L-EM*

- The PMI based methods, *Kmeans(PMI)* and *Newman(PMI)*, performed worse than *Kmeans(TF)* and *Newman( $\chi^2$ )*. This observation is consistent with the conclusion in [39], which observed that the quality of the PMI-based algorithms largely depends on the size of training corpus. The poorer results of *Kmeans(PMI)* and *Newman(PMI)* are probably the result of our small corpora.
- Methods using the labeled data *L* achieve better performances, i.e., *L-Rand*, *L-Kmeans*, *L-LDA*, *DF-LDA* and the proposed *L-EM*. Recall that the labeled data *L* is generated by two pieces of pre-existing knowledge. Thus, the two pieces of pre-existing knowledge proposed in this paper help to improve the results, which is intuitive.
- Compared with *L-Kmeans(TF)*, *L-Kmeans'(TF)*, *L-LDA* and *DF-LDA*, the proposed *L-EM* method outperformed them on average by more than 0.16 in Entropy and 8% in Purity. This shows that the classification based method (*L-EM*) is more suitable than clustering for our task. Note that, the *Kmeans* algorithm corresponds to a particular non-probabilistic limit of EM applied to mixtures of Gaussians [4]. The mixture components of the proposed algorithm are a set of multinomial distributions on which the naive Bayesian classification is based. Thus, we call the proposed algorithm a classification based method.
- Compared with *L-Rand*, the improvements made by *L-EM* are more than 0.42 in Entropy and 11% in Purity, which shows that the augmented EM algorithm is competent in revising the labeled feature expressions in *L* and grouping the unlabeled feature expressions in *U*.

## 5.5 Influence of the number of merges

We varied the number of merges as discussed in Section 3.1 from 0 to 100 to see how it impacts on the performance of the proposed algorithm *L-EM*. The results are given in Figure 5 (they are averages of the 5 data sets). When the number of merges is set to zero, it means that we skip the step 2 of Section 3.1, and the lexical similarity is not incorporated into our algorithm at all. With the growth of the number of merges, we incorporate more and more lexical similarity knowledge with the ranking based on similarities' strength. As shown in Figure 5, the performance of the proposed algorithm *L-EM* increases at first, and then decreases after some critical point. This phenomenon gives two pieces of information: the lexical similarity knowledge helps our task, but only the strong similarities are reliable. Weak similarities can harm the overall performance.

## 6. CONCLUSION

This paper studied the problem of product feature clustering for opinion mining applications. Although it is an unsupervised learning task, due to the poor performances of various clustering algorithms based on distributional and lexical similarities, we casted the problem as a semi-supervised learning task. Two soft constraints based on sharing of words and the lexical similarity were used to identify some initial labeled examples automatically for training. The paper then proposed to use the EM algorithm to solve the problem, which was improved by allowing the labeled examples to switch classes because the constraints can make mistakes. Empirical evaluations using 5 data sets show that the proposed method is superior to 13 baselines, which represent various current state-of-the-art solutions for this class of problems.

## 7. ACKNOWLEDGMENTS

This work was done when the first author was visiting the University of Illinois at Chicago. He was also partially supported by a grant (Grant No: 60875073) from National Natural Science Foundation of China.

## 8. REFERENCES

- [1] Agirre E, Alfonseca E, Hall K, Kravalova J, Pa ca M, and Soroa A. A study on similarity and relatedness using distributional and WordNet-based approaches. in Proceedings of ACL. 2009.19-27
- [2] Alvarez M and Lim S. A Graph Modeling of Semantic Similarity between Words. in Proceeding of the Conference on Semantic Computing. 2007.355-362
- [3] Andrzejewski D, Zhu X, and Craven M. Incorporating domain knowledge into topic modeling via Dirichlet forest priors. in Proceedings of ICML. 2009.25-32
- [4] Bishop C, Pattern recognition and machine learning. 2006: Springer.
- [5] Blei D, Ng A Y, and Jordan M I, Latent Dirichlet Allocation. *Journal of Machine Learning Research*, 2003. 3(3): 993-1022.
- [6] Bollegala D, Matsuo Y, and Ishizuka M. Measuring semantic similarity between words using web search engines. in Proceedings of WWW. 2007.757-766
- [7] Branavan S R K, Chen H, Eisenstein J, and Barzilay R. Learning document-level semantic properties from free-text annotations. in Proceedings of ACL. 2008.569-603
- [8] Brown P, Mercer R, Della Pietra V, and Lai J, Class-based n-gram models of natural language. *Computational Linguistics*, 1992. 18(4): 467-479.
- [9] Carenini G, Ng R, and Zwart E. Extracting knowledge from evaluative text. in Proceedings of International Conference on Knowledge Capture. 2005.11-18
- [10] Chen H, Lin M, and Wei Y. Novel association measures using web search with double checking. in ACL. 2006.1016
- [11] Fellbaum C, WordNet: An electronic lexical database. 1998: MIT press Cambridge, MA.
- [12] Guo H, Zhu H, Guo Z, Zhang X, and Su Z. Product feature categorization with multilevel latent semantic association. in Proceedings of CIKM. 2009.1087-1096
- [13] Harris Z S, Mathematical structures of language. Interscience tracts in pure and applied mathematics, no. 21. 1968, New York: Interscience Publishers. ix, 230 p.
- [14] Hu M and Liu B. Mining and summarizing customer reviews. in Proceedings of SIGKDD. 2004.168-177
- [15] Hughes T and Ramage D. Lexical semantic relatedness with random graph walks. in EMNLP. 2007.581-589
- [16] Jiang J and Conrath D. Semantic similarity based on corpus statistics and lexical taxonomy. in Proceedings of Research in Computational Linguistics. 1997.19-33
- [17] Jin W, Ho H, and Srihari R. OpinionMiner: a novel machine learning system for web opinion mining and extraction. in Proceedings of KDD. 2009.1195-1204
- [18] Kim S and Hovy E. Extracting opinions, opinion holders, and topics expressed in online news media text. in Proceedings of EMNLP. 2006.1065-1074
- [19] Kobayashi N, Inui K, and Matsumoto Y. Extracting aspect-evaluation and aspect-of relations in opinion mining. in Proceedings of EMNLP. 2007.1065-1074
- [20] Ku L, Liang Y-T, and Chen H-H. Opinion Extraction, Summarization and Tracking in News and Blog Corpora. in Proceedings of AAAI. 2006.100-107
- [21] Lee L. Measures of distributional similarity. 1999: Proceedings of ACL.25-32
- [22] Lin D. Automatic retrieval and clustering of similar words. 1998: Proceedings of ACL.768-774
- [23] Lin D. An information-theoretic definition of similarity. in Proceedings of ICML. 1998.296-304
- [24] Lin D and Wu X. Phrase clustering for discriminative learning. in Proceedings of ACL. 2009.1030-1038
- [25] Liu B, Web data mining: Exploring hyperlinks, contents, and usage data. 2006, Springer.
- [26] Liu B, Hu M, and Cheng J. Opinion Observer: Analyzing and Comparing Opinions on the Web. in Proceedings of WWW. 2005.342-351
- [27] MacQueen J. Some methods for classification and analysis of multivariate observations. in Proceedings of Symposium on Mathematical Statistics and Probability. 1966.281-297
- [28] Matsuo Y, Sakaki T, Uchiyama K, and Ishizuka M. Graph-based word clustering using a web search engine. 2006. Proceedings of EMNLP.542-550
- [29] Mei Q, Ling X, Wondra M, Su H, and Zhai C. Topic sentiment mixture: Modeling facets and opinions in weblogs. in Proceedings of WWW. 2007.171-180
- [30] Nigam K, McCallum A, Thrun S, and Mitchell T, Text classification from labeled and unlabeled documents using EM. *Machine Learning*, 2000. 39(2): 103-134.
- [31] Pang B and Lee L, Opinion Mining and Sentiment Analysis. Foundations and Trends in IR. 2008. 1-135.
- [32] Pantel P, Crestan E, Borkovsky A, Popescu A, and Vyas V. Web-scale distributional similarity and entity set expansion. in Proceedings of EMNLP. 2009.938-947
- [33] Pedersen T. Information Content Measures of Semantic Similarity Perform Better Without Sense-Tagged Text. in Proceedings of NAACL HLT. 2010
- [34] Pereira F, Tishby N, and Lee L. Distributional clustering of English words. in Proceedings of ACL. 1993.183-190
- [35] Popescu A-M and Etzioni O. Extracting Product Features and Opinions from Reviews. in EMNLP. 2005.339-346
- [36] Resnik P. Using information content to evaluate semantic similarity in a taxonomy. in IJCAI. 1995.448-453
- [37] Sahami M and Heilman T. A web-based kernel function for measuring the similarity of short text snippets. in Proceedings of WWW. 2006.377-386
- [38] Stoyanov V and Cardie C. Topic identification for fine-grained opinion analysis. in COLING. 2008.817-824
- [39] Su Q, Xiang K, Wang H, Sun B, and Yu S. Using pointwise mutual information to identify implicit features in customer reviews. in ICCPOL. 2006.22-30
- [40] Titov I and McDonald R. Modeling online reviews with multi-grain topic models. in WWW. 2008.111-120
- [41] Wagstaff K, Cardie C, Rogers S, and Schroedl S. Constrained k-means clustering with background knowledge. in In Proceedings of ICML. 2001.577-584
- [42] Yang D and Powers D. Measuring semantic similarity in the taxonomy of WordNet. 2005. Proceedings of the Australasian conference on Computer Science.322
- [43] Zhai Z, Liu B, Xu H, and Jia P, Grouping Product Features Using Semi-supervised Learning with Soft-Constraints, in Proceedings of COLING. 2010.