RNN for Sentiment Analysis: Recursive Deep Models for Semantic Compositionality Over a Sentiment Treebank

Borui(Athena) Ye

University of Waterloo

borui.ye@uwaterloo.ca

July 15, 2015
Overview

1 Introduction

2 Related Work

3 Stanford Sentiment Treebank

4 Recursive Neural Models
   RNN: Recursive Neural Network
   MV-RNN: Matrix-Vector RNN
   RNTN: Recursive Neural Tensor Network
   Tensor Backprop through Structure

5 Experiments
Richard Socher, Alex Perelygin, Jean Y.Wu, Jason Chuang, Christopher D. Manning, Andrew Y. Ng and Christopher Potts, 
**Recursive Deep Models for Semantic Compositionality Over a Sentiment Treebank,** 
Introduction

Semantic Compositionality: to calculate in a systematic way the polarity values of larger syntactic constituents as some function of the polarities of their subconstituents[1].

Corpus (Sentiment Treebank)

- 11,855 sentences based on extracted from movie reviews [2]
- 215,154 phrases parsed from sentences using Stanford Parser[3], each annotated by 3 annotators.

Introduction (Cont.)

This film doesn't care about wit or any other kind of intelligent humor.
Introduction (Cont.)

Experiments

1. Fine-grained Sentiment For All Phrases
2. Full Sentence Binary Sentiment
3. Model Analysis: Contrastive Conjunction
4. Model Analysis: High Level Negation
   - Negating Positive Sentences
   - Negating Negative Sentences
5. Model Analysis: Most Positive and Negative Phrases
Related Work

Semantic Vector Spaces. Distributed similarities of single words. But often fail to distinguish antonyms.

- Co-occurrence of a word and its context [4].
- How often a word appears in a certain syntactic context [5].

Compositionality in Vector Spaces. Most of them capture two word compositions.

- Word vector addition, multiplication, etc. [6]
- Represent phrases as matrixes and define composition method as matrix multiplication. [7]


Related Work (Cont.)

Sentiment Analysis.

- Bag-of-words representations [8].
- Extracting features or polarity shifting rules on syntactic structures [9]

**Recursive Neural Models** Will be covered later.


Stanford Sentiment Treebank

Data retrieval and processing:

- Get movie review excerpts from the rotten tomatoes.com, which includes 10,662 sentences, half positive, half negative.
- Parse sentences using the Stanford Parser.
- Using Amazon Mechanical Turk to label the resulting 215,154 phrases.
Statistics

Findings:

1. Most of the short n-grams are neural;
2. Longer n-grams are evenly distributed;
3. Extreme sentiment degrees rarely happen.
Recursive Neural Models

Tri-gram example of bottom up fashion:

Initialization

- Initialize each word vector using uniform distribution: \( U(-r, r) \), where \( r = 0.0001 \).

- Stack word vectors into matrix \( L \in \mathbb{R}^{d \times |V|} \), where \( d \) is vector dimension, \( |V| \) is vocabulary size.
RNN: Recursive Neural Network[10]

\[ p_1 = f \left( W \begin{bmatrix} b \\ c \end{bmatrix} \right), p_2 = f \left( W \begin{bmatrix} a \\ p_1 \end{bmatrix} \right) \]

where \( f = \tanh \), \( W \in \mathbb{R}^{d \times 2d} \)

**MV-RNN: Matrix-Vector RNN[11]**

**Main Idea:** represent every node in the parse tree both as a vector and a matrix.

\[
p_1 = f \left( W \begin{bmatrix} C \ b \\ B \ c \end{bmatrix} \right), \quad P_1 = f \left( W_M \begin{bmatrix} B \\ C \end{bmatrix} \right)
\]

where \( W, W_M \in \mathbb{R}^{d \times 2d} \)

**Problem:** size of parameters becomes very large and depends on the size of the vocabulary.

**Solution:** use a simple powerful composition function with a fixed number of parameters.
RNTN: Recursive Neural Tensor Network

Main Idea: use the same, tensor-based composition function for all nodes.

Definition

- $h \in \mathbb{R}^d$: output of the tensor product
- $V^{[1:d]} \in \mathbb{R}^{2d \times 2d \times d}$: tensor that defines multiple bilinear forms.
- $V^{[i]} \in \mathbb{R}^{2d \times 2d}$: each slice of $V^{[1:d]}$.

\[
\begin{align*}
    p &= f \left( \begin{bmatrix} b \\ c \end{bmatrix}^T V^{[1:2]} \begin{bmatrix} b \\ c \end{bmatrix} + \begin{bmatrix} b \\ c \end{bmatrix} \right) \\
    h &= \begin{bmatrix} b \\ c \end{bmatrix}^T V^{[1:d]} \begin{bmatrix} b \\ c \end{bmatrix};
    h_i &= \begin{bmatrix} b \\ c \end{bmatrix}^T V^{[i]} \begin{bmatrix} b \\ c \end{bmatrix}
\end{align*}
\]
Intuitively, we can interpret each slice of the tensor as capturing a specific type of composition.

\[
p_1 = f \left( \begin{bmatrix} b \\ c \end{bmatrix}^T V^{[1:d]} \begin{bmatrix} b \\ c \end{bmatrix} + W \begin{bmatrix} b \\ c \end{bmatrix} \right)
\]

\[
p_2 = f \left( \begin{bmatrix} a \\ p_1 \end{bmatrix}^T V^{[1:d]} \begin{bmatrix} a \\ p_1 \end{bmatrix} + W \begin{bmatrix} a \\ p_1 \end{bmatrix} \right)
\]
Each node is assigned a label via:

\[ y^a = \text{softmax}(W_s a) \]

where \( W_s \in \mathbb{R}^{5 \times d} \) is the sentiment classification matrix.
Tensor Backprop through Structure (Cont.)

**Goal:** minimize the KL-divergence between the predicted distribution $y^i \in \mathbb{R}^{C \times 1}$ at node $i$ and the target distribution $t^i \in \mathbb{R}^{C \times 1}$. The error function of a sentence is:

$$E(\theta) = \sum_i \sum_j t^i_j \log y^i_j + \lambda \|\theta\|^2$$

where $\theta = (V, W, W_s, L)$. 
Experiments

Two kinds of experiment:

• Large quantitative evaluations on the test set.
• Linguistic phenomena: contrastive conjunction and negation.

Baselines:

• Bag-of-words features + Naive Bayes (NB)
• Bag-of-words features + SVM (SVM)
• Bag-of-bigram features + Naive Bayes (BiNB)
• Averages of neural word vectors (VecAvg)
• RNN
• MV-RNN
Sentiment Classification

1. Exp. 1: Fine-grained Sentiment For All Phrases
2. Exp. 2: Full Sentence Binary Sentiment
Accuracy

- Recursive models work better on shorter grams.
- RNTN upper bounds other models at most $n$-gram lengths.

<table>
<thead>
<tr>
<th>Model</th>
<th>Fine-grained</th>
<th></th>
<th>Positive/Negative</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>All</td>
<td>Root</td>
<td>All</td>
<td>Root</td>
</tr>
<tr>
<td>NB</td>
<td>67.2</td>
<td>41.0</td>
<td>82.6</td>
<td>81.8</td>
</tr>
<tr>
<td>SVM</td>
<td>64.3</td>
<td>40.7</td>
<td>84.6</td>
<td>79.4</td>
</tr>
<tr>
<td>BiNB</td>
<td>71.0</td>
<td>41.9</td>
<td>82.7</td>
<td>83.1</td>
</tr>
<tr>
<td>VecAvg</td>
<td>73.3</td>
<td>32.7</td>
<td>85.1</td>
<td>80.1</td>
</tr>
<tr>
<td>RNN</td>
<td>79.0</td>
<td>43.2</td>
<td>86.1</td>
<td>82.4</td>
</tr>
<tr>
<td>MV-RNN</td>
<td>78.7</td>
<td>44.4</td>
<td>86.8</td>
<td>82.9</td>
</tr>
<tr>
<td>RNTN</td>
<td><strong>80.7</strong></td>
<td><strong>45.7</strong></td>
<td><strong>87.6</strong></td>
<td><strong>85.4</strong></td>
</tr>
</tbody>
</table>
Exp. 3: Model Analysis: Contrastive Conjunction

**X but Y Structure**: two phrases, X and Y, connect by “but”.

**Experiment result**: the test set includes 131 cases (subset of the original test set), RNTN achieve a accuracy of 41%, compared to MV-RNN (37), RNN (36) and biNB (27).
Exp. 4: Model Analysis: High Level Negation

Set 1: Negating Positive Sentences
Exp. 4: Model Analysis: High Level Negation (Cont.)

Set 2: Negating Negative Sentences
Exp. 4: Model Analysis: High Level Negation (Cont.)

<table>
<thead>
<tr>
<th>Model</th>
<th>Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Negated Positive</td>
</tr>
<tr>
<td>biNB</td>
<td>19.0</td>
</tr>
<tr>
<td>RNN</td>
<td>33.3</td>
</tr>
<tr>
<td>MV-RNN</td>
<td>52.4</td>
</tr>
<tr>
<td>RNTN</td>
<td>71.4</td>
</tr>
</tbody>
</table>
Thank you!