Product Feature Mining: Semantic Clues versus Syntactic Constituents

Liheng Xu, Kang Liu, Siwei Lai and Jun Zhao
National Laboratory of Pattern Recognition
Institute of Automation, Chinese Academy of Sciences, Beijing, 100190, China
{lhxu, kliu, swlai, jzhao}@nlpr.ia.ac.cn

Abstract

Product feature mining is a key subtask in fine-grained opinion mining. Previous works often use syntax constituents in this task. However, syntax-based methods can only use discrete contextual information, which may suffer from data sparsity. This paper proposes a novel product feature mining method which leverages lexical and contextual semantic clues. Lexical semantic clue verifies whether a candidate term is related to the target product, and contextual semantic clue serves as a *soft* pattern miner to find candidates, which exploits semantics of each word in context so as to alleviate the data sparsity problem. We build a semantic similarity graph to encode lexical semantic clue, and employ a convolutional neural model to capture contextual semantic clue. Then Label Propagation is applied to combine both semantic clues. Experimental results show that our semantics-based method significantly outperforms conventional syntax-based approaches, which not only mines product features more accurately, but also extracts more infrequent product features.

1 Introduction

In recent years, opinion mining has helped customers a lot to make informed purchase decisions. However, with the rapid growth of e-commerce, customers are no longer satisfied with the overall opinion ratings provided by traditional sentiment analysis systems. The detailed functions or attributes of products, which are called *product features*, receive more attention. Nevertheless, a product may have thousands of features, which makes it impractical for a customer to investigate them all. Therefore, mining product features automatically from online reviews is shown to be a key step for opinion summarization (Hu and Liu, 2004; Qiu et al., 2009) and fine-grained sentiment analysis (Jiang et al., 2011; Li et al., 2012).

Previous works often mine product features via **syntactic constituent matching** (Popescu and Etzioni, 2005; Qiu et al., 2009; Zhang et al., 2010). The basic idea is that reviewers tend to comment on product features in similar syntactic structures. Therefore, it is natural to mine product features by using syntactic patterns. For example, in Figure 1, the upper box shows a dependency tree produced by Stanford Parser (de Marneffe et al., 2006), and the lower box shows a common syntactic pattern from (Zhang et al., 2010), where <feature/NN> is a wildcard to be fit in reviews and *NN* denotes the required POS tag of the wildcard. Usually, the product name *mp3* is specified, and when *screen* matches the wildcard, it is likely to be a product feature of *mp3*.

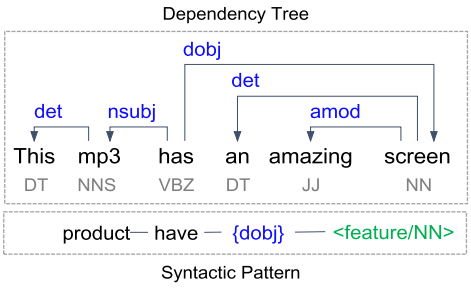


Figure 1: An example of syntax-based product feature mining procedure. The word *screen* matches the wildcard <feature/NN>. Therefore, *screen* is likely to be a product feature of *mp3*.

Generally, such syntactic patterns extract product features well but they still have some limitations. For example, the *product-have-feature* pattern may fail to find the *fm tuner* in a very similar case in Example 1(a), where the product is mentioned by using *player* instead of *mp3*. Similarly, it may also fail on Example 1(b), just with *have* replaced by *support*. In essence, syntactic pattern is

Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics, pages 336–346,
Baltimore, Maryland, USA, June 23-25 2014.

a kind of *one-hot representation* for encoding the contexts, which can only use partial and discrete features, such as some key words (e.g., *have*) or shallow information (e.g., POS tags). Therefore, such a representation often suffers from the data sparsity problem (Turian et al., 2010).

One possible solution for this problem is using a more general pattern such as *NP-VB-feature*, where *NP* represents a noun or noun phrase and *VB* stands for any verb. However, this pattern becomes too general that it may find many irrelevant cases such as the one in Example 1(c), which is not talking about the product. Consequently, it is very difficult for a pattern designer to balance between precision and generalization.

Example 1:
(a) *This player has an fm tuner.*
(b) *This mp3 supports wma file.*
(c) *This review has helped people a lot.*
(d) *This mp3 has some flaws.*

To solve the problems stated above, it is argued that deeper semantics of contexts shall be exploited. For example, we can try to automatically discover that the verb *have* indicates a part-whole relation (Zhang et al., 2010) and *support* indicates a product-function relation, so that both *sth. have* and *sth. support* suggest that terms following them are product features, where *sth.* can be replaced by any terms that refer to the target product (e.g., *mp3*, *player*, etc.). This is called **contextual semantic clue**. Nevertheless, only using contexts is not sufficient enough. As in Example 1(d), we can see that the word *flaws* follows *mp3 have*, but it is not a product feature. Thus, a noise term may be extracted even with high contextual support. Therefore, we shall also verify whether a candidate is really related to the target product. We call it **lexical semantic clue**.

This paper proposes a novel bootstrapping approach for product feature mining, which leverages both semantic clues discussed above. Firstly, some reliable product feature seeds are automatically extracted. Then, based on the assumption that terms that are more semantically similar to the seeds are more likely to be product features, a graph which measures semantic similarities between terms is built to capture lexical semantic clue. At the same time, a semi-supervised convolutional neural model (Collobert et al., 2011) is employed to encode contextual semantic clue. Finally, the two kinds of semantic clues are combined by a Label Propagation algorithm.

In the proposed method, words are represented by continuous vectors, which capture latent semantic factors of the words (Turian et al., 2010). The vectors can be unsupervisedly trained on large scale corpora, and words with similar semantics will have similar vectors. This enables our method to be less sensitive to lexicon change, so that the data sparsity problem can be alleviated . The contributions of this paper include:

- It uses semantics of words to encode contextual clues, which exploits deeper level information than syntactic constituents. As a result, it mines product features more accurately than syntax-based methods.
- It exploits semantic similarity between words to capture lexical clues, which is shown to be more effective than co-occurrence relation between words and syntactic patterns. In addition, experiments show that the semantic similarity has the advantage of mining infrequent product features, which is crucial for this task. For example, one may say "*This hotel has low water pressure*", where *low water pressure* is seldom mentioned, but fatal to someone's taste.
- We compare the proposed semantics-based approach with three state-of-the-art syntax-based methods. Experiments show that our method achieves significantly better results.

The rest of this paper is organized as follows. Section 2 introduces related work. Section 3 describes the proposed method in details. Section 4 gives the experimental results. Lastly, we conclude this paper in Section 5.

2 Related Work

In product feature mining task, Hu and Liu (2004) proposed a pioneer research. However, the association rules they used may potentially introduce many noise terms. Based on the observation that product features are often commented on by similar syntactic structures, it is natural to use patterns to capture common syntactic constituents around product features.

Popescu and Etzioni (2005) designed some syntactic patterns to search for product feature candidates and then used Pointwise Mutual Information (PMI) to remove noise terms. Qiu et al. (2009) proposed eight heuristic syntactic rules to jointly extract product features and sentiment lexicons, where a bootstrapping algorithm named *Double*

Propagation was applied to expand a given seed set. Zhang et al. (2010) improved Qiu's work by adding more feasible syntactic patterns, and the HITS algorithm (Kleinberg, 1999) was employed to rank candidates. Moghaddam and Ester (2010) extracted product features by automatical opinion pattern mining. Zhuang et al. (2006) used various syntactic templates from an annotated movie corpus and applied them to supervised movie feature extraction. Wu et al. (2009) proposed a phrase level dependency parsing for mining aspects and features of products.

As discussed in the first section, syntactic patterns often suffer from data sparsity. Furthermore, most pattern-based methods rely on term frequency, which have the limitation of finding infrequent but important product features. A recent research (Xu et al., 2013) extracted infrequent product features by a semi-supervised classifier, which used word-syntactic pattern co-occurrence statistics as features for the classifier. However, this kind of feature is still sparse for infrequent candidates. Our method adopts a semantic word representation model, which can train dense features unsupervisedly on a very large corpus. Thus, the data sparsity problem can be alleviated.

3 The Proposed Method

We propose a semantics-based bootstrapping method for product feature mining. Firstly, some product feature seeds are automatically extracted. Then, a semantic similarity graph is created to capture lexical semantic clue, and a Convolutional Neural Network (CNN) (Collobert et al., 2011) is trained in each bootstrapping iteration to encode contextual semantic clue. Finally we use Label Propagation to find some reliable new seeds for the training of the next bootstrapping iteration.

3.1 Automatic Seed Generation

The seed set consists of positive labeled examples (i.e. product features) and negative labeled examples (i.e. noise terms). Intuitively, popular product features are frequently mentioned in reviews, so they can be extracted by simply mining frequently occurring nouns (Hu and Liu, 2004). However, this strategy will also find many noise terms (e.g., commonly used nouns like *thing*, *one*, etc.). To produce high quality seeds, we employ a Domain Relevance Measure (DRM) (Jiang and Tan, 2010), which combines term frequency with a domain-specific measuring metric called Likelihood Ratio Test (LRT) (Dunning, 1993). Let $\lambda(t)$ denotes the LRT score of a product feature candidate t,

$$\lambda(t) = \frac{p^{k_1}(1-p)^{n_1-k_1}p^{k_2}(1-p)^{n_2-k_2}}{p_1^{k_1}(1-p_1)^{n_1-k_1}p_2^{k_2}(1-p_2)^{n_2-k_2}} \quad (1)$$

where k_1 and k_2 are the frequencies of t in the review corpus $\mathcal{R}$ and a background corpus[1] $\mathcal{B}$, n_1 and n_2 are the total number of terms in $\mathcal{R}$ and $\mathcal{B}$, $p = (k_1 + k_2)/(n_1 + n_2)$, $p_1 = k_1/n_1$ and $p_2 = k_2/n_2$. Then a modified DRM[2] is proposed,

$$DRM(t) = \frac{tf(t)}{max[tf(\cdot)]} \times \frac{1}{\log df(t)} \times \frac{|\log\lambda(t)| - min|\log\lambda(\cdot)|}{max|\log\lambda(\cdot)| - min|\log\lambda(\cdot)|} \quad (2)$$

where $tf(t)$ is the frequency of t in $\mathcal{R}$ and $df(t)$ is the frequency of t in $\mathcal{B}$.

All nouns in $\mathcal{R}$ are ranked by $DRM(t)$ in descent order, where top N nouns are taken as the positive example set V_s^+. On the other hand, Xu et al. (2013) show that a set of general nouns seldom appear to be product features. Therefore, we employ their General Noun Corpus to create the negative example set V_s^-, where N most frequent terms are selected. Besides, it is guaranteed that $V_s^+ \cap V_s^- = \emptyset$, i.e., conflicting terms are taken as negative examples.

3.2 Capturing Lexical Semantic Clue in a Semantic Similarity Graph

To capture lexical semantic clue, each word is first converted into *word embedding*, which is a continuous vector with each dimension's value corresponds to a semantic or grammatical interpretation (Turian et al., 2010). Learning large-scale word embeddings is very time-consuming (Collobert et al., 2011), we thus employ a faster method named Skip-gram model (Mikolov et al., 2013).

3.2.1 Learning Word Embedding for Semantic Representation

Given a sequence of training words $W = \{w_1, w_2, ..., w_m\}$, the goal of the Skip-gram model is to learn a continuous vector space $EB = \{e_1, e_2, ..., e_m\}$, where e_i is the word embedding of w_i. The training objective is to maximize the

[1] Google-n-Gram (http://books.google.com/ngrams) is used as the background corpus.

[2] The $df(t)$ part of the original DRM is slightly modified because we want a $tf \times idf$-like scheme (Liu et al., 2012).

average log probability of using word w_t to predict a surrounding word w_{t+j},

$$\hat{EB} = \underset{e_t \in EB}{\text{argmax}} \frac{1}{m} \sum_{t=1}^{m} \sum_{-c \le j \le c, j \neq 0} \log p(w_{t+j}|w_t; e_t) \quad (3)$$

where c is the size of the training window. Basically, $p(w_{t+j}|w_t; e_t)$ is defined as,

$$p(w_{t+j}|w_t; e_t) = \frac{exp(e'^T_{t+j} e_t)}{\sum_{w=1}^{m} exp(e'^T_w e_t)} \quad (4)$$

where e'_i is an additional training vector associated with e_i. This basic formulation is impractical because it is proportional to m. A hierarchical softmax approximation can be applied to reduce the computational cost to $log_2(m)$, see (Morin and Bengio, 2005) for details.

To alleviate the data sparsity problem, EB is first trained on a very large corpus[3] (denoted by $\mathcal{C}$), and then fine-tuned on the target review corpus $\mathcal{R}$. Particularly, for phrasal product features, a statistic-based method in (Zhu et al., 2009) is used to detect noun phrases in $\mathcal{R}$. Then, an Unfolding Recursive Autoencoder (Socher et al., 2011) is trained on $\mathcal{C}$ to obtain embedding vectors for noun phrases. In this way, semantics of infrequent terms in $\mathcal{R}$ can be well captured. Finally, the phrase-based Skip-gram model in (Mikolov et al., 2013) is applied on $\mathcal{R}$.

3.2.2 Building the Semantic Similarity Graph

Lexical semantic clue is captured by measuring semantic similarity between terms. The underlying motivation is that if we have known some product feature seeds, then terms that are more semantically similar to these seeds are more likely to be product features. For example, if *screen* is known to be a product feature of *mp3*, and *lcd* is of high semantic similarity with *screen*, we can infer that *lcd* is also a product feature. Analogously, terms that are semantically similar to negative labeled seeds are not product features.

Word embedding naturally meets the demand above: words that are more semantically similar to each other are located closer in the embedding space (Collobert et al., 2011). Therefore, we can use cosine distance between two embedding vectors as the semantic distance measuring metric. Thus, our method does not rely on term frequency to rank candidates. This could potentially improve the ability of mining infrequent product features.

Formally, we create a semantic similarity graph $G = (V, E, W)$, where $V = \{V_s \cup V_c\}$ is the vertex set, which contains the labeled seed set V_s and the unlabeled candidate set V_c; E is the edge set which connects every vertex pair (u, v), where $u, v \in V$; $W = \{w_{uv} : \cos(EB_u, EB_v)\}$ is a function which associates a weight to each edge.

3.3 Encoding Contextual Semantic Clue Using Convolutional Neural Network

The CNN is trained on each occurrence of seeds that is found in review texts. Then for a candidate term t, the CNN classifies all of its occurrences. Since seed terms tend to have high frequency in review texts, only a few seeds will be enough to provide plenty of occurrences for the training.

3.3.1 The architecture of the Convolutional Neural Network

The architecture of the Convolutional Neural Network is shown in Figure 2. For a product feature candidate t in sentence s, every consecutive subsequence q_i of s that containing t with a window of length l is fed to the CNN. For example, as in Figure 2, if $t = \{screen\}$, and $l = 3$, there are three inputs: $q_1 = [the, ipod, \textbf{screen}]$, $q_2 = [ipod, \textbf{screen}, is]$, $q_3 = [\textbf{screen}, is, impressive]$. Partially, t is replaced by a token "*PF*" to remove its lexicon influence[4].

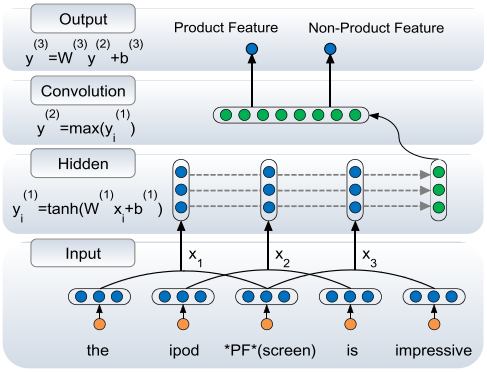


Figure 2: The architecture of the Convolutional Neural Network.

To get the output score, q_i is first converted into a concatenated vector $x_i = [e_1; e_2; ...; e_l]$, where e_j is the word embedding of the *j*-th word. In this way, the CNN serves as a *soft* pattern miner:

[3]Wikipedia(http://www.wikipedia.org) is used in practice.

[4]Otherwise, the CNN will quickly get overfitting on *t*, because very few seed lexicons are used for the training.

since words that have similar semantics have similar low-dimension embedding vectors, the CNN is less sensitive to lexicon change. The network is computed by,

$$y_i^{(1)} = tanh(W^{(1)}x_i + b^{(1)}) \quad (5)$$

$$y^{(2)} = max(y_i^{(1)}) \quad (6)$$

$$y^{(3)} = W^{(3)}y^{(2)} + b^{(3)} \quad (7)$$

where $y^{(i)}$ is the output score of the *i*-th layer, and $b^{(i)}$ is the bias of the *i*-th layer; $W^{(1)} \in \mathbb{R}^{h\times(nl)}$ and $W^{(3)} \in \mathbb{R}^{2\times h}$ are parameter matrixes, where n is the dimension of word embedding, and h is the size of nodes in the hidden layer.

In conventional neural models, the candidate term t is placed in the center of the window. However, from Example 2, when $l = 5$, we can see that the best windows should be the bracketed texts (Because, intuitively, the windows should contain *mp3*, which is a strong evidence for finding the product feature), where $t = \{screen\}$ is at the boundary. Therefore, we use Equ. 6 to formulate a max-convolutional layer, which is aimed to enable the CNN to find more evidences in contexts than conventional neural models.

Example 2:
(a) *The [**screen** of this mp3 is] great.*
(b) *This [mp3 has a great **screen**].*

3.3.2 Training

Let $\theta = \{EB, W^{(\cdot)}, b^{(\cdot)}\}$ denotes all the trainable parameters. The softmax function is used to convert the output score of the CNN to a probability,

$$p(t|X;\theta) = \frac{exp(y^{(3)})}{\sum_{j=1}^{|C|} exp(y_j^{(3)})} \quad (8)$$

where X is the input set for term t, and $C = \{0, 1\}$ is the label set representing product feature and non-product feature, respectively.

To train the CNN, we first use V_s to collect each occurrence of the seeds in $\mathcal{R}$ to form a training set T_s. Then, the training criterion is to minimize cross-entropy over T_s,

$$\hat{\theta} = \underset{\theta}{\operatorname{argmin}} \sum_{i=1}^{|T_s|} -\log \delta_i p(t_i|X_i;\theta) \quad (9)$$

where δ_i is the binomial target label distribution for one entry. Backpropagation algorithm with mini-batch stochastic gradient descent is used to solve this optimization problem. In addition, some useful tricks can be applied during the training. The weight matrixes $W^{(\cdot)}$ are initialized by *normalized initialization* (Glorot and Bengio, 2010). $W^{(1)}$ is pre-trained by an autoencoder (Hinton, 1989) to capture semantic compositionality. To speed up the learning, a momentum method is applied (Sutskever et al., 2013).

3.4 Combining Lexical and Contextual Semantic Clues by Label Propagation

We propose a Label Propagation algorithm to combine both semantic clues in a unified process. Each term $t \in V$ is assumed to have a label distribution $L_t = (p_t^+, p_t^-)$, where p_t^+ denotes the probability of the candidate being a product feature, and on the contrary, $p_t^- = 1 - p_t^+$. The classified results of the CNN which encode contextual semantic clue serve as the prior knowledge,

$$I_t = \begin{cases} (1,0), & if\ t \in V_s^+ \\ (0,1), & if\ t \in V_s^- \\ (r_t^+, r_t^-), & if\ t \in V_c \end{cases} \quad (10)$$

where (r_t^+, r_t^-) is estimated by,

$$r_t^+ = \frac{count^+(t)}{count^+(t) + count^-(t)} \quad (11)$$

where $count^+(t)$ is the number of occurrences of term t that are classified as positive by the CNN, and $count^-(t)$ represents the negative count.

Label Propagation is applied to propagate the prior knowledge distribution I to the product feature distribution L via semantic similarity graph G, so that a product feature candidate is determined by exploring its semantic relations to all of the seeds and other candidates globally. We propose an adapted version on the random walking view of the Adsorption algorithm (Baluja et al., 2008) by updating the following formula until L converges,

$$L^{i+1} = (1-\alpha)\mathbf{M}^T L^i + \alpha \mathbf{D} I \quad (12)$$

where $\mathbf{M}$ is the semantic transition matrix built from G; $\mathbf{D} = Diag[\log tf(t)]$ is a diagonal matrix of log frequencies, which is designed to assign higher "confidence" scores to more frequent seeds; and α is a balancing parameter. Particularly, when $\alpha = 0$, we can set the prior knowledge I without V_c to L^0 so that only lexical semantic clue is used; otherwise if $\alpha = 1$, only contextual semantic clue is used.

3.5 The Bootstrapping Framework

We summarize the bootstrapping framework of the proposed method in Algorithm 1. During bootstrapping, the CNN is enhanced by Label Propagation which finds more labeled examples for training, and then the performance of Label Propagation is also improved because the CNN outputs a more accurate prior distribution. After running for several iterations, the algorithm gets enough seeds, and a final Label Propagation is conducted to produce the results.

Algorithm 1: Bootstrapping using semantic clues

Input: The review corpus $\mathcal{R}$, a large corpus $\mathcal{C}$
Output: The mined product feature list P
Initialization: Train word embedding set EB first on $\mathcal{C}$, and then on $\mathcal{R}$

Step 1: Generate product feature seeds V_s (Section 3.1)

Step 2: Build semantic similarity graph G (Section 3.2)

while *iter* $<$ *MAX_ITER* **do**

  Step 3: Use V_s to collect occurrence set T_s from $\mathcal{R}$ for training

  Step 4: Train a CNN $\mathcal{N}$ on T_s (Section 3.3)
  Apply mini-batch SGD on Equ. 9;

  Step 5: Run Label Propagation (Section 3.4)
  Classify candidates using $\mathcal{N}$ to setup I;
  $L^0 \leftarrow I$;
  repeat
    $L^{i+1} \leftarrow (1-\alpha)\mathbf{M}^T L^i + \alpha \mathbf{D} I$;
  until $||L^{i+1} - L^i||^2 < \varepsilon$;

  Step 6: Expand product feature seeds
  Move top T terms from V_c to V_s;

  iter++

end

Step 7: Run Label Propagation for a final result L_f
Rank terms by L_f^+ to get P, where $L_f^+ > L_f^-$;

4 Experiments

4.1 Datasets and Evaluation Metrics

Datasets: We select two real world datasets to evaluate the proposed method. The first one is a benchmark dataset in Wang et al. (2011), which contains English review sets on two domains (*MP3* and *Hotel*)[5]. The second dataset is proposed by Chinese Opinion Analysis Evaluation 2008 (COAE 2008)[6], where two review sets (*Camera* and *Car*) are selected. Xu et al. (2013) had manually annotated product features on these four domains, so we directly employ their annotation as the gold standard. The detailed information can be found in their original paper.

[5]http://timan.cs.uiuc.edu/downloads.html
[6]http://ir-china.org.cn/coae2008.html

Evaluation Metrics: We evaluate the proposed method in terms of precision(P), recall(R) and F-measure(F). The English results are evaluated by *exact* string match. And for Chinese results, we use an *overlap* matching metric, because determining the exact boundaries is hard even for human (Wiebe et al., 2005).

4.2 Experimental Settings

For English corpora, the pre-processing are the same as that in (Qiu et al., 2009), and for Chinese corpora, the Stanford Word Segmenter (Chang et al., 2008) is used to perform word segmentation. We select three state-of-the-art syntax-based methods to be compared with our method:

DP uses a bootstrapping algorithm named as *Double Propagation* (Qiu et al., 2009), which is a conventional syntax-based method.

DP-HITS is an enhanced version of *DP* proposed by Zhang et al. (2010), which ranks product feature candidates by

$$s(t) = \log tf(t) * importance(t) \qquad (13)$$

where $importance(t)$ is estimated by the HITS algorithm (Kleinberg, 1999).

SGW is the Sentiment Graph Walking algorithm proposed in (Xu et al., 2013), which first extracts syntactic patterns and then uses random walking to rank candidates. Afterwards, word-syntactic pattern co-occurrence statistic is used as feature for a semi-supervised classifier TSVM (Joachims, 1999) to further refine the results. This two-stage method is denoted as ***SGW-TSVM***.

LEX only uses lexical semantic clue. Label Propagation is applied alone in a self-training manner. The dimension of word embedding $n = 100$, the convergence threshold $\varepsilon = 10^{-7}$, and the number of expanded seeds $T = 40$. The size of the seed set N is 40. To output product features, it ranks candidates in descent order by using the positive score $L_f^+(t)$.

CONT only uses contextual semantic clue, which only contains the CNN. The window size l is 5. The CNN is trained with a mini-batch size of 50. The hidden layer size $h = 250$. Finally, $importance(t)$ in Equ. 13 is replaced with r_t^+ in Equ. 11 to rank candidates.

LEX&CONT leverages both semantic clues.

Method	MP3			Hotel			Camera			Car			Avg.
	P	R	F	P	R	F	P	R	F	P	R	F	F
DP	0.66	0.57	0.61	0.66	0.60	0.63	0.71	0.70	0.70	0.72	0.65	0.68	0.66
DP-HITS	0.65	0.62	0.63	0.64	0.66	0.65	0.71	0.78	0.74	0.69	0.68	0.68	0.68
SGW	0.62	0.68	0.65	0.63	0.71	0.67	0.69	0.80	0.74	0.66	0.71	0.68	0.69
LEX	0.64	0.74	0.69	0.65	0.75	0.70	0.69	0.84	0.76	0.68	0.78	0.73	0.72
CONT	0.68	0.65	0.66	0.69	0.68	0.68	0.74	0.77	0.75	0.74	0.70	0.72	0.71
SGW-TSVM	0.73	0.71	0.72	0.75	0.73	0.74	0.78	0.81	0.79	0.76	0.73	0.74	0.75
LEX&CONT	0.74	0.75	0.74	0.75	0.77	0.76	0.80	0.84	0.82	0.79	0.79	0.79	0.78

Table 1: Experimental results of product feature mining. The precision or recall of *CONT* is the average performance over five runs with different random initialization of parameters of the CNN. Avg. stands for the average score.

4.3 The Semantics-based Methods vs. State-of-the-art Syntax-based Methods

The experimental results are shown in Table 1, from which we have the following observations:

(i) Our method achieves the best performance among all of the compared methods. We also equally split the dataset into five subsets, and perform one-tailed t-test ($p \leq 0.05$), which shows that the proposed semantics-based method (*LEX&CONT*) significantly outperforms the three syntax-based strong competitors (*DP*, *DP-HITS* and *SGW-TSVM*).

(ii) *LEX&CONT* which leverages both lexical and contextual semantic clues outperforms approaches that only use one kind of semantic clue (*LEX* and *CONT*), showing that the combination of the semantic clues is helpful.

(iii) Our methods which use only one kind of semantic clue (*LEX* and *CONT*) outperform syntax-based methods (*DP*, *DP-HITS* and *SGW*). Comparing *DP-HITS* with *LEX* and *CONT*, the difference between them is that *DP-HITS* uses a syntax-pattern-based algorithm to estimate $importance(t)$ in Equ. 13, while our methods use lexical or contextual semantic clue instead. We believe the reason that *LEX* or *CONT* is better is that syntactic patterns only use discrete and local information. In contrast, *CONT* exploits latent semantics of each word in context, and *LEX* takes advantage of word embedding, which is induced from global word co-occurrence statistic. Furthermore, comparing *SGW* and *LEX*, both methods are base on random surfer model, but *LEX* gets better results than *SGW*. Therefore, the word-word semantic similarity relation used in *LEX* is more reliable than the word-syntactic pattern relation used in *SGW*.

(iv) *LEX&CONT* achieves the highest recall among all of the evaluated methods. Since *DP* and *DP-HITS* rely on frequency for ranking product features, infrequent candidates are ranked low in their extracted list. As for *SGW-TSVM*, the features they used for the TSVM suffer from the data sparsity problem for infrequent terms. In contrast, *LEX&CONT* is frequency-independent to the review corpus. Further discussions on this observation are given in the next section.

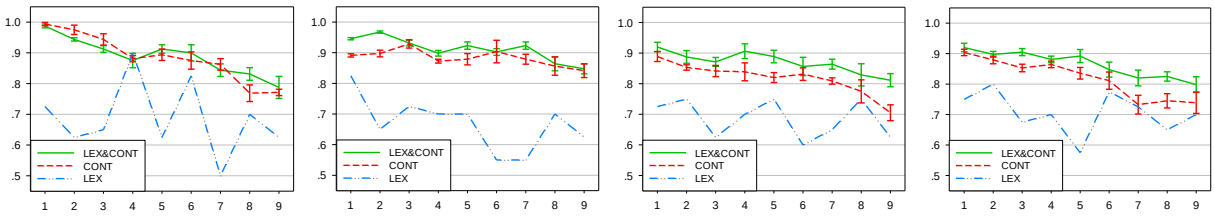
4.4 The Results on Extracting Infrequent Product Features

We conservatively regard 30% product features with the highest frequencies in $\mathcal{R}$ as *frequent features*, so the remaining terms in the gold standard are *infrequent features*. In product feature mining task, frequent features are relatively easy to find. Table 2 shows the recall of all the four approaches for mining frequent product features. We can see that the performance are very close among different methods. Therefore, the recall mainly depends on mining the infrequent features.

Method	MP3	Hotel	Camera	Car
DP	0.89	0.92	0.86	0.84
DP-HITS	0.89	0.91	0.86	0.85
SGW-TSVM	0.87	0.92	0.88	0.87
LEX&CONT	0.89	0.91	0.89	0.87

Table 2: The recall of frequent product features.

Figure 3 gives the recall of infrequent product features, where *LEX&CONT* achieves the best performance. So our method is less influenced by term frequency. Furthermore, *LEX* gets better recall than *CONT* and all syntax-based methods, which indicates that lexical semantic clue does aid to mine more infrequent features as expected.



(a) MP3 (b) Hotel (c) Camera (d) Car

Figure 4: Accuracy (y-axis) of product feature seed expansion at each bootstrapping iteration (x-axis). The error bar shows the standard deviation over five runs.

Method	MP3			Hotel			Camera			Car		
	P	R	F	P	R	F	P	R	F	P	R	F
FW-5	0.62	0.63	0.62	0.64	0.64	0.64	0.68	0.73	0.70	0.67	0.66	0.66
FW-9	0.64	0.65	0.64	0.66	0.68	0.67	0.70	0.76	0.73	0.71	0.70	0.70
CONT	0.68	0.65	0.66	0.69	0.68	0.68	0.74	0.77	0.75	0.74	0.70	0.72

Table 3: The results of convolutional method vs. the results of non-convolutional methods.

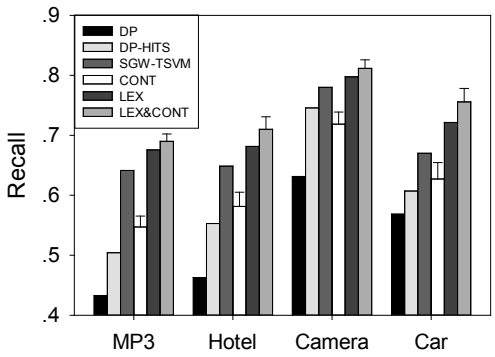


Figure 3: The recall of infrequent features. The error bar shows the standard deviation over five different runs.

4.5 Lexical Semantic Clue vs. Contextual Semantic Clue

This section studies the effects of lexical semantic clue and contextual semantic clue during seed expansion (Step 6 in Algorithm 1), which is controlled by α. When $\alpha = 1$, we get the *CONT*; and if α is set 0, we get the *LEX*. To take into account the correctly expanded terms for both positive and negative seeds, we use Accuracy as the evaluation metric,

$$Accuracy = \frac{\#TP + \#TN}{\#\ Extracted\ Seeds}$$

where TP denotes the *true positive* seeds, and TN denotes the *true negative* seeds.

Figure 4 shows the performance of seed expansion during bootstrapping, in which the accuracy is computed on 40 seeds (20 being positive and 20 being negative) expanded in each iteration. We can see that the accuracies of *CONT* and *LEX&CONT* retain at a high level, which shows that they can find reliable new product feature seeds. However, the performance of *LEX* oscillates sharply and it is very low for some points, which indicates that using lexical semantic clue alone is infeasible. On another hand, comparing *CONT* with *LEX* in Table 1, we can see that *LEX* performs generally better than *CONT*. Although *LEX* is not so accurate as *CONT* during seed expansion, its final performance surpasses *CONT*. Consequently, we can draw conclusion that *CONT* is more suitable for the seed expansion, and *LEX* is more robust for the final result production.

To combine advantages of the two kinds of semantic clues, we set $\alpha = 0.7$ in Step 5 of Algorithm 1, so that contextual semantic clue plays a key role to find new seeds accurately. For Step 7, we set $\alpha = 0.3$. Thus, lexical semantic clue is emphasized for producing the final results.

4.6 The Effect of Convolutional Layer

Two non-convolutional variations of the proposed method are used to be compared with the convolutional method in *CONT*. ***FW-5*** uses a traditional neural network with a fixed window size of 5 to replace the CNN in *CONT*, and the candidate term to be classified is placed in the center of the window. Similarly, ***FW-9*** uses a fixed window size of 9. Note that *CONT* uses a 5-term dynamic window containing the candidate term, so the exploited number of words in the context is equivalent to *FW-9*.

Table 3 shows the experimental results. We can see that the performance of *FW-5* is much worse than *CONT*. The reason is that *FW-5* only exploits half of the context as that of *CONT*, which is not sufficient enough. Meanwhile, although *FW-9* exploits equivalent range of context as that of *CONT*, it gets lower precisions. It is because *FW-9* has approximately two times parameters in the parameter matrix $W^{(1)}$ than that in Equ. 5 of *CONT*, which makes it more difficult to be trained with the same amount of data. Also, lengths of many sentences in the review corpora are shorter than 9. Therefore, the convolutional approach in *CONT* is the most effective way among these settings.

4.7 Parameter Study

We investigate two key parameters of the proposed method: the initial number of seeds N, and the size of the window l used by the CNN.

Figure 5 shows the performance under different N, where the F-Measure saturates when N equates to 40 and beyond. Hence, very few seeds are needed for starting our algorithm.

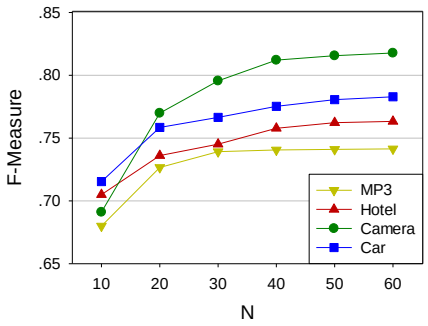


Figure 5: F-Measure vs. N for the final results.

Figure 6 shows F-Measure under different window size l. We can see that the performance is improved little when l is larger than 5. Therefore, $l = 5$ is a proper window size for these datasets.

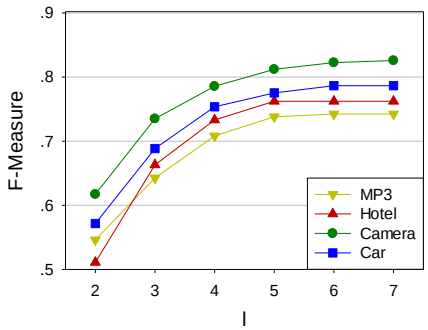


Figure 6: F-Measure vs. l for the final results.

5 Conclusion and Future Work

This paper proposes a product feature mining method by leveraging contextual and lexical semantic clues. A semantic similarity graph is built to capture lexical semantic clue, and a convolutional neural network is used to encode contextual semantic clue. Then, a Label Propagation algorithm is applied to combine both semantic clues. Experimental results prove the effectiveness of the proposed method, which not only mines product features more accurately than conventional syntax-based method, but also extracts more infrequent product features.

In future work, we plan to extend the proposed method to jointly mine product features along with customers' opinions on them. The learnt semantic representations of words may also be utilized to predict fine-grained sentiment distributions over product features.

Acknowledgement

This work was sponsored by the National Basic Research Program of China (No. 2012CB316300), the National Natural Science Foundation of China (No. 61272332 and No. 61202329), the National High Technology Development 863 Program of China (No. 2012AA011102), and CCF-Tencent Open Research Fund. This work was also supported in part by Noahs Ark Lab of Huawei Tech. Ltm.

References

Shumeet Baluja, Rohan Seth, D. Sivakumar, Yushi Jing, Jay Yagnik, Shankar Kumar, Deepak Ravichandran, and Mohamed Aly. 2008. Video suggestion and discovery for youtube: Taking random walks through the view graph. In *Proceedings of the 17th International Conference on World Wide Web*, WWW '08, pages 895–904, New York, NY, USA. ACM.

Pi-Chuan Chang, Michel Galley, and Christopher D. Manning. 2008. Optimizing chinese word segmentation for machine translation performance. In *Proceedings of the Third Workshop on Statistical Machine Translation*, StatMT '08, pages 224–232.

Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. 2011. Natural language processing (almost) from scratch. *J. Mach. Learn. Res.*, 12:2493–2537, November.

Marie-Catherine de Marneffe, Bill MacCartney, and Christopher D. Manning. 2006. Generating typed

dependency parses from phrase structure parses. In *Proceedings of the IEEE / ACL'06 Workshop on Spoken Language Technology*.

Ted Dunning. 1993. Accurate methods for the statistics of surprise and coincidence. *Comput. Linguist.*, 19(1):61–74, March.

Xavier Glorot and Yoshua Bengio. 2010. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the International Conference on Artificial Intelligence and Statistics.*

Geoffrey E. Hinton. 1989. Connectionist learning procedures. *Artificial Intelligence*, 40(1C3):185 – 234.

Minqing Hu and Bing Liu. 2004. Mining and summarizing customer reviews. In *Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '04, pages 168–177, New York, NY, USA. ACM.

Xing Jiang and Ah-Hwee Tan. 2010. Crctol: A semantic-based domain ontology learning system. *Journal of the American Society for Information Science and Technology*, 61(1):150–168.

Long Jiang, Mo Yu, Ming Zhou, Xiaohua Liu, and Tiejun Zhao. 2011. Target-dependent twitter sentiment classification. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies - Volume 1*, HLT '11, pages 151–160, Stroudsburg, PA, USA. Association for Computational Linguistics.

Thorsten Joachims. 1999. Transductive inference for text classification using support vector machines. In *Proceedings of the 16th International Conference on Machine Learning*, pages 200–209.

Jon M. Kleinberg. 1999. Authoritative sources in a hyperlinked environment. *J. ACM*, 46(5):604–632, September.

Fangtao Li, Sinno Jialin Pan, Ou Jin, Qiang Yang, and Xiaoyan Zhu. 2012. Cross-domain co-extraction of sentiment and topic lexicons. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers - Volume 1*, ACL '12, pages 410–419, Stroudsburg, PA, USA. Association for Computational Linguistics.

Kang Liu, Liheng Xu, and Jun Zhao. 2012. Opinion target extraction using word-based translation model. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 1346–1356, Jeju Island, Korea, July. Association for Computational Linguistics.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S. Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems*, pages 3111–3119.

Samaneh Moghaddam and Martin Ester. 2010. Opinion digger: An unsupervised opinion miner from unstructured product reviews. In *Proceedings of the 19th ACM International Conference on Information and Knowledge Management*, CIKM '10, pages 1825–1828, New York, NY, USA. ACM.

Frederic Morin and Yoshua Bengio. 2005. Hierarchical probabilistic neural network language model. In *Proceedings of the international workshop on artificial intelligence and statistics*, AISTATS05, pages 246–252.

Ana-Maria Popescu and Oren Etzioni. 2005. Extracting product features and opinions from reviews. In *Proceedings of the conference on Human Language Technology and Empirical Methods in Natural Language Processing*, HLT '05, pages 339–346.

Guang Qiu, Bing Liu, Jiajun Bu, and Chun Chen. 2009. Expanding domain sentiment lexicon through double propagation. In *Proceedings of the 21st international jont conference on Artifical intelligence*, IJCAI'09, pages 1199–1204.

Richard Socher, Eric H Huang, Jeffrey Pennington, Andrew Y Ng, and Christopher D Manning. 2011. Dynamic pooling and unfolding recursive autoencoders for paraphrase detection. In *NIPS'2011*, volume 24, pages 801–809.

Ilya Sutskever, James Martens, George Dahl, and Geoffrey Hinton. 2013. Distributed representations of words and phrases and their compositionality. In *Proceedings of the 30 th International Conference on Machine Learning*.

Joseph Turian, Lev Ratinov, and Yoshua Bengio. 2010. Word representations: A simple and general method for semi-supervised learning. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, ACL '10, pages 384–394, Stroudsburg, PA, USA. Association for Computational Linguistics.

Hongning Wang, Yue Lu, and ChengXiang Zhai. 2011. Latent aspect rating analysis without aspect keyword supervision. In *Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '11, pages 618–626, New York, NY, USA. ACM.

Janyce Wiebe, Theresa Wilson, and Claire Cardie. 2005. Annotating expressions of opinions and emotions in language. *Language Resources and Evaluation*, 39(2-3):165–210.

Yuanbin Wu, Qi Zhang, Xuanjing Huang, and Lide Wu. 2009. Phrase dependency parsing for opinion mining. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing: Volume 3 - Volume 3*, EMNLP '09, pages 1533–1541, Stroudsburg, PA, USA. Association for Computational Linguistics.

Liheng Xu, Kang Liu, Siwei Lai, Yubo Chen, and Jun Zhao. 2013. Mining opinion words and opinion targets in a two-stage framework. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1764–1773, Sofia, Bulgaria, August. Association for Computational Linguistics.

Lei Zhang, Bing Liu, Suk Hwan Lim, and Eamonn O'Brien-Strain. 2010. Extracting and ranking product features in opinion documents. In *Proceedings of the 23rd International Conference on Computational Linguistics: Posters*, COLING '10, pages 1462–1470, Stroudsburg, PA, USA. Association for Computational Linguistics.

Jingbo Zhu, Huizhen Wang, Benjamin K. Tsou, and Muhua Zhu. 2009. Multi-aspect opinion polling from textual reviews. In *Proceedings of the 18th ACM Conference on Information and Knowledge Management*, CIKM '09, pages 1799–1802, New York, NY, USA. ACM.

Li Zhuang, Feng Jing, and Xiao-Yan Zhu. 2006. Movie review mining and summarization. In *Proceedings of the 15th ACM International Conference on Information and Knowledge Management*, CIKM '06, pages 43–50, New York, NY, USA. ACM.