Code

# Latent Variable Sentiment Grammar

Liwen Zhang[1] Kewei Tu[1] Yue Zhang[2,3]

[1]School of Information Science and Technology, ShanghaiTech University
[2]Institute of Advanced Technology, Westlake Institute for Advanced Study, China
[3]School of Engineering, Westlake University, Hangzhou, China

Paper

## Motivation

Though effective, existing neural network based sentiment analysis methods do not consider explicit sentiment compositionality.

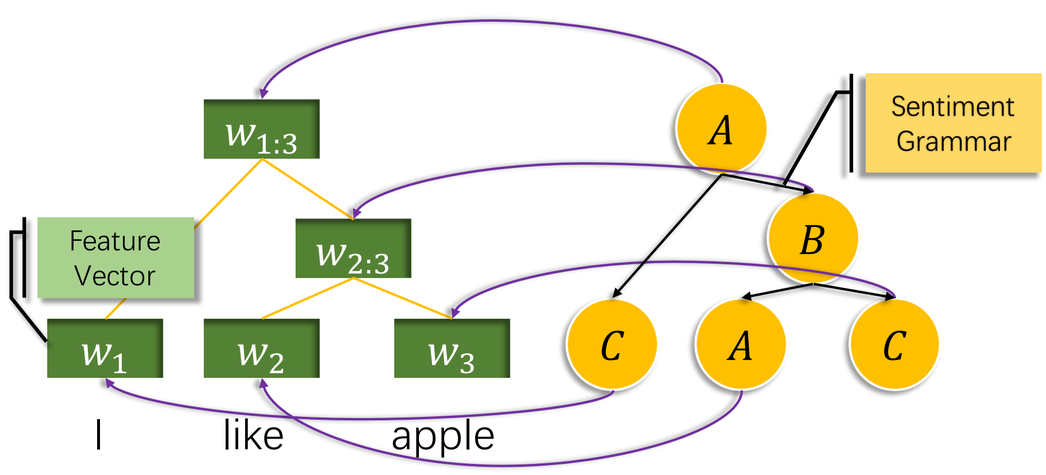
## Proposed Solution

We consider a neural network grammar with latent variables in which nonterminals represent sentiment signals and grammar rules specify sentiment compositions.

## Empirical Results

By using EMLo embeddings, our final model improves fine-grained accuracies by 1.3 points compared to the current best result on Stanford Sentiment TreeBank.

## Weighted Sentiment Grammar



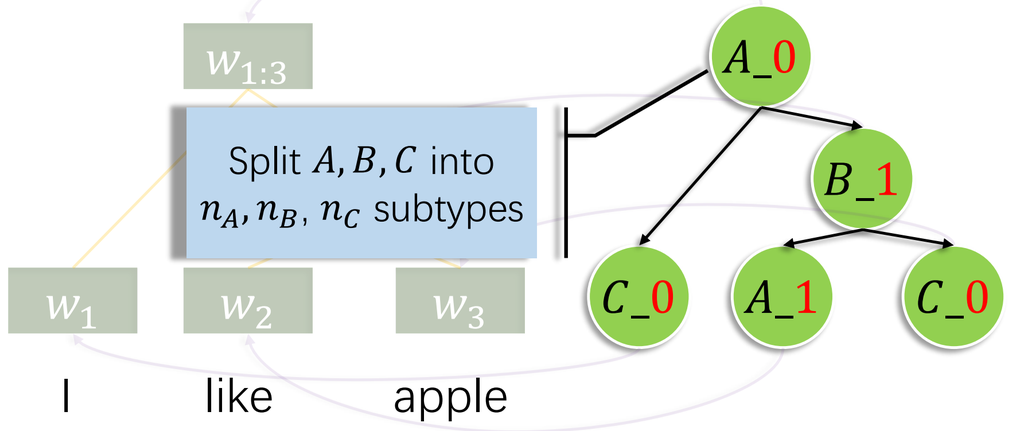

- $A, B, C$ are sentiment polarities.
- Black arrows are transition rules, their weights are trainable positive scores:

$$W(A \to CB) = 1.5$$

- Purple arrows are emission rules, their weights are positive scores computed by a feed forward network based on the phrase feature:

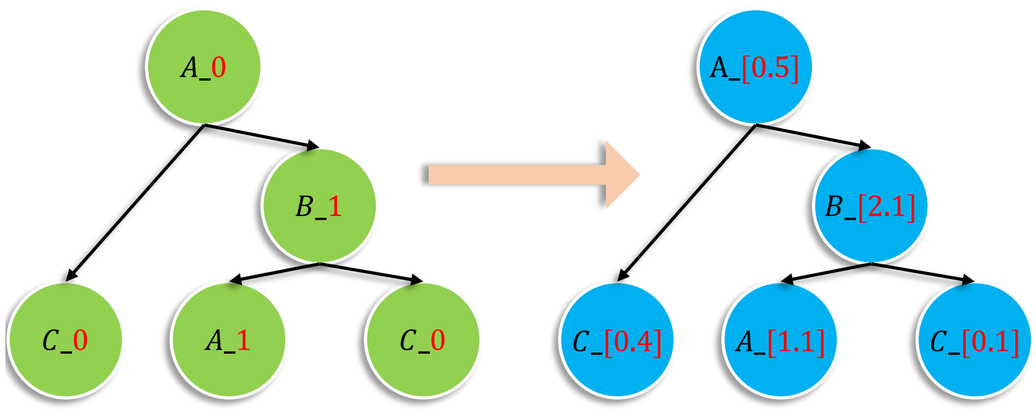
$$W(B \to w_{2:3}) = \exp\left(FFN(w_{2:3})\right) = 0.8$$

## Latent Variable Sentiment Grammar




We refine each sentiment with a discrete latent variable, splitting each observed sentiment polarity into finite unobserved sentiment subtypes (Petrov et al. 2006).
The weights of the refined transition rule $A \to BC$ can be represented by a non-negative rank-3 tensor: $W(A \to BA) \in \mathbb{R}^{n_A \times n_B \times n_C}$
The weights of refined rule $A \to w_{2:3}$ is a non-negative vector:
$W(A \to w_{2:3}) \in \mathbb{R}^{n_A}$ calculated by an exponential function and a single layer perceptron: $W_{A \to w_{2:3}} = \exp(FFN(w_{2:3}))$

## Latent Vector Sentiment Grammar




We also associate each sentiment polarity with a latent vector space representing the infinite set of sentiment subtypes (Zhao et al. 2018) . Here we use Gaussian Mixture to model weight functions:

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

here $\boldsymbol{r}$ is the latent vector representing subtype for sentiment rule $r$.

## 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 $K$.

$$T^* = \text{argmax}_{T \in K} P(T | \boldsymbol{w}, K) = \text{argmax} \frac{S(T|\boldsymbol{w},K)}{\sum_{\hat{T} \in K} S(\hat{T}|\boldsymbol{w},K)}$$

**Weighted Sentiment Grammar**: $T$ can be found using dynamic programming such as the CYK algorithm.
**Latent Variable / Vector Grammar**: Parsing is intractable. Therefore we use the max-rule-product decoding algorithm for approximate parsing:

- Calculate the inside score function $s^A_{\mathbf{I}}(a,i,j)$ and outside score function $s^A_{\mathbf{O}}(a,i,j)$ for each sentiment polarity over each span $_{i:j}$.
- Calculate the score $s(A \to BC, i, k, j)$.

$$\begin{aligned}
s^A_{\mathbf{I}}(a,i,j) &= \sum_{A \to BC \in R_t} \{\sum_{b \in N} \sum_{c \in N}, \int_a \int_b \} W_{A \to w_{i:j}}(a)\ W_{A \to BC}(a,b,c) \\
&\quad \times s^B_{\mathbf{I}}(b,i,k)\ s^C_{\mathbf{I}}(c,k+1,j) \\
s^A_{\mathbf{O}}(a,i,j) &= \sum_{B \to CA \in R_t} \{\sum_{b \in N} \sum_{c \in N}, \int_b \int_c \} W_{A \to w_{i:j}}(a)\ W_{B \to CA}(b,c,a) \\
&\quad \times s^B_{\mathbf{O}}(b,k,j)\ s^C_{\mathbf{I}}(c,k,i-1) \\
&\quad + \sum_{B \to AC \in R_t} \{\sum_{b \in N} \sum_{c \in N}, \int_b \int_c \} W_{A \to w_{i:j}}(a)\ W_{B \to AC}(b,a,c) \\
&\quad \times s^B_{\mathbf{O}}(b,i,k)\ s^C_{\mathbf{I}}(c,j+1,k) \\
s(A \to BC, i, k, j) &= \{\sum_{a \in N} \sum_{b \in N} \sum_{c \in N}, \int_a \int_b \int_c \} W_{A \to BC}(a,b,c) \times s^A_{\mathbf{O}}(a,i,j) \\
&\quad \times s^B_{\mathbf{I}}(b,i,k) \times s^C_{\mathbf{I}}(c,k+1,j)
\end{aligned}$$

## Experiments and Analysis

**Dataset:** Stanford Sentiment TreeBank (SST)
**Code:** https://github.com/Ehaschia/bi-tree-lstm-crf

| Model | 5-class R | 5-class P | 2-class R | 2-class P |
|---|---|---|---|---|
| Pretrain: Glove | | | | |
| 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** |
| Pretrain: ELMo | | | | |
| 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 1. Experimental results. R is sentence accuracy, P is phrase accuracy.

**Analysis:** We analyze the accuracy improvements of phrases of different heights (left) and whether our Latent Vector Sentiment Grammar model can accurately model different emotional subtypes (right) .

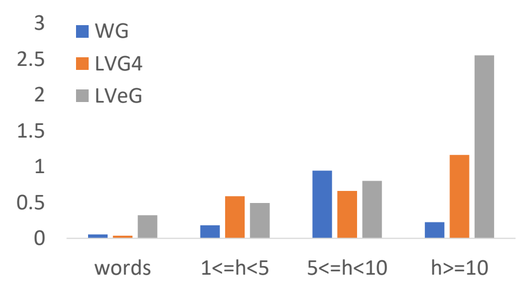



Figure 1. Changes in phrase level 5-class accuracies of our methods over ConTree.


Figure 2. Phrases not longer than 5 in the strong negative sentiment space.