SLTM: A Sentence Level Topic Model for Analysis of Online Reviews

Yuhan Zhang and Haiping Xu
Computer and Information Science Department
University of Massachusetts Dartmouth, Dartmouth, MA 02747, USA
{yzhang5, hxu}@umassd.edu

Abstract—**Due to large amounts of reviews for many similar online products, users often feel difficult to determine which products have the most desirable features that they want. In this paper, we propose a model-based approach to analyzing online reviews and identifying the strengths and weaknesses of a product by its product features. We propose a Sentence Level Topic Model (SLTM), which can classify review sentences into different classes corresponding to different product features. The model contains a hidden layer, called the topic layer, between corpus and words. Once a SLTM has been trained with sufficient labeled data points, it can identify the most related topic (i.e., product feature) for each sentence. To capture a reviewer' opinion, we perform sentiment analysis for each review sentence, and derive the weighted feature preference vectors for the review. Finally, we combine the results of all review comments for a product into a review summary. The case study shows that by comparing the review summaries of similar online products, users may have a much easier time to find their desired products.**

Keywords-Electronic commerce; product review; product feature; topic model; feature extraction; sentiment analysis.

I. Introduction

Many e-commerce websites adopt the average star rating mechanism to help customers with their buying decisions; however, such ratings are not accurate and do not necessarily reflect the actual quality of the products [1]. To deal with this issue, major e-commerce websites allow users to provide reviews for the products they bought. A popular online product listed at e-commerce websites such as Amazon, often receives hundreds or even thousands of reviews. Due to the large amount of reviews, customers often have a hard time to read all of them, and tend to miss important product information. To provide a better view about the products and save customers' time for browsing the reviews, it is necessary to provide an approach that can automatically analyze all reviews of a product and output its strengths and weaknesses in terms of its product features. In this paper, we propose a Sentence Level Topic Model (SLTM), which can be used to classify sentences into different topics, where each topic refers to a product feature (in this paper, we will use the terms "topic" and "feature" interchangeably). The SLTM introduces a hidden layer, called the topic layer, between corpus and words. By applying Bayes' rule, SLTM can effectively identify the product feature in each sentence. When a feature is identified, we perform sentiment analysis of the sentence to find how the reviewer likes or dislikes the feature. When all review comments have been processed, we combine the results into a review summary. To illustrate the feasibility and effectiveness of our approach, we retrieved and processed review comments for a number of similar products from Amazon. We show that by comparing the strengths and weaknesses of product features among similar products, our approach can greatly save customers' time for making decisions on selecting the most desired online products.

In the past decades, review mining has attracted a great deal of attention due to the rapid growth of online reviews. Pang and Lee employed three machine-learning approaches to label the polarity of IMDb movie reviews [2]. They proposed to first extract the subjective portion of text with a graph min-cut algorithm, and then feed them into a sentiment classifier. Rather than applying the straightforward frequency-based bag-of-words feature selection methods, Whitelaw *et al.* defined the concept of "adjectival appraisal groups" headed by an appraising adjective and optionally modified by words like "not" or "very" [3]. Each appraisal group was further assigned four types of features: attitude, orientation, graduation, and polarity. Different from the above approaches, our SLTM method is a statistical approach using topic modeling, which can be directly applied at sentence level to maximize the accuracy for text mining.

In machine learning and Natural Language Processing (NLP), a topic model is defined as a type of statistical model for discovering the abstract "topics" that occur in a collection of documents. An early topic model for text mining, called Latent Semantic Indexing (LSI), proposed by Papadimitriou *et al.*, is an information retrieval technique based on the spectral analysis of the term-document matrix [4]. They showed that under certain conditions, LSI could capture the underlying semantics of the corpus and achieve improved retrieval performance. Blei *et al.* developed a very commonly used topic model, called Latent Dirichlet Allocation (LDA) [5], which is a three-level hierarchical Bayesian model that allows documents to have a mixture of topics. Different from LSI and LDA, since one sentence in a review typically contains at most one topic, our model only requires one matrix, namely the topic-word matrix. Therefore, SLTM is a one-layer classifier, which could be more suitable for analysis of online reviews.

Symbolic techniques are one of the major approaches to detecting sentiment from text [6]. Symbolic techniques assume the corpus is a "bag of words"; therefore, the relationships between the individual words are not considered. Kamps *et al.* used the lexical database WordNet to determine the emotional content of a word along different dimensions [7]. WordNet is a

(DOI reference number: 10.18293/SEKE2016-230)

large lexical database of words containing nouns, verbs, adjectives and adverbs, which are grouped into sets of cognitive synonyms (synsets) that describe different concepts. Based on WordNet, SentiWordNet has been developed as an extended lexical resource for opinion mining [8]. To support sentiment analysis, SentiWordNet assigns each synset of WordNet three sentiment scores, namely positivity, negativity and objectivity. In this paper, we adopt a similar tool called the Stanford CoreNLP toolkit, which supports most of the common core NLP functions, including Part-Of-Speech (POS) tagger and sentiment analysis [9].

II. Topic Model Based Review Analysis

A. A Framework for Sentence Level Topic Modeling

The framework for sentence level topic modeling of online product reviews consists of four major parts, namely review preprocessing, feature extraction, sentiment analysis, and product summary. As shown in Fig. 1, the system is flexible and highly modularized, which supports processing a collection of reviews for a certain online product. The reviews can be processed either sequentially or in parallel as we consider them being written by independent customers.

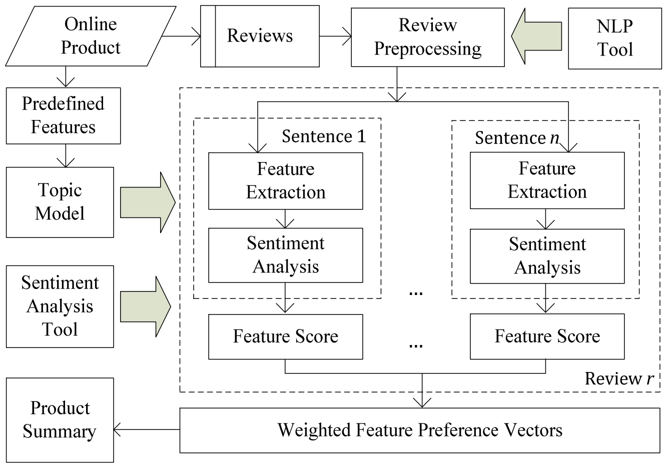


Figure 1. A framework for sentence level topic modeling

To identify product features and analyze sentiment, we first collect the reviews for a class of online products P (e.g. camera). Then we define a set of product features for P. As shown in Fig. 1, the input of the framework is a collection of reviews for product $p \in P$, and the output is a product summary that specifies the weighted feature preference for each predefined product feature. In the review preprocessing module, a review is split into a number of sentences that are parsed and tagged using an NLP tool. The tagged sentences are then sent to the feature extraction modules, where we use SLTM to calculate the probability of each feature. If a feature is identified in a sentence, the feature extraction module outputs the sentence to the sentiment analysis module. The sentiment analysis module then calculates a feature score for the sentence as an integer between 0-4, where 0 representing very negative and 4 representing very positive.

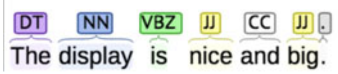
Once we calculated the weighted feature scores for each review r, we combine them in feature preference vectors, and derive the product summary that shows how strongly each of the product features is supported or not supported by buyers.

B. Review Preprocessing

The review preprocessing module processes a single review each time. Using Standford CoreNLP sentence split toolkit, a review is split into a number of sentences. During this process, sentences with less than two words are ignored since a meaningful sentence should at least have a topic word and a sentiment word. Then we use Standford CoreNLP POS tagger to indicate each word in the sentence as noun, verb, adjective, adverb or other tags. We use the following sentence as an example to show how POS tagger parses a sentence:

"The display is nice and big."

Using the CoreNLP toolkit, the parser outputs the sentence with the following POS tags:



In the above tagged sentence, the tags DT, NN, VBZ, JJ, and CC stand for "Determiner", "Noun", "Verb, third person singular present", "Adjective", "Coordinating Conjunction", respectively. After detecting all sentences in a review and parsing each sentence with POS tags, we save each sentence along with the POS tag information into a local file, which will be used to identify product features as described in the following sections.

Based on our observation, a topic in a sentence is usually determined by nouns or adjectives. Thus we can simplify the model by using POS tags to get rid of those noise data (words such as "You", "I", "a", "the", etc.). As in LSA, we consider a sentence a "bag of words". Therefore, we output only the words with tag NN or JJ to the feature extraction module for further processing. In addition, for each word, we consider the lemma only, which is the dictionary form of a set of words. For example, words "likes", "like", "liked" are forms of the same lexeme, with "like" as the lemma.

III. Feature Indetication Using SLTM

A. Topic-Word Matrix

To develop a topic model for product type P, we predefine a list of product features based on customers' interests as well as product specification. The predefined features are considered different topics in our topic model. Suppose we have defined K features for P. We create a *topic-word* matrix Φ with K rows and V columns, where V is a dynamic number that increases by 1 each time a new word appears. Initially, Φ has K rows and 0 column, which is an empty matrix due to the emptiness of the initial vocabulary list. After the matrix has been expanded, each row of the matrix represents the distribution of words in the corresponding topic, while each column indicates how the corresponding word has co-occurred with different topics. The elements of Φ are defined as follows.

Φ_k: distribution of words in topic k (k ranges from $0 \sim K$-1)

$\Phi[k, w]$: counts of word w occurring in topic k (w ranges from $0 \sim V$-1)

The matrix is used to record topic-word co-occurrence information with each cell initially set to 0. To expand the column (vocabulary) in matrix Φ and increase the number in each cell of Φ, we first manually create a training dataset with labeled sentences as follows.

For a review sentence, we tag it with proper product feature if it contains one. For example, the sentence "The display is nice and big." describes the product feature "ViewScreen"; therefore, we put a "#ViewScreen" tag at the end of this sentence:

"*The display is nice and big.*" #ViewScreen

Similarly, we can label the following two review sentences in the same way:

"*They had the best price on the camera, and got it here on time!*" #Price

"*It is also much easier to see camera settings on the big LCD.*" #ViewScreen

Note that a customer may use various words or phrases to describe a feature. For example, words or phrases "display" and "big LCD" all refer to the same feature "ViewScreen".

Since only nouns and adjectives are considered inputs of the topic model, each labeled data point is defined as a bag of nouns and adjectives with its associated feature tag. For example, the data points for the aforementioned three tagged sentences are recorded as follows:

"*display, nice, big*" #ViewScreen
"*best, price, camera, time*" #Price
"*easy, camera, setting, big, LCD*" #ViewScreen

The procedure for setting up the topic-word matrix Φ is presented as Algorithm 1. In this algorithm, we first initialize Φ with the vocabulary list *vl* being empty. Then for each data point *d*, let *f* be the feature (tag) of *d*, and *topicIndex* be the index of *f* in the topic list *tl*. For each word *w* in *d*, check if *w* appears in *vl*. If *w* is in *vl*, let *wordIndex* be the index of *w*; otherwise, add *w* to *vl* and expand Φ by one column for the new word *w*, and let *wordIndex* be the index of the last column of the matrix. Finally, the topic-word co-occurrence count Φ[*topicIndex*][*wordIndex*] is increased by one. Note that the topic-word matrix Φ can be updated in the same way when more labeled data points become available.

Algorithm 1: Set up Topic-Word Matrix Φ

Input: Matrix Φ, topic list *tl*, vocabulary list *vl*, and a training dataset
Output: updated matrix Φ

```
1.  Initialize matrix Φ with vl as an empty list
2.  for each data point d from the training dataset
3.     let f be the feature of d, and topicIndex be the index of f in tl
4.     for each word w in d
5.        if vl contains w
6.           let wordIndex be the index of w in vl
7.        else
8.           add w to vl, add 1 column to Φ, and set wordIndex = |vl| - 1
9.        increase Φ[topicIndex][wordIndex] by 1
10. return matrix Φ
```

B. Feature Identification

Once we have updated the topic-word matrix using all data points from the training dataset, we can use it as a discriminative model to classify sentences into different class.

Let T_0, …, $T_{K\text{-}1}$ be the list of topics, and $P(T_k)$, where $0 \le k \le K\text{-}1$, be the probability that topic T_k appears in a sentence. The probability $P(T_k)$ can be calculated as in Eq. (1).

$$P(T_k) = \frac{N(T_k)}{\sum_{i=0}^{K-1} N(T_i)} \tag{1}$$

where $N(T_k)$ is the number of times that topic T_k has appeared in the training dataset, and $\sum_{i=0}^{K-1} N(T_i)$ is the total number of data points in the training dataset. Obviously, $\sum_{k=0}^{K-1} P(T_k) = 1$.

Let $P(T_k \mid S)$ be the probability that topic T_k appears given sentence S, where $0 \le k \le K\text{-}1$, and $\sum_{k=0}^{K-1} P(T_k \mid S) = 1$. Assume each sentence has at most one topic, and the topic of sentence S *Topic_S* can be determined by Eq. (2).

$$Topic\_S = \begin{cases} T_m & \text{if } P(T_m \mid S) > 0.5 \\ Null & \text{otherwise} \end{cases} \tag{2}$$
$$\text{where } P(T_m \mid S) = \max(P(T_1 \mid S), P(T_2 \mid S), \ldots, P(T_K \mid S))$$

Note that in Eq. (2), 0.5 is the threshold for topic identification. If there is a tie for determining T_m, none of the topics satisfies the condition $P(T_k \mid S) > 0.5$. In this case, the topic of sentence S is determined as "*Null*".

Now consider the calculation of probability of sentence S having topic T_k. By the Bayes' rule, $P(T_k \mid S)$ can be calculated as in Eq. (3.1).

$$P(T_k \mid S) = \frac{P(S \mid T_k) * P(T_k)}{P(S)} \tag{3.1}$$

As we treat S as a "bag of words", where each word has a unigram meaning, to simplify our model, we consider the words bring independent from each other. Let S be $\{w_1, \ldots, w_n\}$. Eq. (3.1) can be rewritten as in Eq. (3.2).

$$P(T_k \mid S) = \frac{P(w_1,\ldots,w_n \mid T_k) * P(T_k)}{P(w_1,\ldots,w_n)} \tag{3.2}$$

where $P(w_1,\ldots,w_n \mid T_k)$ and $P(w_1,\ldots,w_n)$ are defined as in Eq. (4.1-4.2).

$$P(w_1,\ldots,w_n \mid T_k) = \Pi_{i=1}^{n} P(w_i \mid T_k) \tag{4.1}$$

$$P(w_1,\ldots,w_n) = \Pi_{i=1}^{n} P(w_i) \tag{4.2}$$

Note that $P(w_i \mid T_k)$, where $1 \le i \le n$, is the probability that word w_i appears in a sentence given topic T_k. Since Φ_k is the distribution of words in topic T_k, $P(w_i \mid T_k)$ can be calculated as in Eq. (5).

$$P(w_i \mid T_k) = \frac{\Phi[k,i]}{\sum_{l=0}^{V-1} \Phi[k,l]}, \text{where } 1 \le i \le n \tag{5}$$

where $\sum_{l=0}^{V-1} \Phi[k,l]$ is the sum of word counts in topic T_k. Since each $P(w_i \mid T_k)$ could be a very small decimal, to avoid accuracy overflow errors in calculating $P(w_1, \ldots, w_n \mid T_k)$, we first calculate the logarithm of $P(w_1, \ldots, w_n \mid T_k)$ as in Eq. (6.1), and then derive $P(w_1, \ldots, w_n \mid T_k)$ as in Eq. (6.2).

$$\ln(P(w_1,\ldots,w_n \mid T_k)) = \sum_{i=1}^{n} (\ln(\Phi[k,i]) - \ln(\sum_{l=0}^{V-1} \Phi[k,l])) \tag{6.1}$$

$$P(w_1,\ldots,w_n \mid T_k) = e^{\ln(P(w_1,\ldots,w_n \mid T_k))} \tag{6.2}$$

Finally, $P(w_i)$, where $1 \le i \le n$, is the probability that word w_i appears in a sentence. $P(w_i)$ can also be calculated using the topic-word matrix; however, the calculation of $P(w_i)$ is not necessary, as it is a constant for all topics, so is $P(w_1, w_1, \dots w_n)$. Since $\sum_{k=0}^{K-1} P(T_k \mid S) = 1$, we can calculate $P'(T_k \mid S)$ as in Eq. (7.1), and then normalize it into $P(T_k \mid S)$ using Eq. (7.2).

$$P'(T_k \mid S) = P(w_1, \dots, w_n \mid T_k) * P(T_k) \tag{7.1}$$

$$P(T_k \mid S) = \frac{P'(T_k \mid S)}{\sum_{i=0}^{K-1} P'(T_k \mid S)} \tag{7.2}$$

IV. Sentiment Analysis and Review Summary

A. Sentiment Analysis

Once we have extracted product features from review sentences, the next task is to analyze the customer's preference of the extracted feature. Here we utilize Stanford Sentiment Analysis toolkit for this purpose. Stanford Sentiment Analysis model uses a type of Recursive Neural Network (RNN) based on the grammatical structure of a sentence. The training data set of this model is from Stanford Sentiment Treebank, and it is worth mentioning that users can help to improve this model while using it. The toolkit first split the sentence into phases, then phrases into words. For the sentiment calculation, the toolkit uses bottom up algorithm, which calculates the sentiment for each word first, then for each phrase, and eventually for the whole sentence. Since we assume one sentence contains only one topic, the sentiment score of whole sentence is also the score for the topic. According to reference [10], the model's accuracy on a single sentence classification is above 80%, while the accuracy of predicting fine-grained sentiment for all phrases could be even higher.

B. Review Summary

Our approach is to extract features and calculate sentiment scores at the sentence level. Once we have recorded the feature score for each sentence, we can summarize the user preferences of the predefined product features for a certain product. The summarization is based on the procedure for combining weighted feature preference scores, which is described as in Algorithm 2.

Algorithm 2: Feature Preference Score Combination

Input: Feature scores of all reviews for product *p*
Output: Vectors LIKE and DISLIKE indicating how customers like or dislike product *p*, respectively, in terms of the product features.

```
1.  initialize two K-dimension vectors LIKE and DISLIKE to 0.
2.  for each review r for product p
3.      initialize K-dimension vector PREFER to 0.
4.      for each sentence s in r
5.          let f be the feature in s, and tIndex be the index of feature
            f in the vectors; let sentiScore be the sentiment score of s
6.          switch sentiScore
7.              case 0: PREFER[tIndex] = -1;  break; // very negative
8.              case 1: PREFER[tIndex] = -0.7;  break; // negative
9.              case 2: PREFER[tIndex] = 0;  break;     // neutral
10.             case 3: PREFER[tIndex] = 0.7;  break;  // positive
11.             case 4: PREFER[tIndex] = 1;  break; // very positive
12.     for each element i in PREFER, where 0 ≤ i ≤ K-1
13.         if PREFER[i] > 0 LIKE[i] += PREFER[i]
14.         else DISLIKE[i] += PREFER[i] * (-1)
15. return vectors LIKE and DISLIKE
```

In Algorithm 2, all review results are combined to evaluate how customers like or dislike the predefined product features of a product. Note that if a review contains multiple review sentences related to feature *f*, according to the algorithm, only the last sentence given by the reviewer is considered.

V. Case Study

To demonstrate the feasibility of our approach, we use an example of online products from Amazon website. The product type is camera, with listed online product such as "Nikon Coolpix L300 Digital Camera (Black)." Although the products we selected are good ones due to their high average star ratings, customers will still want to know different features of the products. For example, does the battery last long enough or is the camera small and light enough for a long trip? By analyzing the review comments, our approach may help customers to answer such questions.

A. Product Featues

To evaluate the product features of digital camera, we define 9 customer-interested features as well as a "Null" feature, listed as follows.

1. **Null**: is a dummy feature that indicates a sentence having no specific topic.
2. **Lens**: is an optical device on a camera that can change the focus of a light beam through refraction. The quality of lens has a strong impact on the quality of the camera.
3. **SizeWeight**: refers to the appearance of a camera in terms of its size and/or weight.
4. **Price**: indicates whether the price of the camera is reasonable or not.
5. **Resolution**: is a feature of picture quality, and higher resolution typically indicates higher picture quality.
6. **Stabilization**: is a feature that indicates whether a camera can effectively prevent or compensate for unwanted camera movement.
7. **Accessory**: refers to the quality or availability of camera accessories, such as the quality of battery and the availability of SD card option.
8. **Shutter**: is a device associated with a camera that allows light to pass for a determined period of time.
9. **ViewScreen**: also known as LCD or viewfinder, which is a device used to display images.
10. **Mode**: refers to manual mode or automatic mode that can be used in various situations.

Note that the "Null" feature is a dummy one indicating that a sentence does not describe any feature of the camera. For example, sentences such as "I love this camera so much," "My last camera was the Canon Powershot A495," or "So happy I got this," do not make comments on any product feature. Instead, they simply either state a fact or express a general feeling about the camera.

B. Experimental Results

We trained the SLTM using about 1500 labeled data points and achieved 95% accuracy. In the experiments, we chose four different digital cameras sold at Amazon, and produced their review summaries. The selected four digital cameras are listed in Table 1, which have similar prices around $200 and all sold by top-100 camera sellers at Amazon. Moreover, they all have above 4.0 average star ratings; thus, it is hard for a buyer to determine which one is the most suitable one to buy.

Table 1. Four different digital cameras listed at Amazon

Product Name	**ASIN**	**Star Ratings**	**Price**	**Reviews**
Nikon Coolpix	B00HQDBLDO	4.3	$165	384
Canon PowerShot SX520	B00M0QVTOS	4.4	$269	315
Canon Rebel XT (Used)	B0007QKN22	4.1	$205	648
Canon PowerShot SX400	B00M0QVG3W	4.3	$150	364

The review summaries for comparing the four selected cameras are presented in Fig. 2. In the figure, the left-side bars indicate the weighted DISLIKE scores for each product feature; while the right-side bars indicate the weighted LIKE scores for the features. From the figure, we can see that Nikon Coolpix has a really bargain price, perfect size & weight; however, its view screen, battery and lens are not good enough. Cannon SX520's price is also great, it has a good resolution and battery, and its size & weight is acceptable; however, its shutter and view screen are all not satisfactory. Cannon Rebel XT's price, setup, shutter, battery and lens get more complains than the other two, and thus does not look like a good deal. Finally, Cannon SX400 has acceptable price and size & weight, but its setup, view screen, shutter, battery, and lens are its weaknesses. Among the four products, Nikon Coolpix and Cannon SX520 seem to be more desirable products than the other two because they have more strengths than weaknesses in terms of their product features.

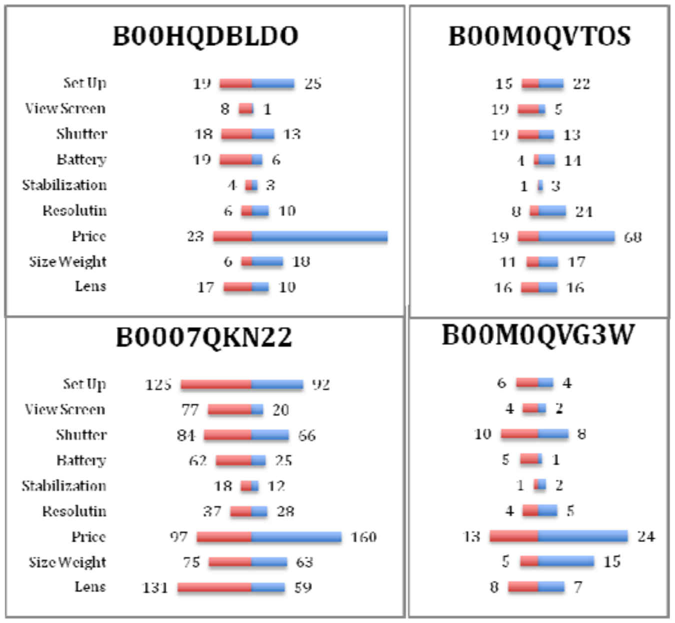


Figure 2. Review summaries of four selected cameras

VI. Conclusions and Future Work

In this paper, we introduced a sentence level topic model for evaluating and comparing online products based on their reviews. In our approach, we use product reviews as pieces of evidence to identify the strengths and weaknesses of a product in terms of its product features. To illustrate the feasibility and effectiveness of our approach, we retrieved product information of online products and their review comments from Amazon for review analysis. Our case study shows that our approach can achieve high accuracy with a training dataset of a reasonable size. As a major benefit of our approach, customers can greatly save their time for reading reviews and comparing similar products based on their product features.

In future research, we plan to develop our own sentiment analysis tool to enhance system performance. We will also study the impacts on performance when considering a sentence as a bag of independent words vs. related words. We will develop useful tools to allow users to help with labeling data points. Finally, we will further validate the feasibility of our approach using labeled datasets from different domains.

References

[1] R. Wei and H. Xu, "A Formal Cost-Effectiveness Analysis Model for Product Evaluation in E-Commerce," In Proceedings *of the 25th International Conference on Software Engineering and Knowledge Engineering (SEKE 2013)*, Boston, MA, USA, June 27-29, 2013, pp. 287-293.

[2] B. Pang and L. Lee, "A Sentimental Education: Sentiment Analysis Using Subjectivity Summarization Based on Minimum Cuts," In *Proceedings of the 42nd Annual Meeting on Association for Computational Linguistics (ACL)*, 2004, pp. 271-278.

[3] C. Whitelaw, N. Garg, and S. Argamon, "Using Appraisal Groups for Sentiment Analysis," In *Proceedings of the 14th ACM International Conference on Information and Knowledge Management (CIKM)*, 2005, pp. 625-631.

[4] C. H. Papadimitriou, P. Raghavan, H. Tamaki, and S. Vempala, "Latent Semantic Indexing: A Probabilistic Analysis," *Journal of Computer and System Sciences*, Vol. 61, No. 2, October 2000, pp. 217-235.

[5] D. M. Blei, A. Y. Ng, M. I. Jordan, "Latent Dirichlet Allocation," *Journal of Machine Learning Research*, Vol. 3, January 2003, pp. 993-1022.

[6] E. Boiy, P. Hens, K. Deschacht, and M.-F. Moens, "Automatic Sentiment Analysis in Online Text," In *Proceedings of the 11th International Conference on Electronic Publishing (ELPUB 2007)*, Vienna, Austria , June 13-15, 2007, pp. 349-360.

[7] J. Kamps, M. Marx, R. J. Mokken, and M. De Rijke, "Using WordNet to Measure Semantic Orientations of Adjectives," In *Proceedings of the 4th International Conference on Language Resources and Evaluation (LREC-04)*, May 2004, pp. 1115-1118.

[8] A. Esuli and F. Sebastiani, "SentiWordNet: A Publicly Available Lexical Resource for Opinion Mining," In *Proceedings of the 5th International Conference on Language Resources and Evaluation (LREC-06)*, 2006, pp. 417-422.

[9] C. D. Manning, M. Surdeanu, J. Bauer, J. Finkel, S. J. Bethard, and D. McClosky, "The Stanford CoreNLP Natural Language Processing Toolkit," In *Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, Baltimore, Maryland, USA, June 2014, pp. 55-60.

[10] Y R. Socher, A. Perelygin, J. Y. Wu *et al.*, "Recursive Deep Models for Semantic Compositionality Over a Sentiment Treebank," In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP 2013)*, Seattle, WA, USA, October 18-21, 2013, pp. 1631-1642.