# RECURSIVE DEEP MODELS FOR SEMANTIC COMPOSITIONALITY OVER A SENTIMENT TREEBANK

**Richard Socher, Alex Perelygin, Jean Y. Wu, Jason Chuang, Christopher D. Manning, Andrew Y. Ng and Christopher Potts**

Presented By: Dwayne Campbell

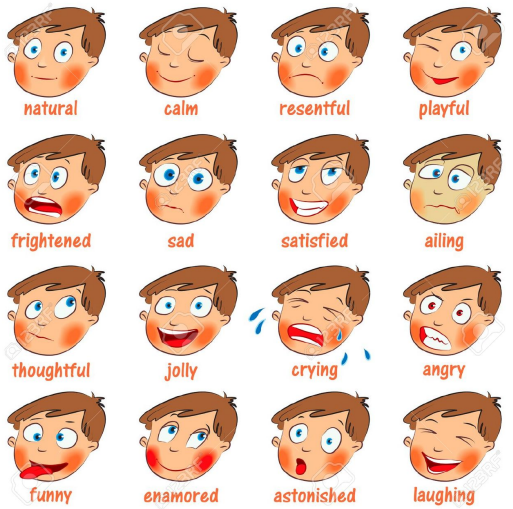
# Overview

- Introduction
- Problem
- Stanford Sentiment Treebank
- Models
- Experiments

# Introduction | Sentiment

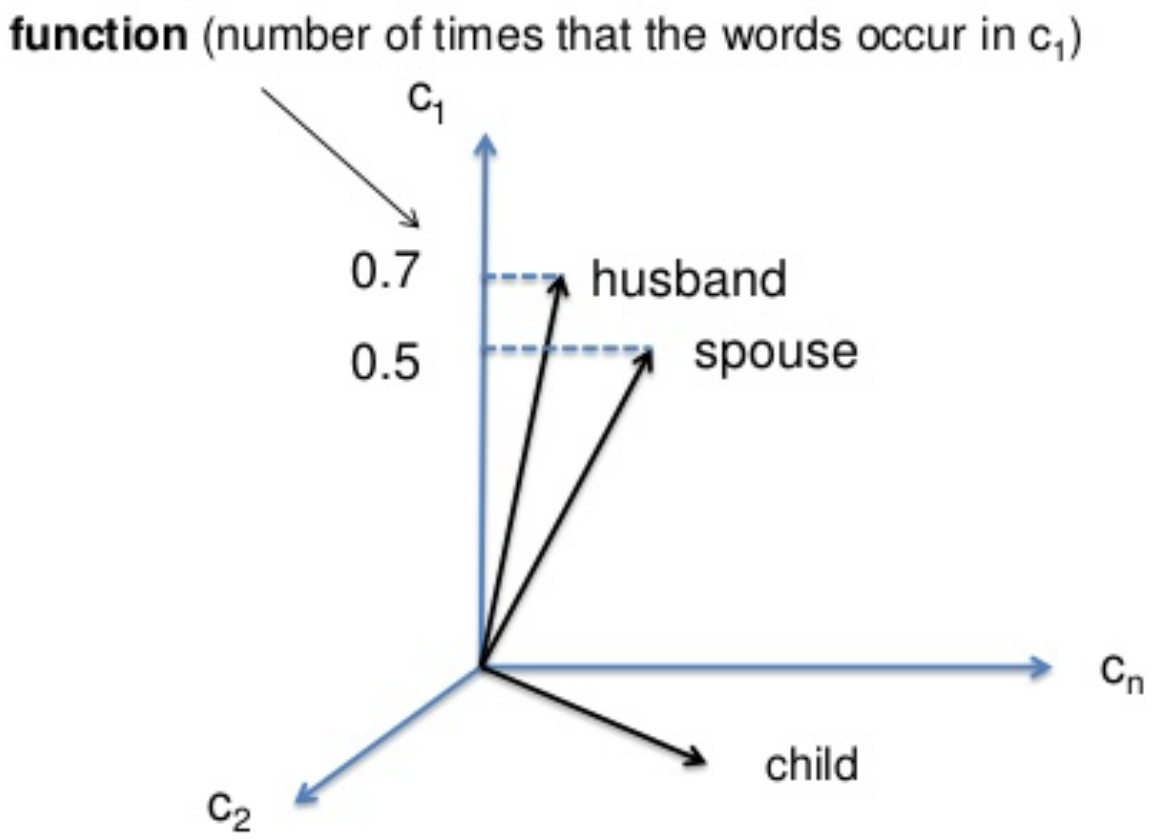
Sentiment ?

- Attitude
- Emotions
- Opinions




Ex. For/against, good/bad, positive/negative

# Introduction | Vector Space Model



# Problem

- Lack of large labeled compositionality corpus and models that can accurately capture the underlying phenomena in such data
- Semantic vector spaces are very useful but cannot express the meaning of longer phrases by themselves

# Stanford Sentiment Treebank

- First corpus with fully labeled parse trees
- 10,662 single sentences extracted from movie reviews
- 215,154 unique phrases generated by the Stanford parser
- Each phrase annotated by 3 human judges

# Stanford Sentiment Treebank

1. 10,662 sentences where obtained and further parsed into 215, 154 phrases using the Stanford Parser
2. Each phrase is annotated by 3 human annotators . Presented with a slider of 25 different values initially set to neutral
3. Phrases were randomly sampled from the set of all phrases


Figure 3: The labeling interface. Random phrases were shown and annotators had a slider for selecting the sentiment and its degree.

- Majority of shorter phrases are neutral. Sentiment often builds up in longer phrases
- Most annotators used 1/5 positions [negative, somewhat negative ,neutral , positive or somewhat positive]
- As a result the main experiment is to recover these five labels

# Stanford Sentiment Treebank

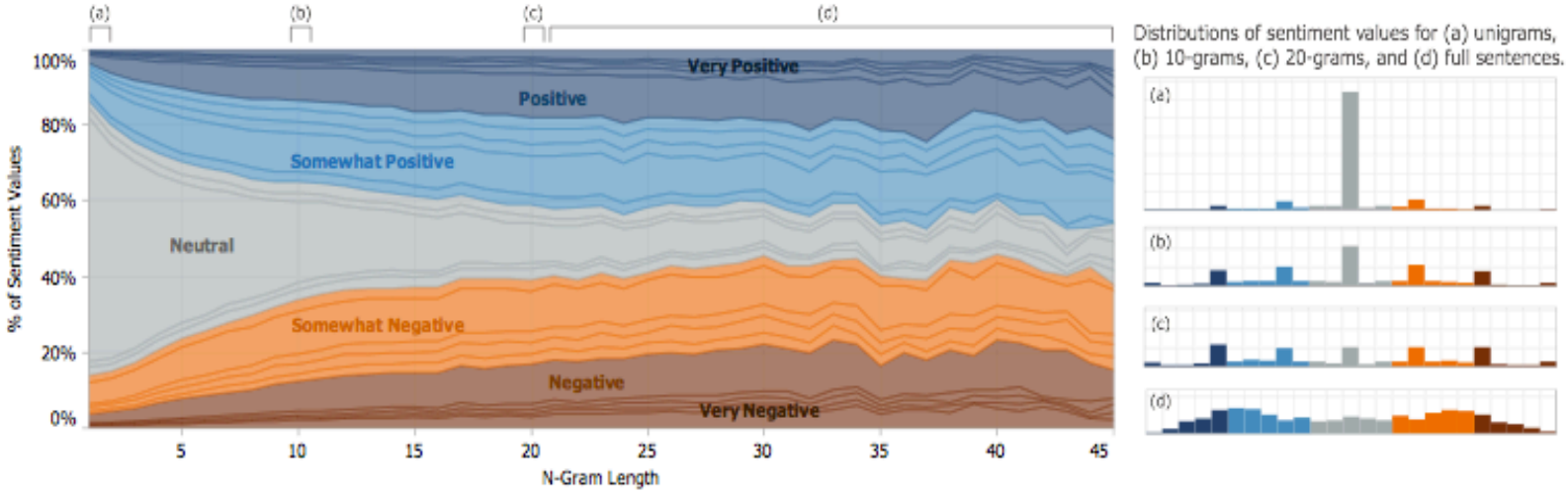



Figure 2: Normalized histogram of sentiment annotations at each $n$-gram length. Many shorter $n$-grams are neutral; longer phrases are well distributed. Few annotators used slider positions between ticks or the extreme values. Hence the two strongest labels and intermediate tick positions are merged into 5 classes.

# Models - General

All models share the following:

- Compute compositional vector representations for phrases of variable length.
- Use the compositional vector representations derived from above as features to classify each phrase.

1. N-grams passed to compositional models, it is then parsed into a binary tree where each leaf node is represented as a vector.
2. Recursive models then compute parent vectors in a bottom up fashion using different type of compositionally functions g(..)

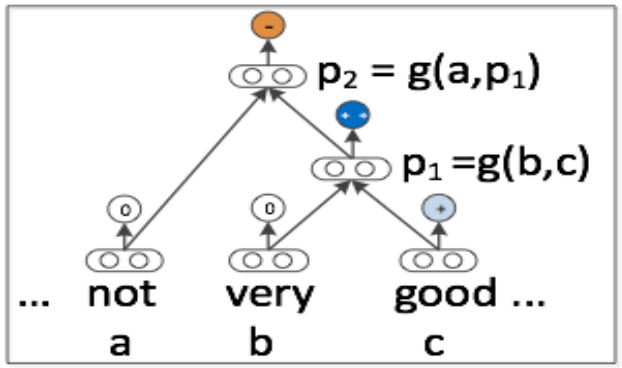



Figure 4: Approach of Recursive Neural Network models for sentiment: Compute parent vectors in a bottom up fashion using a compositionality function $g$ and use node vectors as features for a classifier at that node. This function varies for the different models.

$$y^a = \text{softmax}(W_s a),$$

$$W_s \in \mathbb{R}^{5 \times d}$$

# Model

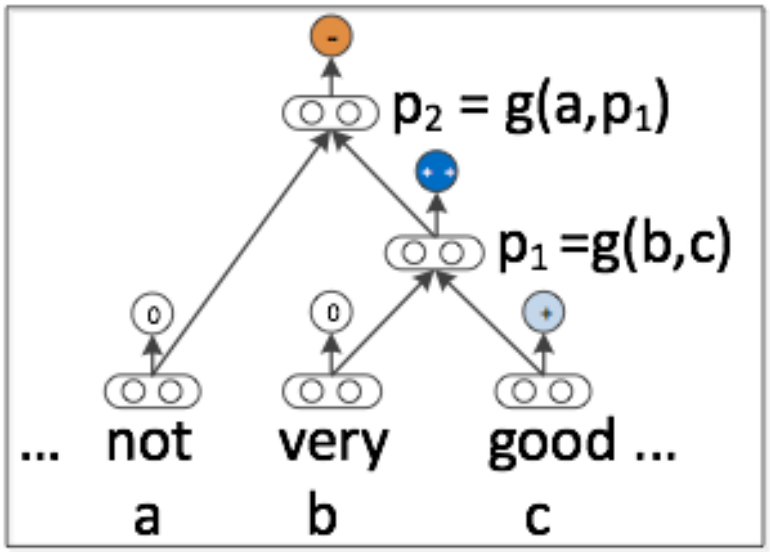



Figure 4: Approach of Recursive Neural Network models for sentiment: Compute parent vectors in a bottom up fashion using a compositionality function $g$ and use node vectors as features for a classifier at that node. This function varies for the different models.

# Model – Recursive Neural Network

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

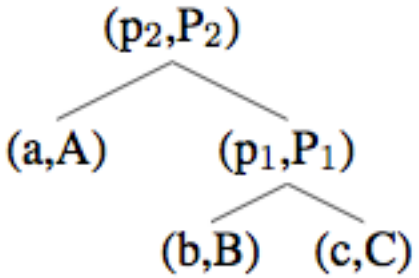
Where f is the tanh

1. It is first determined which parent already has all its children vectors computed.
2. Parent vectors are then computed in a bottom up fashion.
3. Once the parent vectors have been computed they are given to the same softmax classifier to compute its label probability.

*Disadvantage:*

Not enough interaction since the input vectors only implicitly interact through the nonlinearity (squashing) function

# Models – MV-RNN




- The main idea of this model is to represent each word as both a vector and a matrix

$$p_1 = f\left(W\begin{bmatrix} Cb \\ Bc \end{bmatrix}\right), P_1 = f\left(W_M\begin{bmatrix} B \\ C \end{bmatrix}\right),$$

*Disadvantage:*

The number of parameters become very large and is dependent on the vocabulary

# Models – RNTN

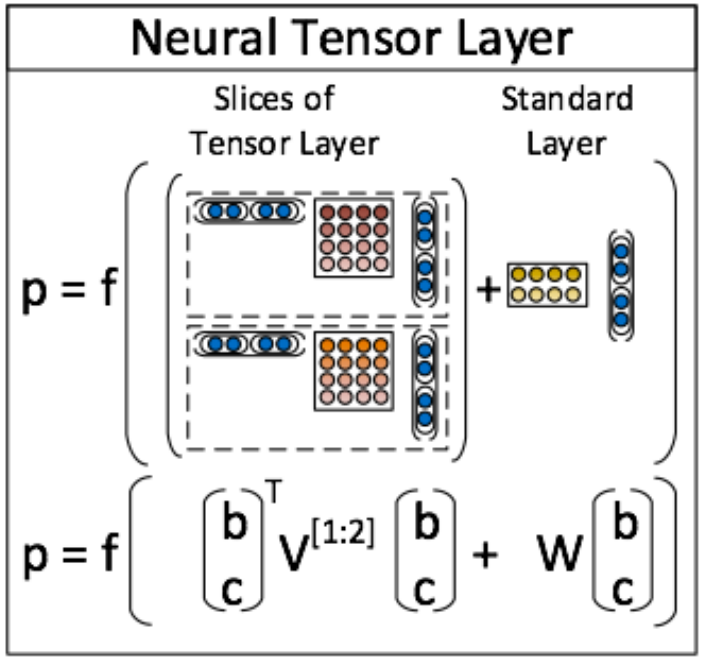



Figure 5: A single layer of the Recursive Neural Tensor Network. Each dashed box represents one of $d$-many slices and can capture a type of influence a child can have on its parent.

$$h = \begin{bmatrix} b \\ c \end{bmatrix}^T V^{[1:d]} \begin{bmatrix} b \\ c \end{bmatrix}; h_i = \begin{bmatrix} b \\ c \end{bmatrix}^T V^{[i]} \begin{bmatrix} b \\ c \end{bmatrix}.$$

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

$$p_2 = f\left(\begin{bmatrix} a \\ p_1 \end{bmatrix}^T V^{[1:d]} \begin{bmatrix} a \\ p_1 \end{bmatrix} + W \begin{bmatrix} a \\ p_1 \end{bmatrix}\right).$$

# Experiments

| Model | Fine-grained | | Positive/Negative | |
|---|---|---|---|---|
| | All | Root | All | Root |
| NB | 67.2 | 41.0 | 82.6 | 81.8 |
| SVM | 64.3 | 40.7 | 84.6 | 79.4 |
| BiNB | 71.0 | 41.9 | 82.7 | 83.1 |
| VecAvg | 73.3 | 32.7 | 85.1 | 80.1 |
| RNN | 79.0 | 43.2 | 86.1 | 82.4 |
| MV-RNN | 78.7 | 44.4 | 86.8 | 82.9 |
| RNTN | **80.7** | **45.7** | **87.6** | **85.4** |

Table 1: Accuracy for fine grained (5-class) and binary predictions at the sentence level (root) and for all nodes.

- sentence treebank were split into (8544), dev(1101) and test splits(2210).
- dev set was used to cross-validate over regularization of weights,word vector sizes, learning rate and mini batch for AdaGrad.

Optimal performance when:

- word vector sizes between 25-30.
- batch sizes between 20 & 30.

# Model Analysis – Contrastive Conjunction

RNTN(41%) , MV-RNN(37%), RNN(36%) & biNB(27%)

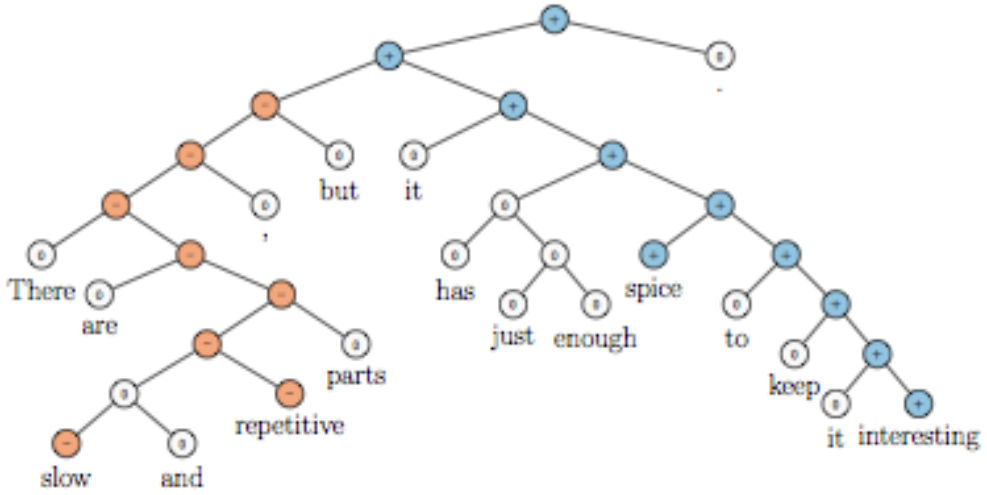



Figure 7: Example of correct prediction for contrastive conjunction *X but Y*.

# Model Analysis – High Level Negation

| Model | Accuracy | |
|---|---|---|
| | Negated Positive | Negated Negative |
| biNB | 19.0 | 27.3 |
| RNN | 33.3 | 45.5 |
| MV-RNN | 52.4 | 54.6 |
| RNTN | **71.4** | **81.8** |

Table 2: Accuracy of negation detection. Negated positive is measured as correct sentiment inversions. Negated negative is measured as increases in positive activations.

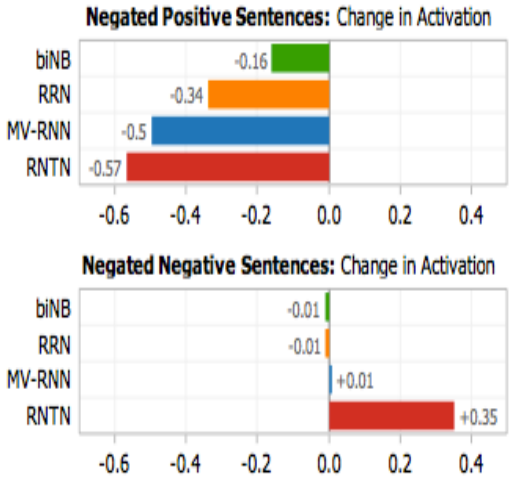



Figure 8: Change in activations for negations. Only the RNTN correctly captures both types. It decreases positive sentiment more when it is negated and learns that negating negative phrases (such as *not terrible*) should increase neutral and positive activations.

# Model Analysis – High Level Negation

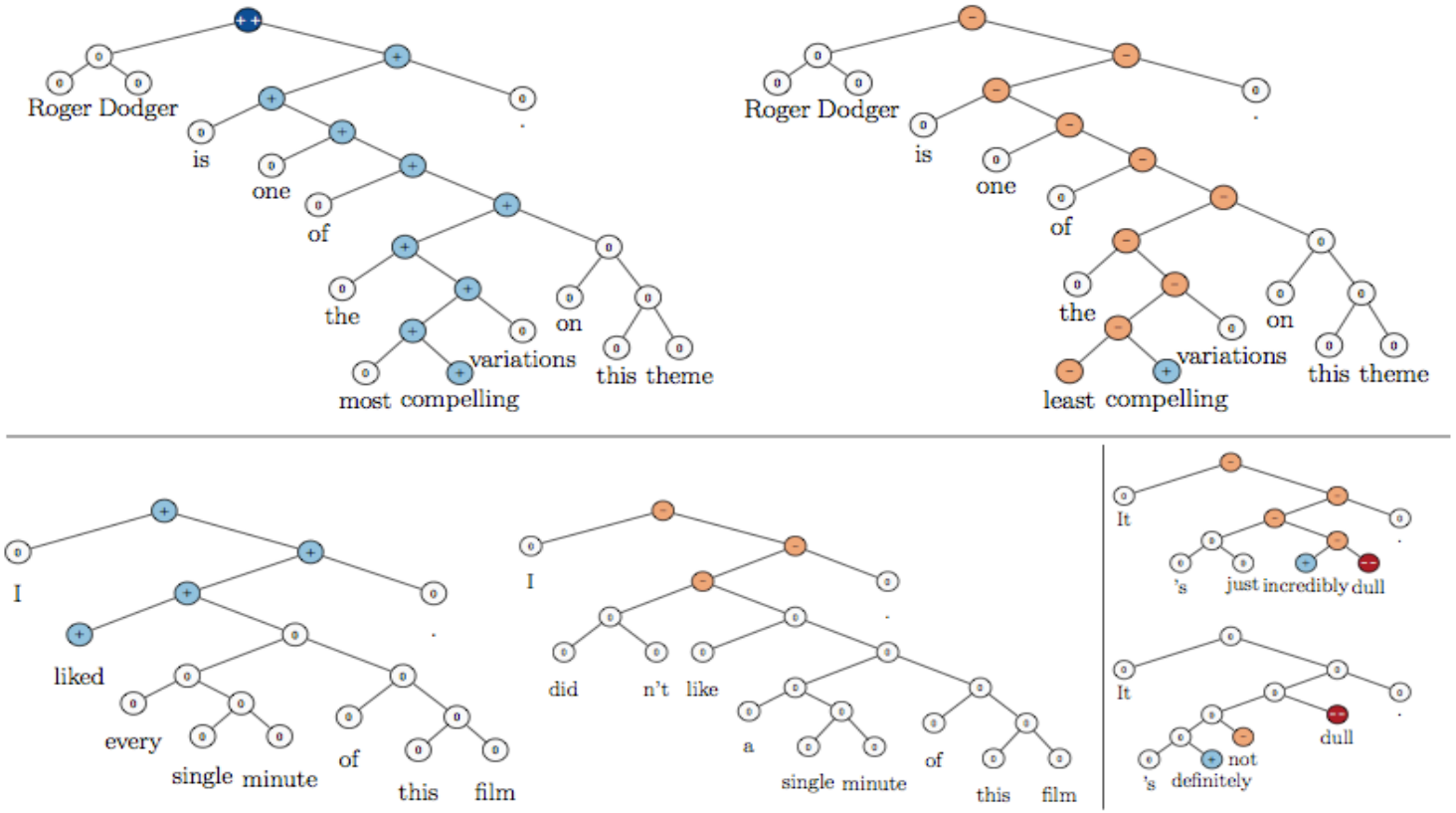



Figure 9: RNTN prediction of positive and negative (bottom right) sentences and their negation.

# End

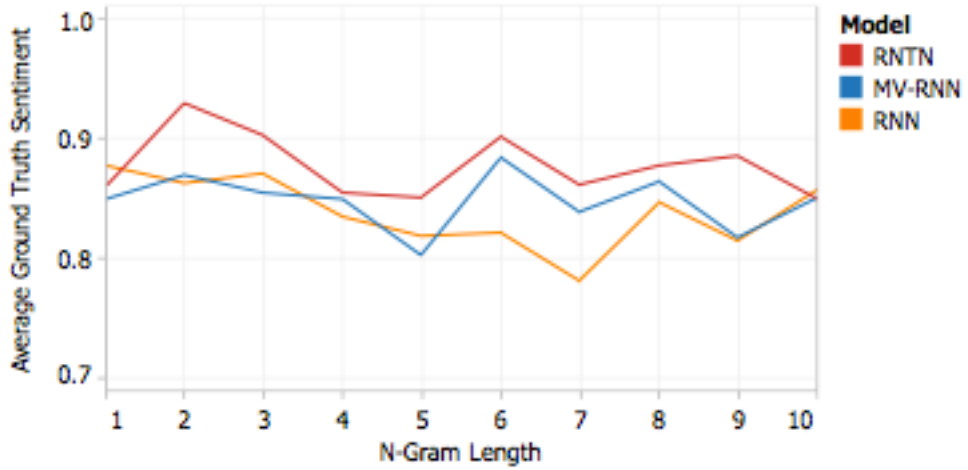



Figure 10: Average ground truth sentiment of top 10 most positive $n$-grams at various $n$. The RNTN correctly picks the more negative and positive examples.

# End

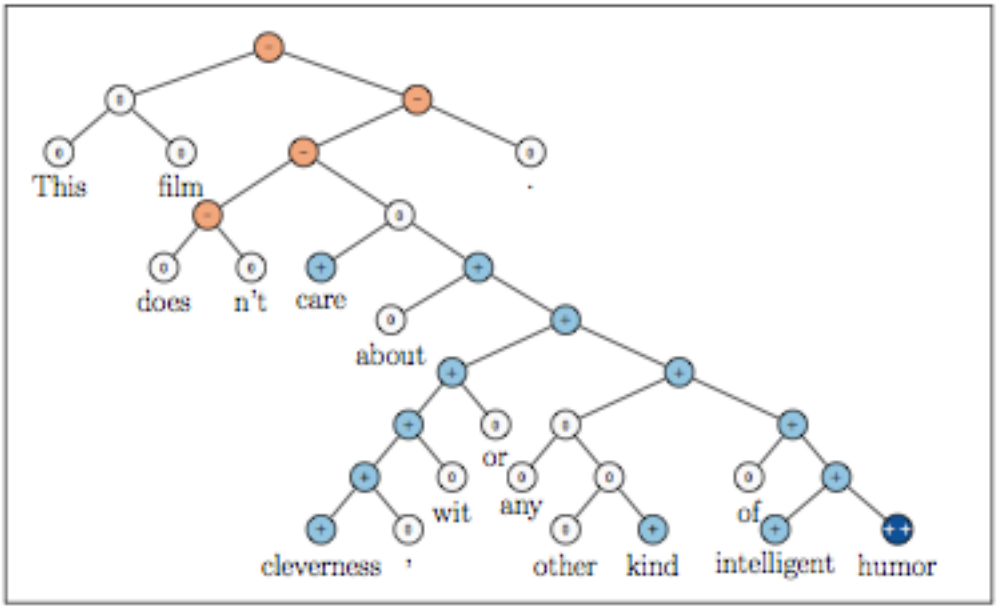



Figure 1: Example of the Recursive Neural Tensor Network accurately predicting 5 sentiment classes, very negative to very positive (– –, –, 0, +, + +), at every node of a parse tree and capturing the negation and its scope in this sentence.