Latent Variable Sentiment Grammar*

Liwen Zhang†, Kewei Tu†, Yue Zhang‡◇
†School of Information Science and Technology,ShanghaiTech University, Shanghai, China
‡Institute of Advanced Technology, Westlake Institute for Advanced Study, China
◇School of Engineering, Westlake University, Hangzhou, China
{zhanglw1,tukw}@shanghaitech.edu.cn
yuezhang@westlake.edu.cn

Abstract

Neural models have been investigated for sentiment classification over constituent trees. They learn phrase composition automatically by encoding tree structures but do not explicitly model sentiment composition, which requires to encode sentiment class labels. To this end, we investigate two formalisms with deep sentiment representations that capture sentiment subtype expressions by latent variables and Gaussian mixture vectors, respectively. Experiments on Stanford Sentiment Treebank (SST) show the effectiveness of sentiment grammar over vanilla neural encoders. Using ELMo embeddings, our method gives the best results on this benchmark.

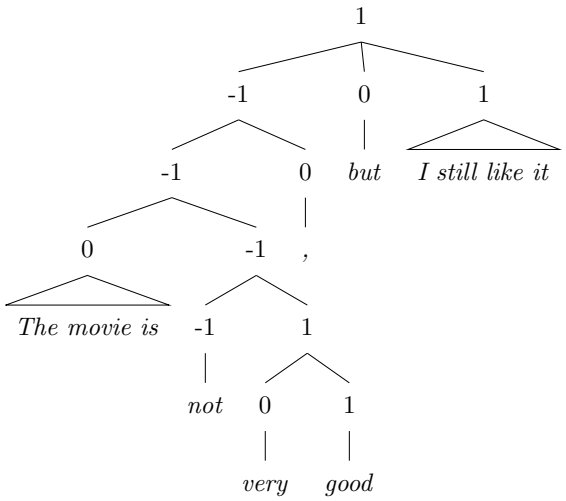


Figure 1: Example of sentiment composition

1 Introduction

Determining the sentiment polarity at or below the sentence level is an important task in natural language processing. Sequence structured models (Li et al., 2015; McCann et al., 2017) have been exploited for modeling each phrase independently. Recently, tree structured models (Zhu et al., 2015; Tai et al., 2015; Teng and Zhang, 2017) were leveraged for learning phrase compositions in sentence representation given the syntactic structure. Such models classify the sentiment over each constituent node according to its hidden vector through tree structure encoding.

Though effective, existing neural methods do not consider explicit *sentiment compositionality* (Montague, 1974). Take the sentence "The movie is not very good, but I still like it" in Figure 1 as example (Dong et al., 2015), over the constituent tree, sentiment signals can be propagated from leaf nodes to the root, going through negation, intensification and contrast according to the context. Modeling such signal channels can intuitively lead to more interpretable and reliable results. To model sentiment composition, direct encoding of sentiment signals (e.g., +1/-1 or more fine-grained forms) is necessary.

To this end, we consider a neural network grammar with latent variables. In particular, we employ a grammar as the backbone of our approach in which nonterminals represent sentiment signals and grammar rules specify sentiment compositions. In the simplest version of our approach, nonterminals are sentiment labels from SST directly, resulting in a weighted grammar. To model more fine-grained emotions (Ortony and Turner, 1990), we consider a latent variable grammar (LVG, Matsuzaki et al. (2005), Petrov et al. (2006)), which splits each nonterminal into subtypes to represent subtle sentiment signals and uses a discrete latent variable to denote the sentiment subtype of a phrase. Finally, inspired by the fact that sentiment can be modeled with a low dimensional continuous space (Mehrabian, 1980), we introduce a Gaussian mixture latent vector grammar (GM-LVeG, Zhao et al. (2018)), which

* Work was done when the first author was visiting Westlake University. The third author is the corresponding author.

associates each sentiment signal with a continuous vector instead of a discrete variable.

Experiments on SST show that explicit modeling of sentiment composition leads to significantly improved performance over standard tree encoding, and models that learn subtle emotions as hidden variables give better results than coarse-grained models. Using a bi-attentive classification network (Peters et al., 2018) as the encoder, out final model gives the best results on SST. To our knowledge, we are the first to consider neural network grammars with latent variables for sentiment composition. Our code will be released at https://github.com/Ehaschia/bi-tree-lstm-crf.

2 Related Work

Phrase-level sentiment analysis Li et al. (2015) and McCann et al. (2017) proposed sequence structured models that predict the sentiment polarities of the individual phrases in a sentence independently. Zhu et al. (2015), Le and Zuidema (2015), Tai et al. (2015) and Gupta and Zhang (2018) proposed Tree-LSTM models to capture bottom-up dependencies between constituents for sentiment analysis. In order to support information flow bidirectionally over trees, Teng and Zhang (2017) introduced a Bi-directional Tree-LSTM model that adds a top-down component after Tree-LSTM encoding. These models handle sentiment composition implicitly and predict sentiment polarities only based on embeddings of current nodes. In contrast, we model sentiment explicitly.

Sentiment composition Moilanen and Pulman (2007) introduced a seminal model for sentiment composition (Montague, 1974), composed positive, negative and neutral (+1/-1/0) singles hierarchically. Taboada et al. (2011) proposed a lexicon-based method for addressing sentence level contextual valence shifting phenomena such as negation and intensification. Choi and Cardie (2008) used a structured linear model to learn semantic compositionality relying on a set of manual features. Dong et al. (2015) developed a statistical parser to learn the sentiment structure of a sentence. Our method is similar in that grammars are used to model semantic compositionality. But we consider neural methods instead of statistical methods for sentiment composition. Teng et al. (2016) proposed a simple weighted-sum model of introducing sentiment lexicon features to LSTM for sentiment analysis. They used -2 to 2 represent sentiment polarities. In contrast, we model sentiment subtypes with latent variables and combine the strength of neural encoder and hierarchical sentiment composition.

Latent Variable Grammar There has been a line of work using discrete latent variables to enrich coarse-grained constituent labels in phrase-structure parsing (Johnson, 1998; Matsuzaki et al., 2005; Petrov et al., 2006; Petrov and Klein, 2007). Our work is similar in that discrete latent variables are used to model sentiment polarities. To our knowledge, we are the first to consider modeling fine-grained sentiment signals by investigating different types of latent variables. Recently, there has been work using continuous latent vectors for modeling syntactic categories (Zhao et al., 2018). We consider their grammar also in modeling sentiment polarities.

3 Baseline

We take the constituent Tree-LSTM as our baseline, which extends sequential LSTM to tree-structured network topologies. Formally, our model computes a parent representation from its two children in a Tree-LSTM:

$$\begin{bmatrix} \boldsymbol{i} \\ \boldsymbol{f}_l \\ \boldsymbol{f}_r \\ \boldsymbol{o} \\ \boldsymbol{g} \end{bmatrix} = \begin{bmatrix} \sigma \\ \sigma \\ \sigma \\ \sigma \\ \tanh \end{bmatrix} \left(\boldsymbol{W}_t \begin{bmatrix} \boldsymbol{x} \\ \boldsymbol{h}_l \\ \boldsymbol{h}_r \end{bmatrix} + \boldsymbol{b}_t \right) \tag{1}$$

$$\boldsymbol{c}_p = \boldsymbol{i} \otimes \boldsymbol{g} + \boldsymbol{f}_l \otimes \boldsymbol{c}_l + \boldsymbol{f}_r \otimes \boldsymbol{c}_r \tag{2}$$

$$\boldsymbol{h}_p = \boldsymbol{o} \otimes \tanh(\boldsymbol{c}_p) \tag{3}$$

where $\boldsymbol{W}_t \in \mathbb{R}^{5D_h \times 3D_h}$ and $\boldsymbol{b}_t \in \mathbb{R}^{3D_h}$ are trainable parameters, $\otimes$ is the Hadamard product and $\boldsymbol{x}$ represents the input of leaf node. Our formulation is a special case of the N-ary Tree-LSTM (Tai et al., 2015) with $N = 2$.

Existing work (Tai et al. (2015), Zhu et al. (2015)) performs softmax classification on each node according to the state vetcor $\boldsymbol{h}$ on each node for sentiment analysis. We follow this method in our baseline model.

4 Sentiment Grammars

We investigate sentiment grammars as a layer of structured representation on top of a tree-LSTM,

which model the correlation between different sentiment labels over a tree. Depending on how a sentiment label is represented, we develop three increasingly complex models. In particular, the first model, which is introduced in Section 4.1, uses a weighted grammar to model the first-order correlation between sentiment labels in a tree. It can be regarded as a PCFG model. The second model, which is introduced in Section 4.2, introduces a discrete latent variable for a refined representation of sentiment classes. Finally, the third model, which is introduced in Section 4.3, considers a continuous latent representation of sentiment classes.

4.1 Weighted Grammars

Formally, a sentiment grammar is defined as $\mathcal{G} = (N, S, \Sigma, R_t, R_e, W_t, W_e)$, where $N = \{A, B, C, ...\}$ is a finite set of sentiment polarities, $S \in N$ is the start symbol, Σ is a finite set of terminal symbols representing words such that $N \cap \Sigma = \varnothing$, R_t is the transition rule set containing production rules of the form $X \rightarrow \alpha$ where $X \in N$ and $\alpha \in N^+$; R_e is the emission rule set containing production rules of the form $X \rightarrow \boldsymbol{w}$ where $X \in N$ and $\boldsymbol{w} \in \Sigma^+$. W_t and W_e are sets of weights indexed by production rules in R_t and R_e, respectively. Different from standard formal grammars, for each sentiment polarity in a parse tree our sentiment grammar invokes one emission rule to generate a string of terminals and invokes zero or one transition rule to product its child sentiment polarities. This is similar to the behavior of hidden Markov models. Therefore, in a parse tree each non-leaf node is a sentiment polarity and is connected to exactly one leaf node which is a string of terminals. The terminals that are connected to the parent node can be obtained by concatenating the leaf nodes of its child nodes. Figure 2 shows an example for our sentiment grammar. In this paper, we only consider R_t in the Chomsky normal form (CNF) for clarity of presentation. However, it is straightforward to extend our formulation to the general case.

The score of a sentiment tree T conditioned on a sentence $\boldsymbol{w}$ is defined as follows:

$$S(T|\boldsymbol{w}, K) = \prod_{r_t \in T} W_n(r_t) \times \prod_{r_e \in T} W_e(r_e) \quad (4)$$

where r_t and r_e represent a transition rule and an emission rule in sentiment parse tree T, respectively. We specify the transition weights W_n with a non-negative rank-3 tensor. We compute

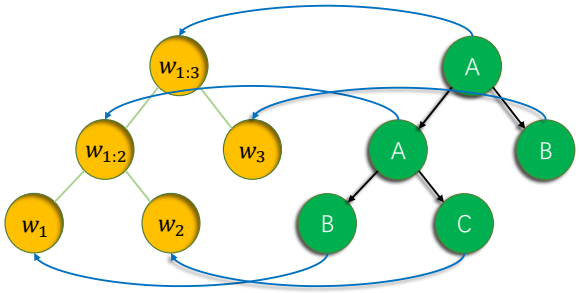


Figure 2: Sentiment grammar example. Here yellow nodes are leaf nodes of green constituent nodes, blue line $B \rightarrow w_1$ and black line $A \rightarrow BC$ represent an emission rule and a transition rule, respectively.

the non-negative weight of each emission rule $W_e(X \rightarrow \boldsymbol{w}_{i:j})$ by applying a single layer perceptron f_X and an exponential function to the neural encoder state vector $\boldsymbol{h}_{i:j}$ representing the constituent $\boldsymbol{w}_{i:j}$.

Sentiment grammars provides a principled way for explicitly modeling sentiment composition, and through parameterizing the emission rules with neural encoders, it can take the advantage of deep learning. In particular, by adding a weighted grammar on top of a tree-LSTM, our model is reminiscent of LSTM-CRF in the sequence structure.

4.2 Latent Variable Grammars

Inspired by categorical models (Ortony and Turner, 1990) which regard emotions as an overlay over a series of basic emotions, we extend our sentiment grammars with Latent Variable Grammars (LVGs; Petrov et al. (2006)), which refine each constituent tree node with a discrete latent variables, splitting each observed sentiment polarity into finite unobserved sentiment subtypes. We refer to trees over unsplit sentiment polarities as *unrefined trees* and trees over sentiment subtypes as *refined trees*.

Suppose that the sentiment polarities A, B and C of a transition rule $A \rightarrow BC$ are split into n_A, n_B and n_C subtypes, respectively. The weights of the refined transition rule can be represented by a non-negative rank-3 tensor $W_{A \rightarrow BC} \in \mathbb{R}^{n_A \times n_B \times n_C}$. Similarly, given an emission rule $A \rightarrow \boldsymbol{w}_{i:j}$, the weights of its refined rules by splitting A into n_A subtypes is a non-negative vector $W_{A \rightarrow \boldsymbol{w}_{i:j}} \in \mathbb{R}^{n_A}$ calculated by an exponential function and a single layer perceptron f_A:

$$W_{A \rightarrow \boldsymbol{w}_{i:j}} = \exp(f_A(\boldsymbol{h}_{i:j})) \quad (5)$$

where $\boldsymbol{h}_{i:j}$ is the vector representation of constituent $\boldsymbol{w}_{i:j}$. The score of a refined parse tree

is defined as the product of weights of all transition rules and emission rules that make up the refined parse tree, similar to Equation 4. The score of an unrefined parse tree is then defined as the sum of the scores of all refined trees that are consistent with it.

Note that Weighted Grammar (WG) can be viewed as a special case of LVGs where each sentiment polarity has one subtype.

4.3 Gaussian Mixture Latent Vector Grammars

Inspired by continuous models (Mehrabian, 1980) which model emotions in a continuous low dimensional space, we employ Latent Vector Grammars (LVeGs) (Zhao et al., 2018) that associate each sentiment polarity with a latent vector space representing the set of sentiment subtypes. We follow the idea of Gaussian Mixture LVeGs (GM-LVeGs) (Zhao et al., 2018), which uses Gaussian mixtures to model weight functions. Because Gaussian mixtures have the nice property of being closed under product, summation, and marginalization, learning and parsing can be done efficiently using dynamic programming

In GM-LVeG, the weight function of a transition or emission rule r is defined as a Gaussian mixture with K_r mixture components:

$$W_r(\boldsymbol{r}) = \sum_{k=1}^{K_r} \rho_{r,k} \mathcal{N}(\boldsymbol{r}|\boldsymbol{\mu}_{r,k}, \boldsymbol{\Sigma}_{r,k}) \tag{6}$$

where $\boldsymbol{r}$ is the concatenation of the latent vectors representing subtypes for sentiment polarities in rule r, $\rho_{r,k} > 0$ is the k-th mixing weight (the K_r mixture weights do not necessarily sum up to 1), and $\mathcal{N}(\boldsymbol{r}|\boldsymbol{\mu}_{r,k}, \boldsymbol{\Sigma}_{r,k})$ denotes the k-th Gaussian distribution parameterized by mean $\boldsymbol{\mu}_{r,k}$ and co-variance matrix $\boldsymbol{\Sigma}_{r,k}$. For an emission rule $A \rightarrow \boldsymbol{w}_{i:j}$, all the Gaussian mixture parameters are calculated by single layer perceptrons from the vector representation $\boldsymbol{h}_{i:j}$ of constituent $\boldsymbol{w}_{i:j}$:

$$\begin{aligned} \rho_{r,k} &= \exp(f_A^{\rho,k}(\boldsymbol{h}_{i:j})) \\ \boldsymbol{\mu}_{r,k} &= f_A^{\boldsymbol{\mu},k}(\boldsymbol{h}_{i:j}) \\ \boldsymbol{\Sigma}_{r,k} &= \exp(f_A^{\boldsymbol{\Sigma},k}(\boldsymbol{h}_{i:j})) \end{aligned} \tag{7}$$

For the sake of computational efficiency, we use Gaussian distributions with diagonal co-variance matrices.

4.4 Parsing

The goal of our task is to find the most probable sentiment parse tree T^*, given a sentence $\boldsymbol{w}$ and its constituency parse tree skeleton K. The polarity of the root node represents the polarity of the whole sentence, and the polarity of a constituent node is considered as the polarity of the phrase spanned by the node. Formally, T^* is defined as:

$$T^* = \underset{T \in G(\boldsymbol{w},K)}{\operatorname{argmax}} P(T|\boldsymbol{w}, K) \tag{8}$$

where $G(\boldsymbol{w}, K)$ denotes the set of unrefined sentiment parse trees for $\boldsymbol{w}$ with skeleton K. $P(T|w, K)$ is defined based on the parse tree score Equation 4:

$$P(T|\boldsymbol{w}, K) = \frac{S(T|\boldsymbol{w}, K)}{\sum_{\hat{T} \in K} S(\hat{T}|\boldsymbol{w}, K)}. \tag{9}$$

Note that unlike syntactic parsing, on SST we do not need to consider structural ambiguity, and thus resolving only rule ambiguity.

T^* can be found using dynamic programming such as the CYK algorithm for WG. However, parsing becomes intractable with LVGs and LVeGs since we have to compute the score of an unrefined parse tree by summing over all of its refined versions. We use the best performing max-rule-product decoding algorithm (Petrov et al., 2006; Petrov and Klein, 2007) for approximate parsing, which searches for the parse tree that maximizes the product of the posteriors (or expected counts) of unrefined rules in the parse tree. The detailed procedure is described below, which is based on the classic inside-outside algorithm.

For LVGs, we first use dynamic programming to recursively calculate the inside score function $s_{\mathbf{I}}^A(a, i, j)$ and outside score function $s_{\mathbf{O}}^A(a, i, j)$ for each sentiment polarity over each span $\boldsymbol{w}_{i:j}$ consistent with skeleton K using Equation 10 and Equation 11 in Table 1, respectively. Similarly for LVeGs, we recursively calculate inside score function $s_{\mathbf{I}}^A(\boldsymbol{a}, i, j)$ and outside score function $s_{\mathbf{O}}^A(\boldsymbol{a}, i, j)$ in LVeG are calculated by Equation 13 and Equation 14 in Table 1, in which we replace the sum of discrete variables in Equation 10-11 with the integral of continuous vectors. Next, using Equation 12 and Equation 15 in Table 1, we calculate the score $s(A \rightarrow BC, i, k, j)$ for LVG and LVeG, respectively, where $\langle A \rightarrow BC, i, k, j \rangle$ represents an anchored transition rule $A \rightarrow BC$ with A, B and C spanning phrase $\boldsymbol{w}_{i:j}$, $\boldsymbol{w}_{i,k-1}$ and

$$s^A_{\mathbf{I}}(a,i,j) = \sum_{A\to BC\in R_t} \sum_{b\in N}\sum_{c\in N} W_{A\to \boldsymbol{w}_{i:j}}(a)\ W_{A\to BC}(a,b,c) \times s^B_{\mathbf{I}}(b,i,k)\ s^C_{\mathbf{I}}(c,k+1,j) \quad (10)$$

$$s^A_{\mathbf{O}}(a,i,j) = \sum_{B\to CA\in R_t} \sum_{b\in N}\sum_{c\in N} W_{A\to \boldsymbol{w}_{i:j}}(a)\ W_{B\to CA}(b,c,a) \times s^B_{\mathbf{O}}(b,k,j)\ s^C_{\mathbf{I}}(c,k,i-1)$$

$$+ \sum_{B\to AC\in R_t} \sum_{b\in N}\sum_{c\in N} W_{A\to \boldsymbol{w}_{i:j}}(a)\ W_{B\to AC}(b,a,c) \times s^B_{\mathbf{O}}(b,i,k)\ s^C_{\mathbf{I}}(c,j+1,k) \quad (11)$$

$$s(A\to BC,i,k,j) = \sum_{a\in N}\sum_{b\in N}\sum_{c\in N} W_{A\to BC}(a,b,c) \times s^A_{\mathbf{O}}(a,i,j) \times s^B_{\mathbf{I}}(b,i,k) \times s^C_{\mathbf{I}}(c,k+1,j)\,. \quad (12)$$

$$s^A_{\mathbf{I}}(\boldsymbol{a},i,j) = \sum_{A\to BC\in R_t} \iint W_{A\to \boldsymbol{w}_{i:j}}(\boldsymbol{a}) W_{A\to BC}(\boldsymbol{a},\boldsymbol{b},\boldsymbol{c}) \times s^B_{\mathbf{I}}(\boldsymbol{b},i,k) s^C_{\mathbf{I}}(\boldsymbol{c},k+1,j)\, d\boldsymbol{b} d\boldsymbol{c} \quad (13)$$

$$s^A_{\mathbf{O}}(\boldsymbol{a},i,j) = \sum_{B\to CA\in R_t} \iint W_{A\to \boldsymbol{w}_{i:j}}(\boldsymbol{a}) W_{B\to CA}(\boldsymbol{b},\boldsymbol{c},\boldsymbol{a}) \times s^B_{\mathbf{O}}(\boldsymbol{b},k,j) s^C_{\mathbf{I}}(\boldsymbol{c},k,i-1)\, d\boldsymbol{b} d\boldsymbol{c}$$

$$+ \sum_{B\to AC\in R_t} \iint W_{A\to \boldsymbol{w}_{i:j}}(\boldsymbol{a}) W_{B\to AC}(\boldsymbol{b},\boldsymbol{a},\boldsymbol{c}) \times s^B_{\mathbf{O}}(\boldsymbol{b},i,k) s^C_{\mathbf{I}}(\boldsymbol{c},j+1,k)\, d\boldsymbol{b} d\boldsymbol{c} \quad (14)$$

$$s(A\to BC,i,k,j) = \iiint W_{A\to BC}(\boldsymbol{a},\boldsymbol{b},\boldsymbol{c}) \times s^A_{\mathbf{O}}(\boldsymbol{a},i,j) \times s^B_{\mathbf{I}}(\boldsymbol{b},i,k) \times s^C_{\mathbf{I}}(\boldsymbol{c},k+1,j)\, d\boldsymbol{a} d\boldsymbol{b} d\boldsymbol{c} \quad (15)$$

Table 1: Equation 10-12 calculate the inside score, outside score and production rule score for LVG, respectively. Equation 13-15 is used for LVeG. Equation 10 and Equation 13 are the inside score functions of a sentiment polarity A over its span $\boldsymbol{w}_{i:j}$ in the sentence $\boldsymbol{w}_{1:n}$. Equation 11 and Equation 14 are the outside score functions of a sentiment polarity A over a span $\boldsymbol{w}_{i:j}$ in the sentence $\boldsymbol{w}_{1:n}$. Equation 12 and Equation 15: the production rule score function of a rule $A \to BC$ with sentiment polarities A, B, and C spanning words $\boldsymbol{w}_{i:j}$, $\boldsymbol{w}_{i,k-1}$, and $\boldsymbol{w}_{k+1:j}$ respectively. Here we use lower case letters a, b, $c\ldots$ represent discrete subtypes of sentiment polarities A, B, $C\ldots$ in LVG and use bold lower case letters $\boldsymbol{a}$, $\boldsymbol{b}$, $\boldsymbol{c}\ldots$ represent continuous subtypes of sentiment polarities in LVeG. Note that spans such as $\boldsymbol{w}_{i:j}$ mentioned above are all given by the skeleton K of sentence $\boldsymbol{w}_{1:n}$.

$\boldsymbol{w}_{k+1:j}$ (all being consistent with skeleton K), respectively. The posterior (or expected count) of $\langle A\to BC,i,k,j\rangle$ can be calculate as follows:

$$q(A\to BC,i,k,j) = \frac{s(A\to,BC,i,k,j)}{s_{\mathbf{I}}(S,1,n)}, \quad (16)$$

where $s_{\mathbf{I}}(S,1,n)$ is the inside score for the start symbol S over the whole sentence $\boldsymbol{w}_{1:n}$. Then we can run CYK algorithm to identify the parse tree that maximizes the product of rule posteriors. It's objective function is given by:

$$T^*_q = \operatorname*{argmax}_{T_q\in G(\boldsymbol{w},K)} \prod_{e\in T} q(e) \quad (17)$$

where e ranges over all the transition rules in the sentiment parse tree T.

Note that the equations in Table 1 are tailored for our sentiment grammars and differ from their standard versions in two aspects. First, we take into account the additional emission rules in the inside and outside computation; second, the parse tree skeleton is assumed given and hence the split point k is prefixed in all the equations.

4.5 Learning

Given a training dataset $D = \{T_i, \boldsymbol{w}_i, K_i | i = 1\ldots m\}$ containing m samples, where T_i is the gold sentiment parse tree for the sentence $\boldsymbol{w}_i$ with its corresponding gold tree skeleton K_i. The discriminative learning objective is to minimize the negative log conditional likelihood:

$$\mathcal{L}(\Theta) = -\log \prod_{i=1}^{m} P_\Theta(T_i|\boldsymbol{w}_i, K_i)\,, \quad (18)$$

where Θ represents the set of trainable parameters of our models. We optimize the objective with gradient-based methods. In particular, gradients are first calculated over the sentiment grammar layer,

before being back-propagated to the tree-LSTM layer.

The gradient computation for the three models involves computing expected counts of rules, which has been described in Section 4.4. For WG and LVG, the derivative of W_r, the parameter of an unrefined production rule r is:

$$\frac{\partial \mathcal{L}(\Theta)}{\partial W_r} = \sum_{i=1}^{m} (\mathbb{E}_\Theta[f_r(t)|T_i] - \mathbb{E}_\Theta[f_r(t)|\boldsymbol{w}_i]) \quad (19)$$

where $\mathbb{E}_\Theta[f_r(t)|T_i]$ denotes the expected count of the unrefined production rule r with respect to P_Θ in the set of refined trees t, which are consistent with the observed parse tree T. Similarly, we use $\mathbb{E}_\Theta[f_r(t)|\boldsymbol{w}_i]$ for the expectation over all derivations of the sentence $\boldsymbol{w}_i$.

For LVeG, the derivative with respect to Θ_r, the parameters of the weight function $W_r(\boldsymbol{r})$ of an unrefined production rule r is:

$$\frac{\partial \mathcal{L}(\Theta)}{\partial \Theta_r} = \sum_{i=1}^{m} \int \left(\frac{\partial W_r(\boldsymbol{r})}{\partial \Theta_r} \times \frac{\mathbb{E}_\Theta[f_r(t)|\boldsymbol{w}_i] - \mathbb{E}_\Theta[f_r(t)|T_i]}{W_r(\boldsymbol{r})} \right) d\boldsymbol{r} \,. \quad (20)$$

The two expectations in Equation 19 and 20 can be efficiently computed using the inside-outside algorithm in Table 1. The derivative of the parameters of neural encoder can be derived from the derivative of the parameters of the emission rules.

5 Experiments

To investigate the effectiveness of modeling sentiment composition explicitly and using discrete variables or continuous vectors to model sentiment subtypes, we compare standard constituent Tree-LSTM (ConTree) with our models ConTree+WG, ConTree+LVG and ConTree+LVeG, respectively. To show the universality of our approaches, we also experiment with the combination of a state-of-the-art sequence structured model, bi-attentive classification network (BCN, Peters et al. (2018)), with our model: BCN+WG, BCN+LVG and BCN+LVeG.

5.1 Data

We use Stanford Sentiment TreeBank (SST, Socher et al. (2013)) for our experiments. Each constituent node in a phrase-structured tree is manually assigned an integer sentiment polarity from 0 to 4, which correspond to five sentiment classes: very negative, negative, neutral, positive and very positive, respectively. The root label represents the sentiment label of the whole sentence. The constituent node label represents the sentiment label of the phrase it spans. We perform both binary classification (-1, 1) and fine-grained classification (0-4), called SST-2 and SST-5, respectively. Following previous work, we use the labels of all phrases and gold-standard tree structures for training and testing. For binary classification, we merge all positive labels and negative labels.

5.2 Experimental Settings

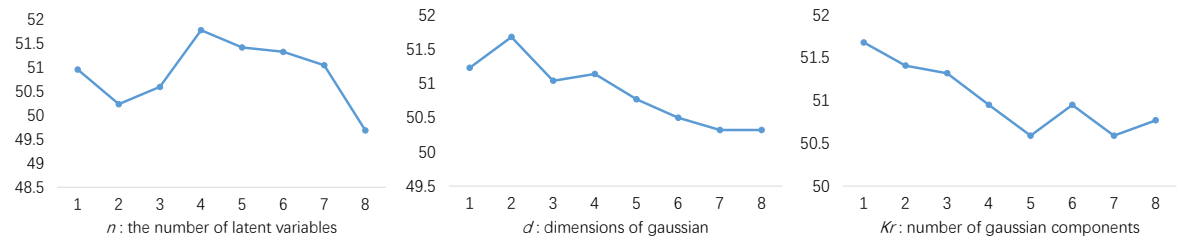
Hyper-parameters For ConTree, word vectors are initialized using Glove (Pennington et al., 2014) 300-dimensional embeddings and are updated together with other parameters. We set the hidden size of hidden units is 300. Adam (Kingma and Ba, 2014) is used to optimize the parameters with learning rate is 0.001. We adopt Dropout after the Embedding layer with a probability of 0.5. The sentence level mini-batch size is 32. For BCN experiment, we follow the model setting in McCann et al. (2017) except the sentence level mini-batch is set to 8.

5.3 Development Experiments

We use the SST development dataset to investigate different configurations of our latent variables and Gaussian mixtures. The best performing parameters on the development set are used in all following experiments.

LVG subtype numbers To explore the suitable number of latent variables to model subtypes of a sentiment polarity, we evaluate our ConTree+LVG model with different number of latent variables from 1 to 8. Figure 6(a) shows that there is an upward trend while the number of hidden variables n increases from 1 to 4. After reaching the peak when $n = 4$, the accuracy decreases as the number of latent variable continue to increase. We thus choose $n = 4$ for remaining experiments.

LVeG Gaussian dimensions We investigate the influence of the latent vector dimension on the accuracy for ConTree+LVeG. The component number of Gaussian mixtures is fixed to 1, Figure 6(b) illuminates that as the dimension increases from 1 to 8, there is a rise of accuracy from 1 to 2, followed by a decrease from 2 to 8. Thus we set the Gaussian dimension to 2 for remaining experiments.



(a) (b) (c)

Figure 6: Sentence level accuraicies on the development dataset. Figure (a) shows the performance of ConTree+LVG with different latent variables. Figure (b) and Figure (c) show the performance of ConTree+LVeG with different Gaussian dims and Gaussian mixture component numbers, respectively.

Model	SST-5 Root	SST-5 Phrase	SST-2 Root	SST-2 Phrase
ConTree (Le and Zuidema, 2015)	49.9	-	88.0	-
ConTree (Tai et al., 2015)	51.0	-	88.0	-
ConTree (Zhu et al., 2015)	50.1	-	-	-
ConTree (Li et al., 2015)	50.4	**83.4**	86.7	-
ConTree (Our implementation)	51.5	82.8	89.4	86.9
ConTree + WG	51.7	83.0	89.7	88.9
ConTree + LVG4	52.2	83.2	**89.8**	89.1
ConTree + LVeG	**52.9**	**83.4**	**89.8**	**89.5**

Table 2: Experimental results with constituent Tree-LSTMs.

Model	SST-5		SST-2	
	Root	Phrase	Root	Phrase
BCN(P)	54.7	-	-	-
BCN(O)	54.6	83.3	91.4	88.8
BCN+WG	55.1	**83.5**	91.5	90.5
BCN+LVG4	55.5	**83.5**	91.7	91.3
BCN+LVeG	**56.0**	**83.5**	**92.1**	**91.6**

Table 3: Experimental results with ELMo. BCN(P) is the BCN implemented by Peters et al. (2018). BCN(O) is the BCN implemented by ourselves.

LVeG Gaussian mixture component numbers Future 6(c) shows the performance of different component numbers with fixing the Gaussian dimension to 2. With the increase of Gaussian component number, the fine-grained sentence level accuracy declines slowly. The best performance is obtained when the component number $K_r = 1$, which we choose for remaining experiments.

5.4 Main Results

We re-implement constituent Tree-LSTM (ConTree) of Tai et al. (2015) and obtain better results than their original implementation. We then integrate ConTree with Weighted Grammars (ConTree+WG), Latent Variable Grammars with a subtype number of 4 (ConTree+LVG4), and Latent Variable Grammars (ConTree+LVeG), respectively. Table 2 shows the experimental results for sentiment classification on both SST-5 and SST-2 at the sentence level (Root) and all nodes (Phrase).

The performance improvement of ConTree+WG over ConTree reflects the benefit of handling sentiment composition explicitly. Particularly the phrase level binary classification task, ConTree+WG improves the accuracy by 2 points.

Compared with ConTree+WG, ConTree+LVG4 improves the fine-grained sentence level accuracy by 0.5 point, which demonstrates the effectiveness of modeling the sentiment subtypes with discrete variables. Similarly, incorporating Latent Vector Grammar into the constituent Tree-LSTM, the performance improvements, especially on the sentence level SST-5, demonstrate the effectiveness of modeling sentiment subtypes with continuous vectors. The performance improvements of ConTree+LVeG over ConTree+LVG4 show the advantage of infinite subtypes over finite subtypes.

There has also been work using large-scale external datasets to improve performances of sentiment classification. Peters et al. (2018) combined bi-attentive classification network (BCN, McCann et al. (2017)) with a pretrained language model

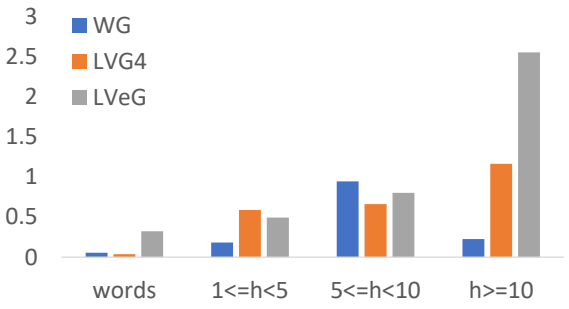


Figure 7: Changes in phrase level 5-class accuracies of our methods over ConTree.

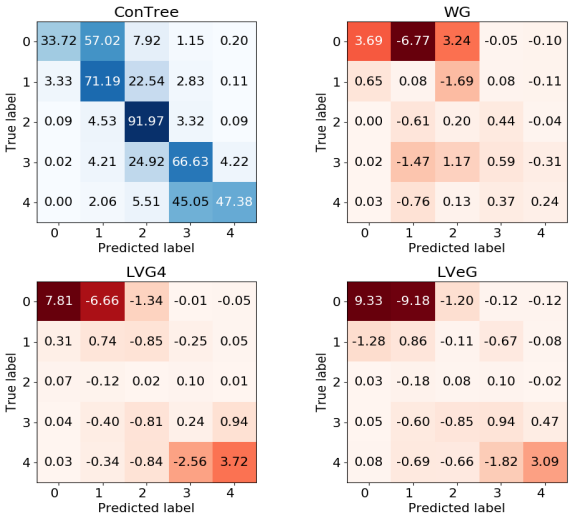


Figure 8: Top left is the normalized confusion matrix on the 5-class phrase level test dataset for ConTree. The others are the performance changs of our model over ConTree. Value in each cell is written with a unit of $\times 10^{-2}$

with character convolutions on a large-scale corpus (ELMo) and reported an accuracy of 54.7 on sentence-level SST-5. For fair comparison, we also augment our model with ELMo. Table 3 shows that our methods beat the baseline on every task. BCN+WG improves accuracies on all task slightly by modeling sentiment composition explicitly. The obvious promotion of BCN+LVG4 and BCN+LVeG shows that explicitly modeling sentiment composition with fine-grained sentiment subtypes is useful. Particularly, BCN+LVeG improves the sentence level classification accurracies by 1.4 points (fine-grained) and 0.7 points (binary) compared to BCN (our implementation), respectively. To our knowledge, we achieve the best results on the SST dataset.

5.5 Analysis

We make further analysis of our methods based on the constituent Tree-LSTM model. In the following, using WG, LVG and LVeG denote our three methods, respectively.

Impact on words and phrases Figure 7 shows the accuracy improvements over ConTree on phrases of different heights. Here the height h of a phrase in parse tree is defined as the distance between its corresponding constituent node and the deepest leaf node in its subtree. The improvement of our methods on word nodes, whose height is 0, is small because neural networks and word embeddings can already capture the emotion of words. In fact, the accuracy of ConTree on word nodes reaches 98.1%. As the height increases, the performance of our methods increase, expect for the accuracies of WG when $h \geq 10$ since the coarse-grained sentiment representation is far difficulty for handling too many sentiment compositions over the tree structure. The performance improvements of LVG4 and LVeG when $h \geq 10$ show modeling fine-grained sentiment signals can represent sentiment of higher phrases better.

Impact on sentiment polarities Figure 8 shows the performance changes of our models over ConTree on different sentiment polarities. The accuracy of every sentiment polarity on WG over ConTree improves slightly. Compared with ConTree, the accuracies of LVG4 and LVeG on extreme sentiments (the strong negative and strong positive sentiments) receive significant improvement. In addition, the proportion of extreme emotions misclassified as weak emotions (the negative and positive sentiments) drops dramatically. It indicates that LVG4 and LVeG can capture the subtle difference between extreme sentiments and weak sentiments by modeling sentiment subtypes explicitly.

Visualization of sentiment subtypes To investigate whether our LVeG can accurately model different emotional subtypes, we visualize all the strong negative sentiment phrases with length below 6 that are classified correctly in a 2D space. Since in LVeG, 2-dimension 1-component Gaussian mixtures are used to model a distribution over subtypes of a specific sentiment of phrases, we directly represent phrases by their Gaussian means $\boldsymbol{\mu}$. From Figure 9, we see that boring emotions such as "Extremely boring" and "boring" (green dots) are located at the bottom left, stupid emotions such as

Figure 9: Visualization of correlation of phrases not longer than 5 in the strong negative sentiment space

"stupider" and "Ridiculous" (red dots) are mainly located at the top right and negative emotions with no special emotional tendency such as "hate" and "bad" (blue dots) are evenly distributed throughout the space. This demonstrates that LVeG can capture sentiment subtypes.

6 Conclusion

We presented a range of sentiment grammars for using neural networks to model sentiment composition explicitly, and empirically showed that explicit modeling of sentiment composition with fine-grained sentiment subtypes gives better performance compared to state-of-the-art neural network models in sentiment analysis. By using EMLo embeddings, our final model improves fine-grained accuracies by 1.3 points compare to the current best result.

Acknowledgments

This work was supported by the Major Program of Science and Technology Commission Shanghai Municipal (17JC1404102) and NSFC (No. 61572245) . We would like to thank the anonymous reviewers for their careful reading and useful comments.

References

Yejin Choi and Claire Cardie. 2008. Learning with compositional semantics as structural inference for subsentential sentiment analysis. In *Proceedings of the conference on Empirical Methods in Natural Language Processing*, pages 793–801. Association for Computational Linguistics.

Li Dong, Furu Wei, Shujie Liu, Ming Zhou, and Ke Xu. 2015. A statistical parsing framework for sentiment classification. *Computational Linguistics*, pages 265–308.

Amulya Gupta and Zhu Zhang. 2018. To attend or not to attend: A case study on syntactic structures for semantic relatedness. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics*, pages 2116–2125. Association for Computational Linguistics.

Mark Johnson. 1998. Pcfg models of linguistic tree representations. *Computational Linguistics*, pages 613–632.

Diederik Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *International Conference on Learning Representations*.

Phong Le and Willem Zuidema. 2015. Compositional distributional semantics with long short term memory. In *Proceedings of the Fourth Joint Conference on Lexical and Computational Semantics*, pages 10–19. Association for Computational Linguistics.

Jiwei Li, Thang Luong, Dan Jurafsky, and Eduard Hovy. 2015. When are tree structures necessary for deep learning of representations? In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 2304–2314. Association for Computational Linguistics.

Takuya Matsuzaki, Yusuke Miyao, and Jun'ichi Tsujii. 2005. Probabilistic CFG with latent annotations. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics*, pages 75–82, Ann Arbor, Michigan. Association for Computational Linguistics.

Bryan McCann, James Bradbury, Caiming Xiong, and Richard Socher. 2017. Learned in translation: Contextualized word vectors. In *Advances in Neural Information Processing Systems*, pages 6294–6305.

Albert Mehrabian. 1980. *Basic dimensions for a general psychological theory: Implications for personality, social, environmental, and developmental studies*. Oelgeschlager, Gunn & Hain Cambridge, MA.

Karo Moilanen and Stephen Pulman. 2007. Sentiment composition. In *Proceedings of Recent Advances in Natural Language Processing*, pages 378–382.

Richard Montague. 1974. *Formal Philosophy; Selected Papers of Richard Montague*. New Haven: Yale University Press.

Andrew Ortony and Terence J. Turner. 1990. What's basic about basic emotions? *Psychological Review*, 97(3):315–331.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on Empirical Methods in Natural Language Processing*, pages 1532–1543. Association for Computational Linguistics.

Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2227–2237. Association for Computational Linguistics.

Slav Petrov, Leon Barrett, Romain Thibaux, and Dan Klein. 2006. Learning accurate, compact, and interpretable tree annotation. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics*, pages 433–440. Association for Computational Linguistics.

Slav Petrov and Dan Klein. 2007. Improved inference for unlexicalized parsing. In *Human Language Technologies 2007: The Conference of the North American Chapter of the Association for Computational Linguistics; Proceedings of the Main Conference*, pages 404–411. Association for Computational Linguistics.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D Manning, Andrew Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 conference on Empirical Methods in Natural Language Processing*, pages 1631–1642. Association for Computational Linguistics.

Maite Taboada, Julian Brooke, Milan Tofiloski, Kimberly Voll, and Manfred Stede. 2011. Lexicon-based methods for sentiment analysis. *Computational linguistics*, pages 267–307.

Kai Sheng Tai, Richard Socher, and Christopher D. Manning. 2015. Improved semantic representations from tree-structured long short-term memory networks. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing*, pages 1556–1566. Association for Computational Linguistics.

Zhiyang Teng, Duy Tin Vo, and Yue Zhang. 2016. Context-sensitive lexicon features for neural sentiment analysis. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1629–1638. Association for Computational Linguistics.

Zhiyang Teng and Yue Zhang. 2017. Head-lexicalized bidirectional tree lstms. *Transactions of the Association for Computational Linguistics*, 5:163–177.

Yanpeng Zhao, Liwen Zhang, and Kewei Tu. 2018. Gaussian mixture latent vector grammars. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics*, pages 1181–1189. Association for Computational Linguistics.

Xiaodan Zhu, Parinaz Sobihani, and Hongyu Guo. 2015. Long short-term memory over recursive structures. In *International Conference on Machine Learning*, pages 1604–1612.