PhraseRNN: Phrase Recursive Neural Network for Aspect-based Sentiment Analysis

Thien Hai Nguyen **Kiyoaki Shirai**
School of Information Science
Japan Advanced Institute of Science and Technology
1-1 Asahidai, Nomi, Ishikawa 923-1292, Japan
{nhthien, kshirai}@jaist.ac.jp

Abstract

This paper presents a new method to identify sentiment of an aspect of an entity. It is an extension of RNN (Recursive Neural Network) that takes both dependency and constituent trees of a sentence into account. Results of an experiment show that our method significantly outperforms previous methods.

1 Introduction

Aspect-based sentiment analysis (ABSA) has been found to play a significant role in many applications such as opinion mining on product or restaurant reviews. It is a task to determine an attitude, opinion and emotions of people toward aspects in a sentence. For example, given a sentence "Except the design, the phone is bad for me", the system should classify positive and negative as the sentiments for the aspects 'design' and 'phone', respectively.

The simple approach is to calculate a sentiment score of a given aspect as the weighted sum of opinion scores, which are defined by a sentiment lexicon, of all words in the sentence (Liu and Zhang, 2012; Pang and Lee, 2008). This method is further improved by identifying the aspect-opinion relations using tree kernel method (Nguyen and Shirai, 2015a).

Other researches have attempted to use unsupervised topic modeling methods. To identify the sentiment category of the aspect, topic models which can simultaneously exploit aspect and sentiment have been proposed, such as TSLDA (Nguyen and Shirai, 2015b), ASUM (Jo and Oh, 2011), JST (Lin and He, 2009) and FACTS model (Lakkaraju et al., 2011).

Recursive Neural Network (RNN) is a kind of deep neural network. Using distributed representations of words (aka word embedding) (Bengio et al., 2003; Hinton, 1986), RNN merges word representations to represent phrases or sentences. It is one of the best methods to predict sentiment labels for the phrases (Socher et al., 2011; Socher et al., 2012; Socher et al., 2013). AdaRNN (Adaptive Recursive Neural Network) is an extension of RNN for Twitter sentiment classification (Dong et al., 2014a; Dong et al., 2014b).

This paper proposes a new method PhraseRNN for ABSA. It is an extended model of RNN and AdaRNN, which are briefly introduced in Section 2. The basic idea is to make the representation of the target aspect richer by using syntactic information from both the dependency and constituent trees of the sentence.

2 Recursive Neural Networks for ABSA

In RNN and AdaRNN, given a sentence containing a target aspect, "binary dependency tree" is built from a dependency tree of the sentence. Intuitively, it represents syntactic relations associated with the aspect. Each word (leaf) or phrase (internal node) in the binary dependency tree is represented as a d-dimensional vector. From bottom to up, the representations of a parent node v is calculated by combination of left and right child vector representations (v_l and v_r) using a global function g in RNN:

$$g(v_l, v_r) = W \begin{bmatrix} v_l \\ v_r \end{bmatrix} + b \quad (1)$$

where $W \in \Re^{d \times 2d}$ is the composition matrix and $b \in \Re^d$ is the bias vector. Then $v = f(g(v_l, v_r))$ where f is a nonlinear function such as $tanh$.

Instead of using only a global function g, AdaRNN employed n compositional functions $G = \{g_1, \cdots, g_n\}$ and selected them depending on the linguistic tags and combined vectors as follows:

$$v = f\left(\sum_{i=1}^{n} P(g_i|v_l, v_r, e) g_i(v_l, v_r)\right) \quad (2)$$

Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pages 2509–2514,
Lisbon, Portugal, 17-21 September 2015.

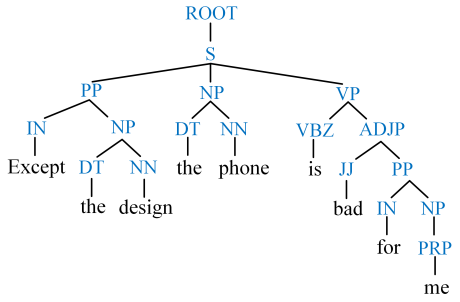

Figure 1: Example of a Constituent Tree

where $P(g_i|v_l, v_r, e)$ is the probability of function g_i given the child vectors v_l, v_r and external feature vector e. The probabilities are estimated as Equation (3).

$$\begin{bmatrix} P(g_1|v_l, v_r, e) \\ \cdots \\ P(g_n|v_l, v_r, e) \end{bmatrix} = softmax\left(\beta R \begin{bmatrix} v_l \\ v_r \\ e \end{bmatrix}\right) \quad (3)$$

where $\beta \in \Re$ is a hyper-parameter, and $R \in \Re^{n\times(2d+|e|)}$ is the parameter matrix.

The vector of the root node of the binary dependency tree is regarded as a representation of the target aspect. It is fed to a logistic regression to predict the sentiment category of the aspect.

3 PhraseRNN: Phrase Recursive Neural Network

In this model, a representation of an aspect will be obtained from a "target dependent binary phrase dependency tree" constructed by combining the constituent and dependency trees. In addition, instead of using a list of global functions G as in AdaRNN, two kinds of composition functions G in inner-phrase and H in outer-phrase are used.

3.1 Building Hierarchical Structure

First, the basic phrases (noun phrases, verb phrases, preposition phrases and so on) are extracted from the constituent tree of the sentence. For example, a list of phrases P = {PP[Except the design], NP[the phone], VP[is bad for me]} is extracted from the constituent tree in Figure 1.

Given a dependency tree and a list of phrases, a phrase dependency tree is created by Algorithm 1. The input is a dependency tree $T = (V, E)$ consisting of a set of vertices $V = \{v_1, \cdots, v_{|V|}\}$ and a set of relation edges $E = \{(r_{ji}, v_i, v_j)\}$ between two vertices, and a list of phrases $P = \{p_1, \cdots, p_K\}$ extracted from the constituent tree. The output is a phrase dependency tree $pT = (pV, pE)$ where $pV = \{T_1, \cdots, T_K\}$ ($T_i = (V_i, E_i)$ is a subtree) and $pE = \{(r_{ji}, T_i, T_j)\}$ (a set of relations between two subtrees). With the dependency tree and the phrase list in Figure 2(a), the algorithm will output a phrase dependency tree in Figure 2(b).

Algorithm 1: Convert to Phrase Dependency Tree

Input: dependency tree $T = (V, E)$, phrase list $P = \{p_1, \cdots, p_K\}$
Output: phrase dependency tree: $pT = (pV, pE)$ where $pV = \{T_1, \cdots, T_K\}, T_i = (V_i, E_i)$ and $pE = \{(r_{ji}, T_i, T_j)\}$

```
for each phrase p_i ∈ P do
    V_i ← {v_j | v_j ∈ p_i}
end
for each edge (r_nm, v_m, v_n) ∈ E do
    v_m ∈ p_k, v_n ∈ p_l
    if k = l then
        E_k ← E_k ∪ {(r_nm, v_m, v_n)}
    else
        pE ← pE ∪ {(r_nm, T_k, T_l)}
    end
end
```

The phrase dependency tree is transformed into a target dependent binary phrase dependency tree bpT by Algorithm 2. The input of the algorithm is a phrase dependency tree $pT = (pV, pE)$ and a target word v_t (the aspect word we want to predict the sentiment category). The output is the binary tree bpT. Note that the leaves of the binary tree bpT are binary subtrees $bT_1, \cdots, bT_K$ which are the binary versions of subtrees $T_1, \cdots, T_K$. On the other hand, the leaves of binary subtree bT_i are the words in phrase p_i. bpT and bT_i are obtained by *convert* function defined as Algorithm 3. It can convert an arbitrary tree to a binary tree [1]. Figure 2(c) and Figure 3 show the outputs for the aspect 'design' and 'phone', respectively.

3.2 Constructing the Aspect Representation

Each node in the binary tree is represented as a d-dimensional vector. In this research, we use the pre-trained Google News dataset [2] by word2vec algorithms (Mikolov et al., 2013). Each word is

[1] Note that *convert* function returns a tree represented by nested brackets such as [PP,[NP,VP]].

[2] https://code.google.com/p/word2vec/

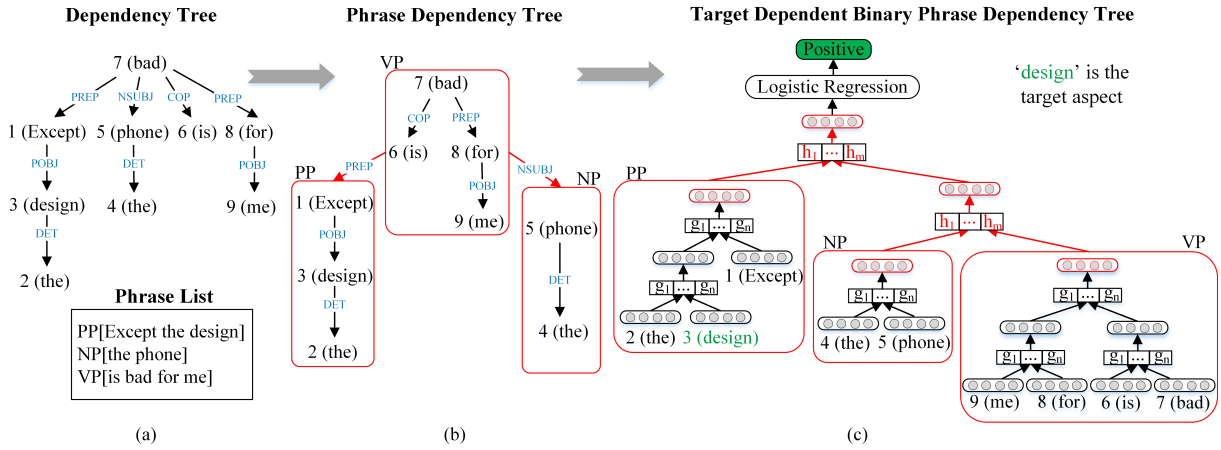

Figure 2: Hierarchical Structures in PhraseRNN: (a) Dependency Tree, (b) Phrase Dependency Tree and (c) Target Dependent Binary Phrase Dependency Tree

Algorithm 2: Convert to Target Dependent Binary Phrase Dependency Tree

Input: phrase dependency tree: $pT = (pV, pE)$, target v_t
Output: target dependent binary phrase dependency tree: bpT

for $T_i = (V_i, E_i) \in pV$ **do**
  if $v_t \in V_i$ **then**
    $h \leftarrow v_t$
  else
    $h \leftarrow$ vertex having no head in E_i
  end
  $bT_i \leftarrow convert(E_i, h)$
end
$T_{v_t} \leftarrow T_i$ that contains v_t
$bpT \leftarrow convert(pE, T_{v_t})$
Replace all T_i in bpT with bT_i

Algorithm 3: Convert to a Binary Tree

Function *convert*(E, v_t)**:**
  $v \leftarrow v_t$
  for $v_i \rightarrow v_t$, $v_t \rightarrow v_i$ *in* E **do**
    if $v_t \rightarrow v_i$ **then**
      $E' \leftarrow E \setminus \{v_t \rightarrow v_i\}$
      $w \leftarrow [convert(E, v_i), v]$
    else
      $E' \leftarrow E \setminus \{v_i \rightarrow v_t\}$
      $w \leftarrow [v, convert(E, v_i)]$
    end
    $v \leftarrow w$
  end
  return v
end

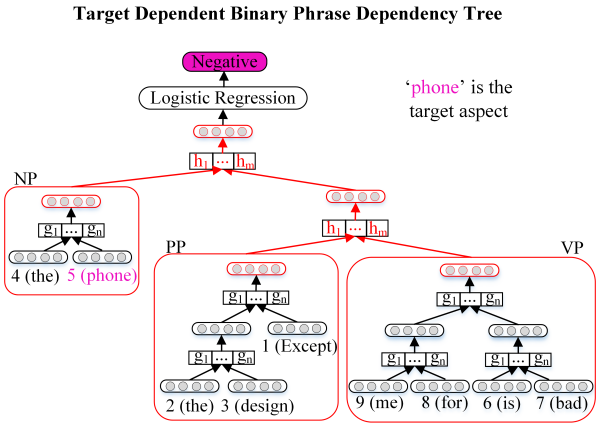

Figure 3: Another Target Dependent Binary Phrase Dependency Tree (Target Aspect 'phone')

represented as a 300-dimensional vector in this pre-trained dataset.

PhraseRNN uses two kinds of composition function $G = \{g_1, \cdots, g_n\}$ for inner-phrase and $H = \{h_1, \cdots, h_m\}$ for outer-phrase. n and m are the number of functions in G and H, respectively.

The vector of the parent node v_{in} in the binary subtree bT_i, where v_l and v_r are the vectors of the left and right children, is computed as:

$$v_{in} = f\left(\sum_{i=1}^{n} P(g_i|v_l, v_r, e_{in}) g_i(v_l, v_r)\right) \quad (4)$$

where e_{in} is the external feature vector. $P(g_i|v_l, v_r, e_{in})$ is the probability of function g_i given the child vectors v_l, v_r and e_{in}. It is

defined as Equation (5).

$$\begin{bmatrix} P(g_1|v_l, v_r, e_{in}) \\ \cdots \\ P(g_n|v_l, v_r, e_{in}) \end{bmatrix} = softmax\left(\beta R \begin{bmatrix} v_l \\ v_r \\ e_{in} \end{bmatrix}\right) \quad (5)$$

where $\beta \in \Re$ is a hyper-parameter, and $R \in \Re^{n\times(2d+|e_{in}|)}$ is the parameter matrix.

In the target dependent binary phrase dependency tree bpT, the vector of the parent node v_{out}, where the vectors of the left and right children are v_l and v_r, is computed as:

$$v_{out} = f\left(\sum_{i=1}^{m} P(h_i|v_l, v_r, e_{out}) h_i(v_l, v_r)\right) \quad (6)$$

$P(h_i|v_l, v_r, e_{out})$ is the probability of function h_i given the child vectors v_l, v_r and external feature vector e_{out} as shown in Equation (7).

$$\begin{bmatrix} P(h_1|v_l, v_r, e_{out}) \\ \cdots \\ P(h_m|v_l, v_r, e_{out}) \end{bmatrix} = softmax\left(\beta S \begin{bmatrix} v_l \\ v_r \\ e_{out} \end{bmatrix}\right) \quad (7)$$

where $S \in \Re^{m\times(2d+|e_{out}|)}$ is the parameter matrix.

The external features e_i (e_{in} and e_{out}) of the node v_i consists of three types of features: $Label_l$, $Label_r$ and $DepType_i$. $Label_l$ and $Label_r$ are the labels of the left and right child nodes, respectively. If node v_l is a leaf word, $Label_l$ is the POS of the word v_l. Otherwise, it is the non-terminal symbol of the lowest common parent of descendants of v_l in the constituent tree. For example, the $Label$ of the node combined from 'the' and 'design' in Figure 2(c) is 'NP' which is the lowest common parent of these two words in the constituent tree in Figure 1. $DepType_i$ is the dependency relation for node v_i. If the left and right children of v_i are leaf nodes, it is the direct relation in the dependency tree between them. Otherwise, $DepType_i$ is the relation between head words of the left and right nodes. For instance, in Figure 2(c), let a be the parent of 'is' and 'bad', b is the parent of 'for' and 'me', c is the parent of a and b. $DepType$ of a and b are 'COP' and 'POBJ' that are direct relations between two child nodes in the dependency tree in Figure 2(a). While, $DepType$ of c is 'PREP' that is the dependency relation between two head words 'bad' and 'for'. e_i is a binary vector where the weight of the vector represents the presence of each feature.

We suppose a batch training data consisting of B instances $\{(x^{(1)}, t^{(1)}), \cdots, (x^{(B)}, t^{(B)})\}$, where $x^{(b)}$ and $t^{(b)}$ are the aspect and its sentiment category of b-th instance. Let $y^{(b)}$ be the predicted sentiment category for aspect $x^{(b)}$ by PhraseRNN. The goal is to minimize the loss function which is the sum of the mean of negative log likelihood and L2 regularization penalty in a batch training set as in Equation (8).

$$L = -\frac{1}{B}\sum_{b=1}^{B} log(P(y^{(b)} = t^{(b)}|x^{(b)}, \theta)) + \lambda \sum_{\theta_i \in \theta} \| \theta_i \|^2 \quad (8)$$

where λ is a constant controlling the degree of penalty, θ is all the parameters in the model.

Stochastic gradient descent is used to optimize the loss function. Backpropagation is employed to propagate the errors from the top node to the leaf nodes. The derivatives of parameters are used to update the parameters.

4 Evaluation

We use the restaurant reviews dataset in SemEval2014 Task 4 consisting of over 3000 English sentences. For each aspect, "positive", "negative" or "neutral" is annotated as its polarity. Dataset is divided into three parts: 70% training, 10% development and 20% test.

We compare the following methods:

ASA w/o RE: It defines a sentiment score of a given aspect as the weighted sum of opinion scores of all words in the sentence, where the weight is defined by the distance from the aspect (Liu and Zhang, 2012; Pang and Lee, 2008).

ASA with RE: It improves "ASA w/o RE" by firstly identifying the aspect-opinion relations using tree kernel, then integrating them to the sentiment calculation (Nguyen and Shirai, 2015a).

RNN: It uses only one global function g_1 over the binary dependency tree.

AdaRNN: It uses multi-composition functions $G = \{g_1, \cdots, g_n\}$ over a binary dependency tree (Dong et al., 2014a).

PhraseRNN-1: our PhraseRNN with only one global function: $G = H = g_1$

PhraseRNN-2: our PhraseRNN with two global functions. One for inner-phrase, the other for outer-phrase: $G = g_1$ and $H = h_1$

PhraseRNN-3: our PhraseRNN with multiple global functions: $G = H = \{g_1, \cdots, g_n\}$

PhraseRNN-4: our PhraseRNN with two lists of global functions. One for inner-phrase, the other for outer-phrase: $G = \{g_1, \cdots, g_n\}$ and $H = \{h_1, \cdots, h_m\}$

Stanford CoreNLP (Manning et al., 2014) is used to parse the sentence and obtain constituent and dependency trees. For RNN, AdaRNN and PhraseRNN, the optimal parameters, which minimize the error in the development set, are used for the sentiment classification of the test set. We set $\beta = 1$ for AdaRNN and PhraseRNN since it is reported that $\beta = 1$ is the best parameter (Dong et al., 2014a). The optimized number of composition functions n and $m = \frac{n}{2}$ are selected by grid search with $n = \{2, 4, 6, 8, 10\}$ on the development set. $\lambda = 0.0001$ is employed. Accuracy (**A**), Precision (**P**), Recall (**R**) and F-measure (**F**) are used as evaluation metrics [3].

Table 1 shows the results of the methods. Differences of PhraseRNN and RNN are verified by statistical significance tests. We use the paired randomization test because it does not require additional assumption about distribution of outputs (Smucker et al., 2007). The results indicate that four variations of our PhraseRNN outperform "ASA w/o RE", "ASA with RE", RNN and AdaRNN methods from 5.35% to 19.44% accuracy and 8% to 16.48% F-measure. Among four variations, PhraseRNN-2 and PhraseRNN-3 achieved the best performance. By using different global functions in the inner and outer phrases, PhraseRNN-2 improves PhraseRNN-1 by 2.54% F-measure while keeping the comparable accuracy. Using multi-composition functions is also effective since PhraseRNN-3 was better than PhraseRNN-1 by 1.55% accuracy. PhraseRNN-4 improved PhraseRNN-3 by 6.38% precision while keeping comparable in other metrics.

Since our PhraseRNN-1 and PhraseRNN-3 outperform RNN and AdaRNN (the models relying on the binary dependency tree) respectively, we can conclude that our target dependent binary phrase dependency tree is much effective than binary dependency tree for ABSA.

In the data used in (Dong et al., 2014a), one sentence contains only one aspect. On the other hand, two or more aspects can be appeared in one sentence in SemEval 2014 data. It is common in the real text. To examine in which cases our method is better than the others, we conduct an additional experiment by dividing the test set into three disjoint subsets. The first subset (**S1**) contains sentences having only one aspect. The second subset (**S2**) and third subset (**S3**) have two or more aspects in each sentence. All aspects in a sentence in S2 have the same sentiment category, while different sentiment categories in S3. The number of aspects in S1, S2 and S3 are 200, 323 and 187, respectively.

[3] Precision, Recall and F-measure are the average for three polarity categories weighted by the number of true instances.

Table 1: Results of ABSA

Methods	A	P	R	F
ASA w/o RE	46.76	54.63	46.76	48.06
ASA with RE	52.39	53.91	52.39	52.54
RNN	60.85	53.59	60.85	54.21
AdaRNN	60.42	36.78	60.42	45.73
PhraseRNN-1	64.65†	58.59†	64.65†	59.67*
PhraseRNN-2	63.94†	**62.40***	63.94†	**62.21***
PhraseRNN-3	**66.20***	53.88	**66.20***	59.32*
PhraseRNN-4	65.92*	60.26†	65.92*	59.80*

Notes: Statistical significance test of PhraseRNN comparing to RNN.
* Significant at the 1 percent level.
† Significant at the 5 percent level.

Table 2: The Number of Correctly Identified Aspects in Subsets S1, S2 and S3

Methods	S1	S2	S3
ASA w/o RE	98 (49.00)	156 (48.30)	78 (41.71)
ASA with RE	111 (55.50)	176 (54.49)	85 (45.45)
RNN	123 (61.50)	226 (69.97)	83 (44.39)
AdaRNN	117 (58.50)	234 (72.45)	78 (41.71)
PhraseRNN-1	**129** (**64.50**)	248 (76.78)	82 (43.85)
PhraseRNN-2	125 (62.50)	247 (76.47)	82 (43.85)
PhraseRNN-3	125 (62.50)	**257** (**79.57**)	88 (47.06)
PhraseRNN-4	128 (64.00)	250 (77.40)	**90** (**48.13**)

Table 2 shows the number of aspects where their sentiments are correctly identified by the methods in the subsets S1, S2 and S3. The accuracies are also shown in parentheses. Among three subsets, S3 is the most difficult and ambiguous case. In all methods, the performance in S3 is worse than S1 and S2. Comparing with other methods in each subset, PhraseRNN improves the accuracy in S2 more than in S1 and S3.

5 Conclusion

We proposed PhraseRNN to identify the sentiment of the aspect in the sentence. Propagating the semantics through the binary dependency tree in RNN and AdaRNN could not be enough to represent the sentiment of the aspect. A new hierarchical structure was constructed by integrating the dependency relations and phrases. The results indicated that our PhraseRNN outperformed "ASA w/o RE", "ASA with RE", RNN and AdaRNN.

References

Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Janvin. 2003. A neural probabilistic language model. *The Journal of Machine Learning Research (JMLR)*, 3:1137–1155.

Li Dong, Furu Wei, Chuanqi Tan, Duyu Tang, Ming Zhou, and Ke Xu. 2014a. Adaptive recursive neural network for target-dependent twitter sentiment classification. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 49–54.

Li Dong, Furu Wei, Ming Zhou, and Ke Xu. 2014b. Adaptive multi-compositionality for recursive neural models with applications to sentiment analysis. In *Twenty-Eighth AAAI Conference on Artificial Intelligence (AAAI)*, pages 1537–1543.

Geoffrey E Hinton. 1986. Learning distributed representations of concepts. In *Proceedings of the eighth annual conference of the cognitive science society*, volume 1, page 12. Amherst, MA.

Yohan Jo and Alice H Oh. 2011. Aspect and sentiment unification model for online review analysis. In *Proceedings of the fourth ACM international conference on Web search and data mining*, pages 815–824. ACM.

Himabindu Lakkaraju, Chiranjib Bhattacharyya, Indrajit Bhattacharya, and Srujana Merugu. 2011. Exploiting coherence for the simultaneous discovery of latent facets and associated sentiments. In *Proceedings of the Eleventh SIAM International Conference on Data Mining (SDM)*, pages 498–509. SIAM / Omnipress.

Chenghua Lin and Yulan He. 2009. Joint sentiment/topic model for sentiment analysis. In *Proceedings of the 18th ACM conference on Information and knowledge management (CIKM)*, pages 375–384. ACM.

Bing Liu and Lei Zhang. 2012. A survey of opinion mining and sentiment analysis. In *Mining Text Data*, pages 415–463. Springer.

Christopher D. Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J. Bethard, and David McClosky. 2014. The Stanford CoreNLP natural language processing toolkit. In *Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations (ACL)*, pages 55–60.

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 (NIPS)*, pages 3111–3119.

Thien Hai Nguyen and Kiyoaki Shirai. 2015a. Aspect-based sentiment analysis using tree kernel based relation extraction. In Alexander Gelbukh, editor, *Computational Linguistics and Intelligent Text Processing (CICLing)*, volume 9042 of *Lecture Notes in Computer Science*, pages 114–125. Springer International Publishing.

Thien Hai Nguyen and Kiyoaki Shirai. 2015b. Topic modeling based sentiment analysis on social media for stock market prediction. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics (ACL), Volume 1: Long Papers*, pages 1354–1364. The Association for Computer Linguistics.

Bo Pang and Lillian Lee. 2008. Opinion mining and sentiment analysis. *Foundations and trends in information retrieval*, 2(1-2):1–135.

Mark D Smucker, James Allan, and Ben Carterette. 2007. A comparison of statistical significance tests for information retrieval evaluation. In *Proceedings of the sixteenth ACM conference on Conference on information and knowledge management (CIKM)*, pages 623–632. ACM.

Richard Socher, Cliff C Lin, Chris Manning, and Andrew Y Ng. 2011. Parsing natural scenes and natural language with recursive neural networks. In *Proceedings of the 28th international conference on machine learning (ICML)*, pages 129–136.

Richard Socher, Brody Huval, Christopher D Manning, and Andrew Y Ng. 2012. Semantic compositionality through recursive matrix-vector spaces. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 1201–1211. Association for Computational Linguistics.

Richard Socher, Alex Perelygin, Jean Y Wu, Jason Chuang, Christopher D Manning, Andrew Y Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the conference on empirical methods in natural language processing (EMNLP)*, volume 1631, page 1642. Citeseer.