Constituency Parsing

Natural Language Processing

(based on revision of Chris Manning Lectures)

Announcement

- TA announcements (if any)...

Suggested Readings

1. https://web.stanford.edu/~jurafsky/slp3/12.pdf (context-free grammar)
2. https://web.stanford.edu/~jurafsky/slp3/13.pdf (constituency grammar)
3. Parsing with Compositional Vector Grammars
4. Constituency Parsing with a Self-Attentive Encoder

Recap: Context-free grammar / Constituency grammar

NP -> Det N
e.g., the cat

NP -> Det (Adj) N
e.g., the large cat

PP -> P NP
e.g., by the door

NP -> Det (Adj) N (PP)
e.g., the large cat by the door

NP -> Det (Adj)* N (PP)
e.g., the large cute furry cat by the door

VP -> V PP
e.g., talk to the cat

S -> NP VP
e.g., the cat walked behind the dog

More grammar rules! As much as we want....

Composition of meanings

How can we work out the meaning of larger phrases?

*The **snowboarder** is leaping over a mogul*

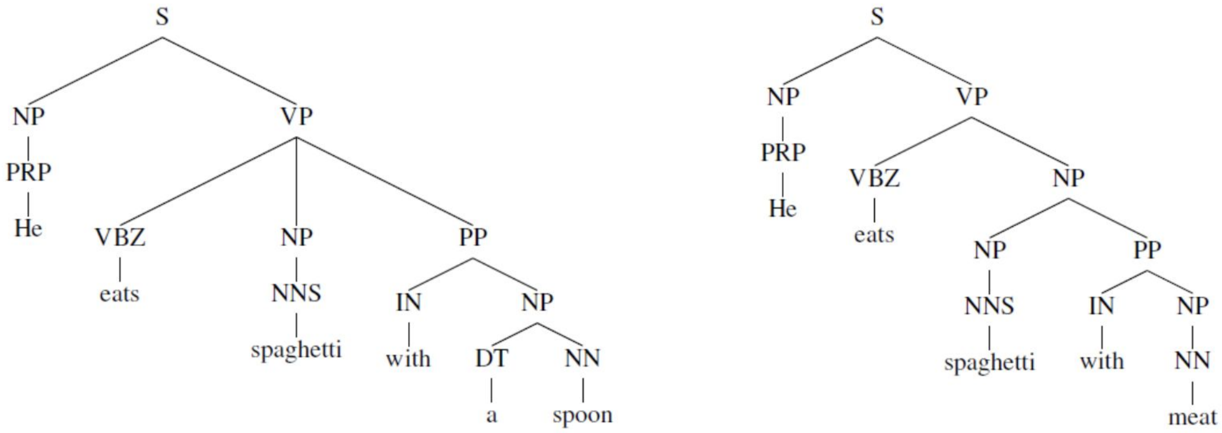
*A **person on a snowboard** jumps into the air*

People interpret the meaning of larger text units – entities, descriptive terms, facts, arguments, stories – by **semantic composition of smaller elements**

Language in recursive structure

Language can be expressed in **recursive structure** (i.e., context-free grammar or constituency grammar)

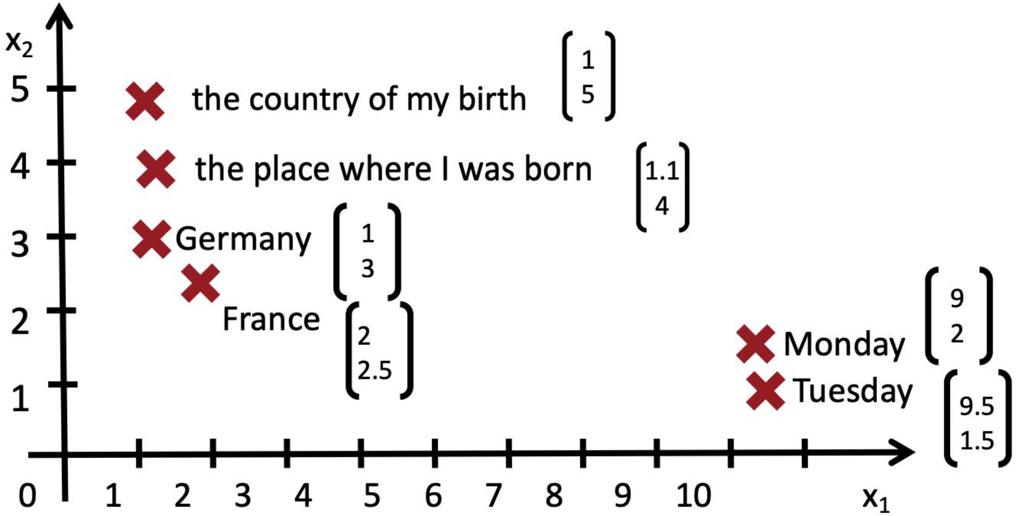
[The person standing next to [the man from [the company that purchased [the firm that you used to work at]]]]



Version 1: Tree + RNN

How to start now? [Socher et al., ICML 2011]

Can we get some phrase embeddings?

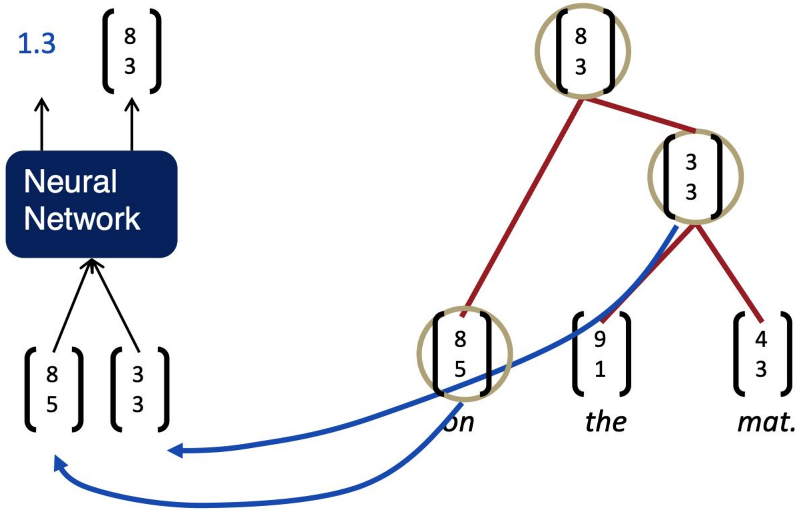


Parsing Natural Scenes and Natural Language with Recursive Neural Networks, Socher et al. 2011, https://www-nlp.stanford.edu/pubs/SocherLinNgManning_ICML2011.pdf

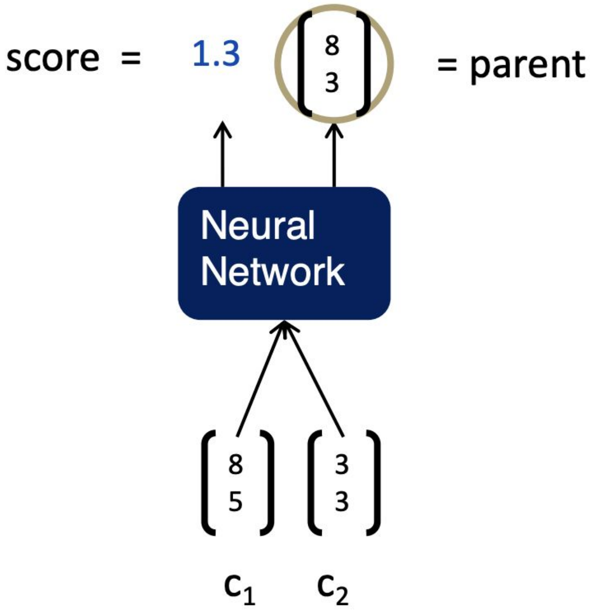
Input and output

Input: two children’s embeddings

Output: (1) **composition goodness score**, (2) **combined embeddings**



Input and output



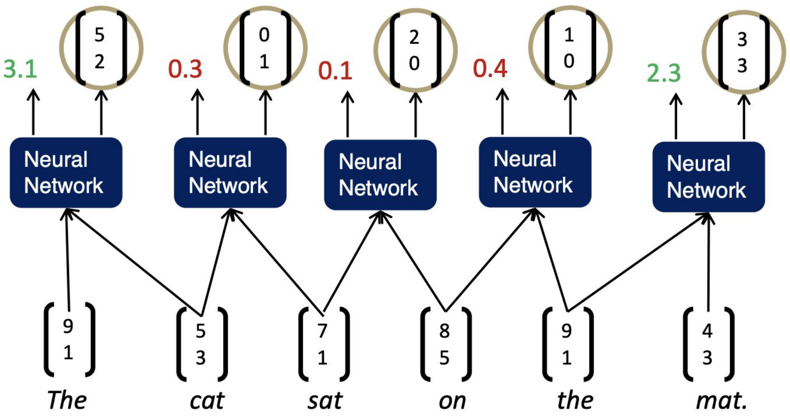
Note: Same W is used at all nodes. [;] is concatenation

$$\text{parent} = \tanh(\mathbf{W}[\mathbf{c}_1; \mathbf{c}_2] + \mathbf{b})$$

$$\text{score} = \mathbf{U}^\top \text{parent}$$

Greedy Parsing

Algorithm:
Compute all contiguous pair of inputs. Choose the best score and merge. And repeat until no more nodes can be merged.



The **score** of a tree is computed by the sum of the parsing decision scores at each node where x is the sentence (the cat sat on the mat) and y is the result tree:

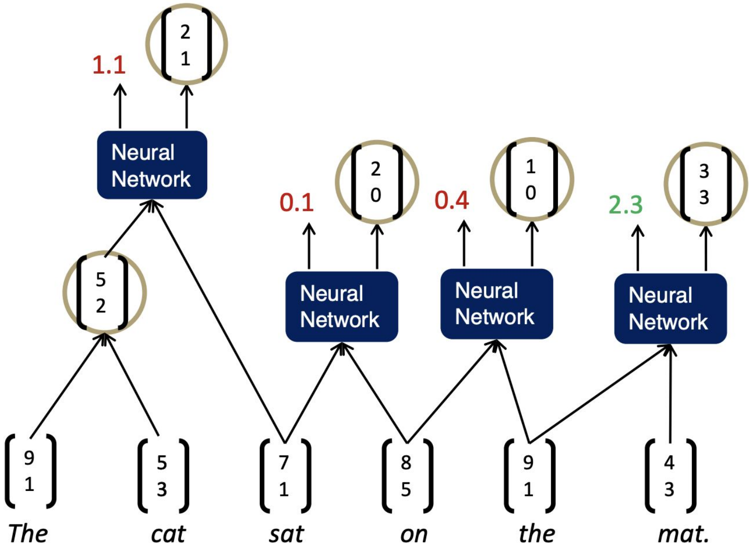
$$s(x, y) = \sum_{n \in \text{nodes}(y)} s_n$$

The loss is based on **max-margin parsing** (Taskar et al., 2004), where the loss is defined as (**x** is the sentence, **y** is the true parse tree, **yhat** is the best parse tree, **A** is the function that yields the tree, **Δ** defines the margin we want to penalize for each wrong merge)

$$J = \sum_i s(x, y) - \max_{\hat{y} \in A(x)} \left(s(x, \hat{y}) + \Delta(\hat{y}, y) \right)$$

Greedy Parsing

Algorithm:
Compute all contiguous pair of inputs. Choose the best score and merge. And repeat until no more nodes can be merged.



The **score** of a tree is computed by the sum of the parsing decision scores at each node where x is the sentence (the cat sat on the mat) and y is the result tree:

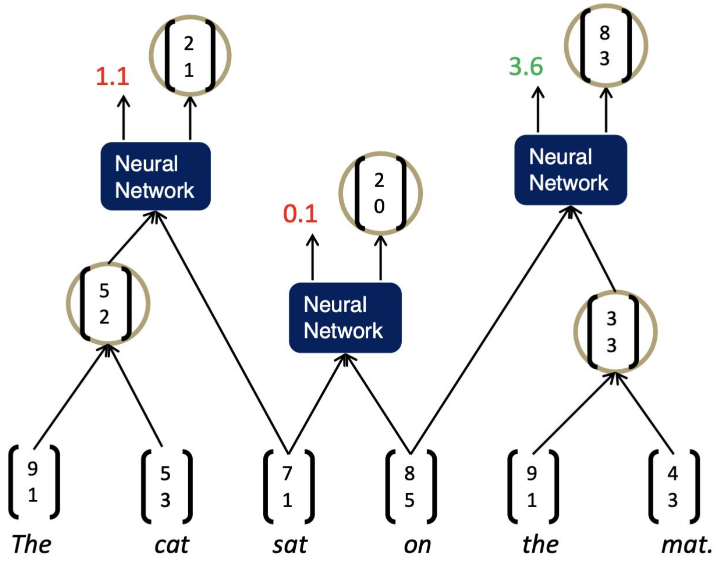
$$s(x, y) = \sum_{n \in \text{nodes}(y)} s_n$$

The loss is based on **max-margin parsing** (Taskar et al., 2004), where the loss is defined as (**x** is the sentence, **y** is the true parse tree, **yhat** is the best parse tree, **A** is the function that yields the tree, **Δ** defines the margin we want to penalize for each wrong merge)

$$J = \sum_i s(x, y) - \max_{\hat{y} \in A(x)} \left(s(x, \hat{y}) + \Delta(\hat{y}, y) \right)$$

Greedy Parsing

Algorithm:
Compute all contiguous pair of inputs. Choose the best score and merge. And repeat until no more nodes can be merged.



The **score** of a tree is computed by the sum of the parsing decision scores at each node where x is the sentence (the cat sat on the mat) and y is the result tree:

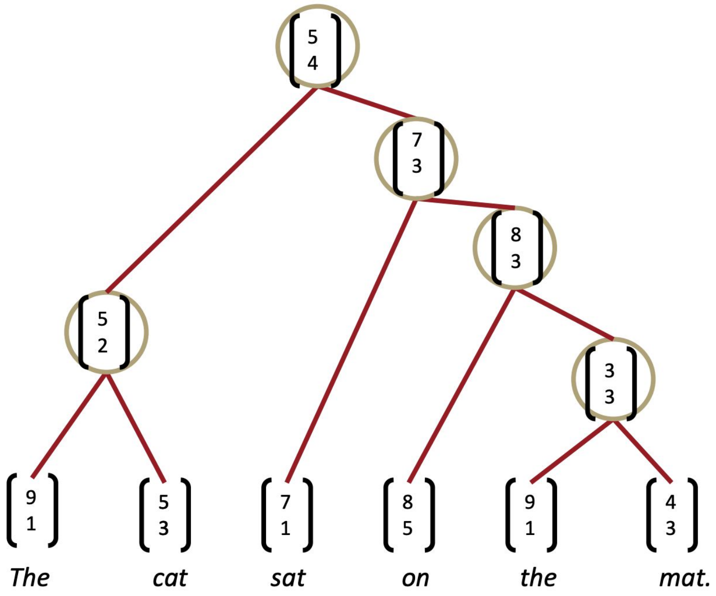
$$s(x, y) = \sum_{n \in \text{nodes}(y)} s_n$$

The loss is based on **max-margin parsing** (Taskar et al., 2004), where the loss is defined as (**x** is the sentence, **y** is the true parse tree, **yhat** is the best parse tree, **A** is the function that yields the tree, **Δ** defines the margin we want to penalize for each wrong merge)

$$J = \sum_i s(x, y) - \max_{\hat{y} \in A(x)} \left(s(x, \hat{y}) + \Delta(\hat{y}, y) \right)$$

Greedy Parsing

Algorithm:
Compute all contiguous pair of inputs. Choose the best score and merge. And repeat until no more nodes can be merged.



The **score** of a tree is computed by the sum of the parsing decision scores at each node where x is the sentence (the cat sat on the mat) and y is the result tree:

$$s(x,y) = \sum_{n \in \text{nodes}(y)} s_n$$

The loss is based on **max-margin parsing** (Taskar et al., 2004), where the loss is defined as (**x** is the sentence, **y** is the true parse tree, **yhat** is the best parse tree, **A** is the function that yields the tree, **Δ** defines the margin we want to penalize for each wrong merge)

$$J = \sum_i s(x,y) - \max_{\hat{y} \in A(x)} \left(s(x,\hat{y}) + \Delta(\hat{y},y) \right)$$

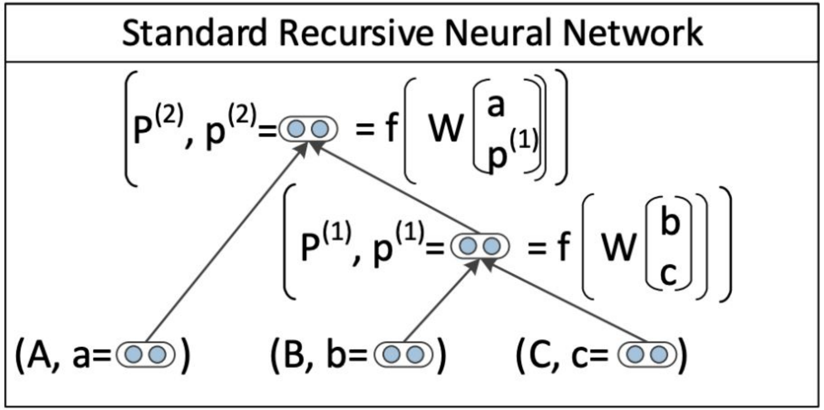
Discussion

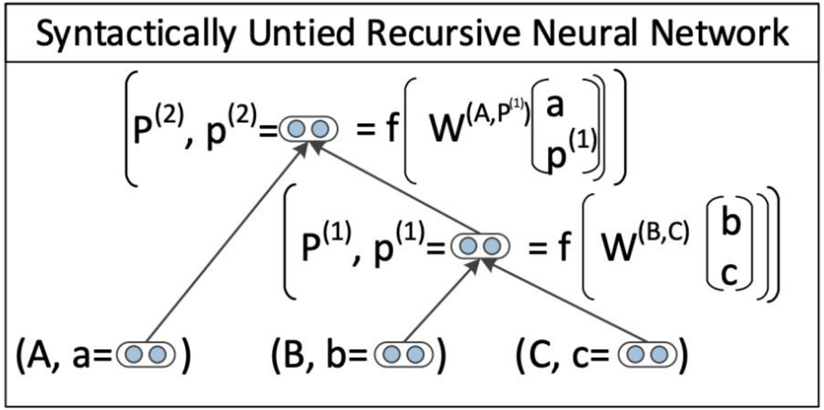
- Potential limitations
    - **Single weight matrix** could NOT capture more complex, higher order composition and parsing long sentences
        - E.g., different semantics between
            - det + noun (e.g., the cat). Here the activation should be higher for “cat”, i.e., the second word
            - np + cc (e.g., cat and). Here the activation should be higher for “cat”, i.e., the first word
        - Addressed in version 2
    - **No interactions between input words**
        - Simple concatenation assumes no interaction. Does not work for some cases, e.g., “very good”; here “very” amplifies “good”, or “should have been good”; here “should have been” negates “good”
        - Addressed in version 3

Version 2: Multiple W + Tree + RNN

Multiple W + Tree + RNN [Socher et al., ACL 2013]

- **Idea**: simply assign a different matrix for every combinations!
- **Problem**: speed
- **Solution**: calculate score only for some very likely combination.....also use some shared W for very similar compositions





Parsing with compositional vector grammars, Socher et al. 2013, https://aclanthology.org/P13-1045.pdf

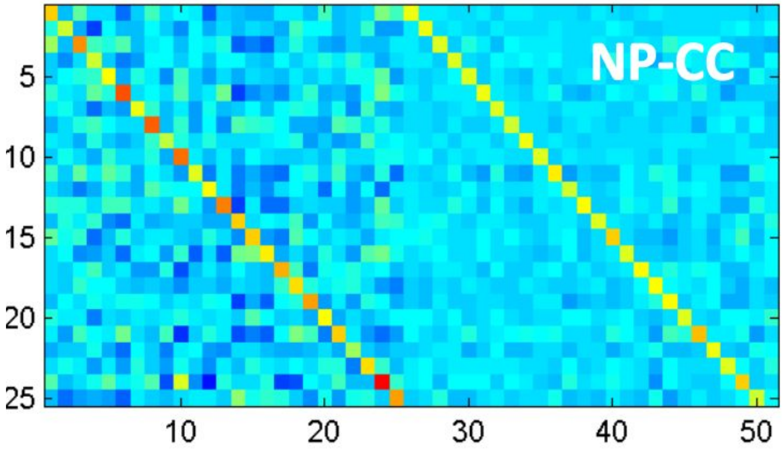
Multiple W + Tree + RNN

SU-RNN (multiple W matrix) seems to outperform other manually featured parser at that time.

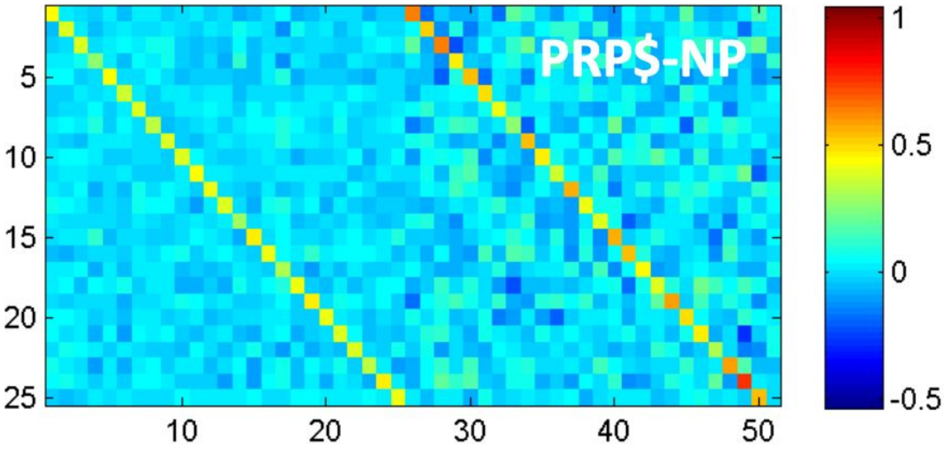
Parser	dev (all)	test$\leq$ 40	test (all)
Stanford PCFG	85.8	86.2	85.5
Stanford Factored	87.4	87.2	86.6
Factored PCFGs	89.7	90.1	89.4
Collins			87.7
SSN (Henderson)			89.4
Berkeley Parser			90.1
CVG (RNN)	85.7	85.1	85.0
CVG (SU-RNN)	91.2	91.1	90.4

Multiple W + Tree + RNN

Different W seems to be able to activate differently based on the composition



e.g., **the cat** and



e.g., at the **door**

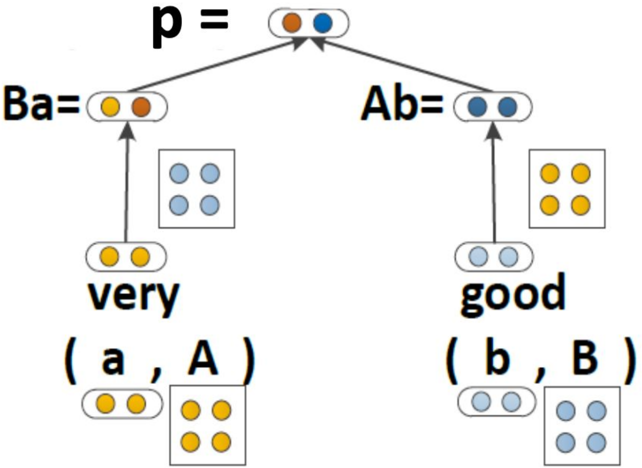
Discussion

- Potential limitations
  - **Speed** remains a bit problem of this approach
  - Again, **no interactions** between input words

Version 3: Matrix-Vector + Tree + RNN

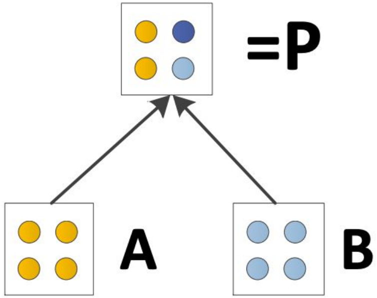
Matrix-Vector + Tree + RNN [Socher et al., EMNLP 2012]

$$p = f\left(W\begin{bmatrix} Ba \\ Ab \end{bmatrix}\right)$$



$$\boxed{P = g(A, B) = W_M\begin{bmatrix} A \\ B \end{bmatrix}}$$

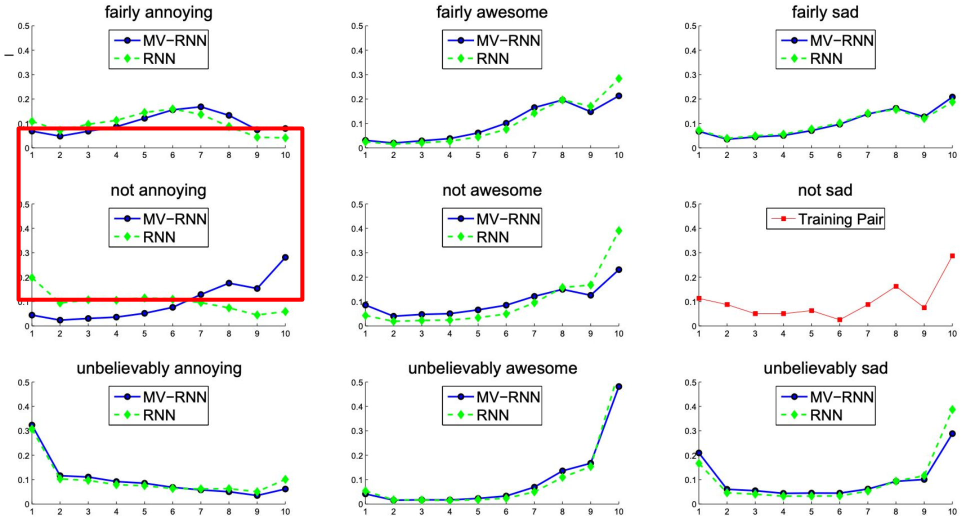
$$W_M \in \mathbb{R}^{n \times 2n}$$



Semantic Compositionality through Recursive Matrix-Vector Spaces, Socher et al. 2012, https://dl.acm.org/doi/pdf/10.5555/2390948.2391084

Matrix-Vector + Tree + RNN

x-axis is the prediction (10 is very positive); y-axis is the probability distribution. For “not annoying”, notice MV-RNN was able to predict 10 with relatively higher probability, while RNN predict a much lower probability for 10.



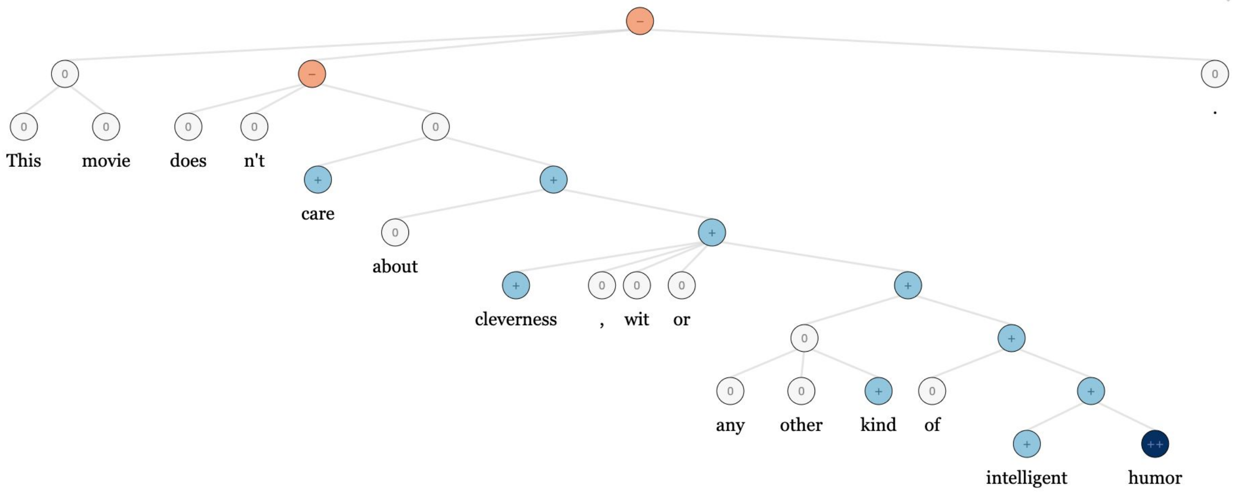
A little bit of sentiment analysis

Is the piece of text **positive**, **negative**, or **neutral**?

- Sentiment is “easy”, especially for very long documents ~90%
    - ….***loved*****…..*****great*****……*****impressed*****…..*****marvelous***
- **However**, if the model “memorizes”, it will not be able to predict this kind of sentence correctly:
    - *The movie should have been* ***better*** *and more* ***entertaining****.*

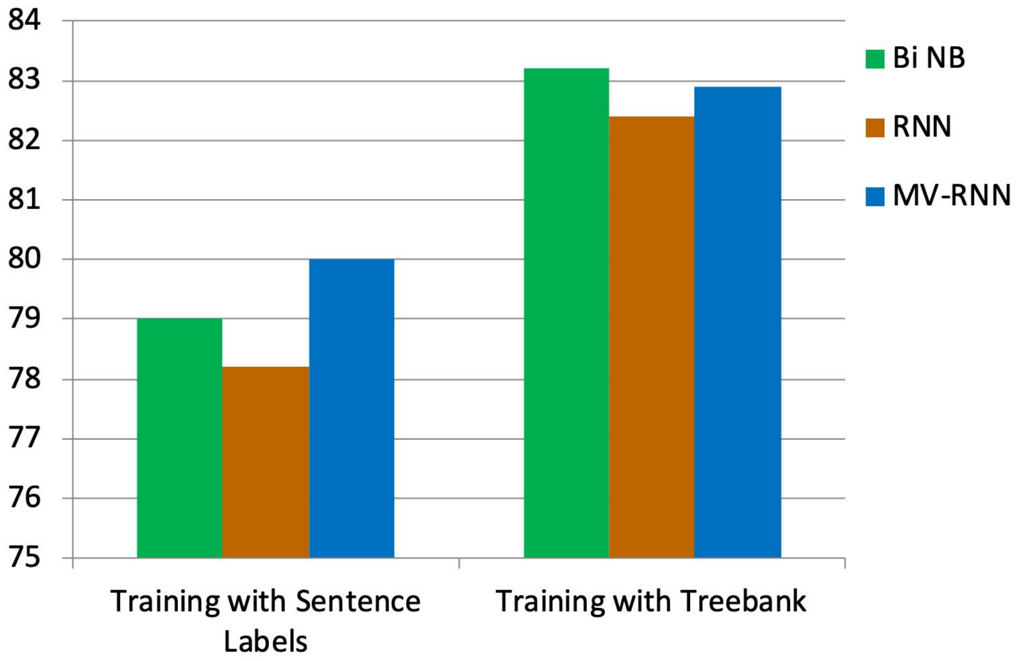
Stanford Sentiment Treebank

- 215,154 **<u>phrases (not sentences!)</u>** labeled in 11,855 sentences
- Can actually help train and test compositions



http://nlp.stanford.edu:8080/sentiment

Better dataset helped all models



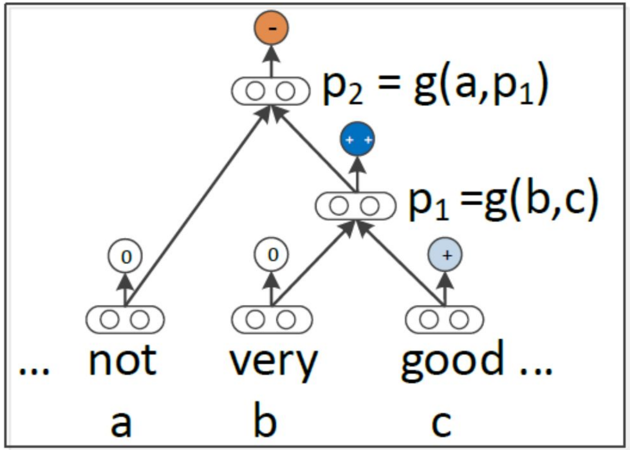
Discussion

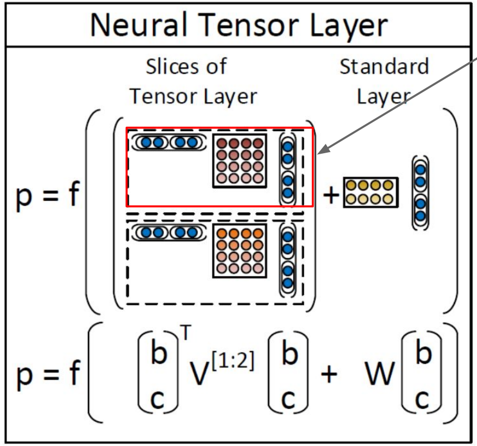
- Potential limitations
    - **Every word comes with an extra matrix....**

Version 4: Tensor + Tree + RNN

Matrix-Vector + Tree + RNN [Socher et al., EMNLP 2013]

- Less parameters than MV-RNN
- Allows the two word or phrase vectors to interact via multiplication through a middle tensor (3-dimensional) matrix



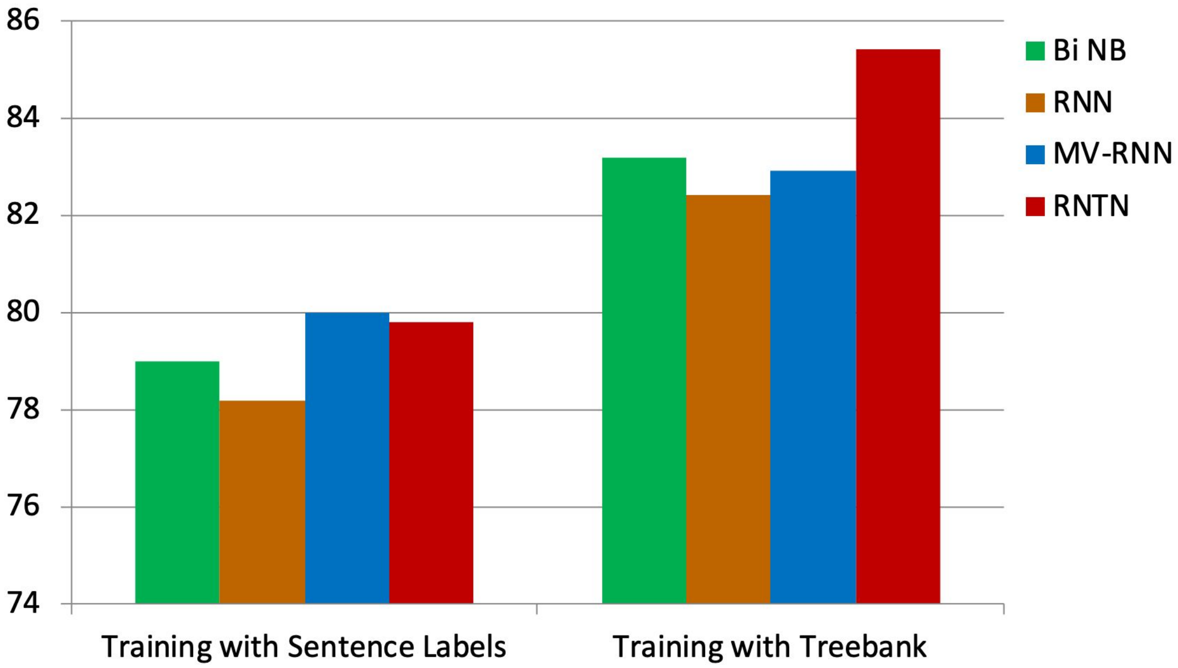


-For each slice
(1 x 4) @ (4 x 4) @ (4 x 1) = (1, 1

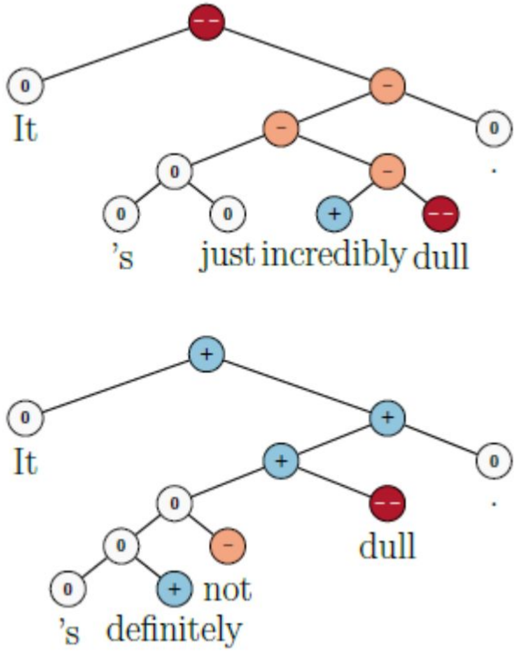
- By having two slices of V, where this number "two" is simply the dimension of the word embedding, we will get a two-dimensional vector. Then we just add with another transformed concatenated vectors

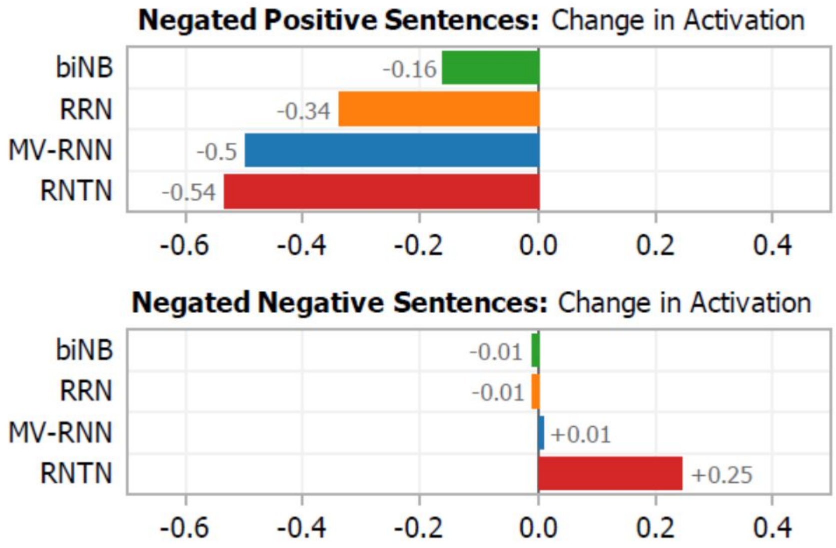
Recursive Deep Models for Semantic Compositionality Over a Sentiment Treebank, Socher et al. 2013, https://aclanthology.org/D13-1170.pdf

Accuracy improves to 85.4



Negating negatives should increase positive activation





Takeaways

- Potential limitations
  - **Cannot think of one at that time…..(maybe plug more powerful model than RNN?)**
  - Many work follows, e.g., using LSTM instead (Tai et al., ACL 2015)
- **Sentiment analysis** or **sentence classification** once you make it a very short sentence!
- Nowadays, **not many works utilized such tree-based approach**. Many possible reasons:
  - Due to the tree structure, does not allow GPU to work efficiently, because the operations are not uniform
  - Many other models can well capture compositionality, especially **CNN** (bigrams, trigrams, etc.) or even **attention** (all possible grams!)
- **No one uses anymore does not mean we should not study. At least, we learn the "underlying" philosophy how researchers think which is even more important!**
- Nevertheless, such tree-based method can be very useful for something like program translation (Chen et al., NeurolPS 2018) when the language is very structured, unlike natural language.

Improved Semantic Representations From Tree-Structured Long Short-Term Memory Networks, Tai et al., 2015, https://dl.acm.org/doi/pdf/10.5555/2390948.2391084
Tree-to-tree Neural Networks for Program Translation, Chen et al. 2018, https://papers.nips.cc/paper/2018/file/d759175de8ea5b1d9a2660e45554894f-Paper.pdf