Convolutional neural network (CNN)

① why it appeared?

From RNNs to Convolutional Neural Nets

- Main Convolutional Neural Net (CNN/ConvNet) idea:
  - What if we compute vectors for every possible word subsequence of a certain length?
- Example: "tentative deal reached to keep government open" computes vectors for:
  - tentative deal reached, deal reached to, reached to keep, to keep government, keep government open
- Regardless of whether subsequence is grammatical or a natural linguistic constituent
  - Not very linguistically or cognitively plausible
- Then group them afterwards (more soon)

There's no transformer at that time

We just going to form representations of multi-word units

② Standard way to achieve ↑

What is a convolution anyway?

- 1d discrete convolution generally: $(f * g)[n] = \sum_{m=-M}^{M} f[n-m]g[m].$
- Convolution is classically used to extract features from images
  - Models position-invariant identification
  - Longer version in cs231n!
- 2d example →
- Yellow color and red numbers show filter (=kernel) weights
- Green shows input
- Pink shows output

Image

Convolved Feature

From Stanford UFLDL wiki

classic case of convolutional neural networks is in Vision

↓

for language we have no 2D picture, only 1D

↓

A 1D convolution for text

tentative	0.2	0.1	−0.3	0.4
deal	0.5	0.2	−0.3	−0.1
reached	−0.1	−0.3	−0.2	0.4
to	0.3	−0.3	0.1	0.1
keep	0.2	−0.3	0.4	0.2
government	0.1	0.2	−0.1	−0.1
open	−0.4	−0.4	0.2	0.3

Apply a **filter** (or **kernel**) of size 3

3	1	2	−3
−1	2	1	−3
1	1	−1	1

+ 1.0 sigmoid(·) . . .

t,d,r	−1.0	0.0	0.50
d,r,t	−0.5	0.5	0.38
r,t,k	−3.6	-2.6	0.93
t,k,g	−0.2	0.8	0.31
k,g,o	0.3	1.3	0.21

+ bias

→ non-linearity

what if we want to keep the size unchanged?

↓

1D convolution for text with padding

Ø	0.0	0.0	0.0	0.0
tentative	0.2	0.1	−0.3	0.4
deal	0.5	0.2	−0.3	−0.1
reached	−0.1	−0.3	−0.2	0.4
to	0.3	−0.3	0.1	0.1
keep	0.2	−0.3	0.4	0.2
government	0.1	0.2	−0.1	−0.1
open	−0.4	−0.4	0.2	0.3
Ø	0.0	0.0	0.0	0.0

Ø,t,d	−0.6
t,d,r	−1.0
d,r,t	−0.5
r,t,k	−3.6
t,k,g	−0.2
k,g,o	0.3
g,o,Ø	−0.5

Apply a **filter** (or **kernel**) of size 3

3	1	2	−3
−1	2	1	−3
1	1	−1	1

Could also use (zero) padding = 2
Also called "wide convolution"

if we only had one filter, thing are pretty limiting

↓

3 channel 1D convolution with padding = 1 and 3 filters

Ø	0.0	0.0	0.0	0.0
tentative	0.2	0.1	−0.3	0.4
deal	0.5	0.2	−0.3	−0.1
reached	−0.1	−0.3	−0.2	0.4
to	0.3	−0.3	0.1	0.1
keep	0.2	−0.3	0.4	0.2
government	0.1	0.2	−0.1	−0.1
open	−0.4	−0.4	0.2	0.3
Ø	0.0	0.0	0.0	0.0

Ø,t,d	−0.6	0.2	1.4
t,d,r	−1.0	1.6	−1.0
d,r,t	−0.5	−0.1	0.8
r,t,k	−3.6	0.3	0.3
t,k,g	−0.2	0.1	1.2
k,g,o	0.3	0.6	0.9
g,o,Ø	−0.5	−0.9	0.1

Apply 3 **filters** of size 3

3	1	2	−3
−1	2	1	−3
1	1	−1	1

1	0	0	1
1	0	−1	−1
0	1	0	1

1	−1	2	−1
1	0	−1	3
0	2	2	1

channel

output channel = the number of kernal
input channel is fixed

eg. RGB (3) —conv→ 16 channel
the channel here is the different features learned

And we want to Summerize all of these filters

↓

Ø,t,d	−0.6	0.2	1.4
t,d,r	−1.0	1.6	−1.0
d,r,t	−0.5	−0.1	0.8
r,t,k	−3.6	0.3	0.3
t,k,g	−0.2	0.1	1.2
k,g,o	0.3	0.6	0.9
g,o,Ø	−0.5	−0.9	0.1
max p	0.3	1.6	1.4

Ø,t,d	−0.6	0.2	1.4
t,d,r	−1.0	1.6	−1.0
d,r,t	−0.5	−0.1	0.8
r,t,k	−3.6	0.3	0.3
t,k,g	−0.2	0.1	1.2
k,g,o	0.3	0.6	0.9
g,o,Ø	−0.5	−0.9	0.1
ave p	−0.87	0.26	0.53

max pooling
↳ like doing feature detector
eg. going to know, does this match anywhere in this piece of text is somewhere it using an "I" word

ave pooling
↳ we can think of it as a tool that measure some quality of the text
eg. casualness, learnedness...

③ Some notions

Other (maybe less useful) notions: stride = 2

Ø	0.0	0.0	0.0	0.0
tentative	0.2	0.1	−0.3	0.4
deal	0.5	0.2	−0.3	−0.1
reached	−0.1	−0.3	−0.2	0.4
to	0.3	−0.3	0.1	0.1
keep	0.2	−0.3	0.4	0.2
government	0.1	0.2	−0.1	−0.1
open	−0.4	−0.4	0.2	0.3
Ø	0.0	0.0	0.0	0.0

Ø,t,d	−0.6	0.2	1.4
d,r,t	−0.5	−0.1	0.8
t,k,g	−0.2	0.1	1.2
g,o,Ø	−0.5	−0.9	0.1

Local max pool, stride = 2

Ø	0.0	0.0	0.0	0.0
tentative	0.2	0.1	−0.3	0.4
deal	0.5	0.2	−0.3	−0.1
reached	−0.1	−0.3	−0.2	0.4
to	0.3	−0.3	0.1	0.1
keep	0.2	−0.3	0.4	0.2
government	0.1	0.2	−0.1	−0.1
open	−0.4	−0.4	0.2	0.3
Ø	0.0	0.0	0.0	0.0

Ø,t,d	−0.6	0.2	1.4
t,d,r	−1.0	1.6	−1.0
d,r,t	−0.5	−0.1	0.8
r,t,k	−3.6	0.3	0.3
t,k,g	−0.2	0.1	1.2
k,g,o	0.3	0.6	0.9
g,o,Ø	−0.5	−0.9	0.1
Ø	−Inf	−Inf	−Inf

Apply 3 **filters** of size 3

3	1	2	−3
−1	2	1	−3
1	1	−1	1

1	0	0	1
1	0	−1	−1
0	1	0	1

1	−1	2	−1
1	0	−1	3
0	2	2	1

Ø,t,d,r	−0.6	1.6	1.4
d,r,t,k	−0.5	0.3	0.8
t,k,g,o	0.3	0.6	1.2
g,o,Ø,Ø	−0.5	−0.9	0.1

conv1d, *k*-max pooling over time, *k* = 2

Ø	0.0	0.0	0.0	0.0
tentative	0.2	0.1	−0.3	0.4
deal	0.5	0.2	−0.3	−0.1
reached	−0.1	−0.3	−0.2	0.4
to	0.3	−0.3	0.1	0.1
keep	0.2	−0.3	0.4	0.2
government	0.1	0.2	−0.1	−0.1
open	−0.4	−0.4	0.2	0.3
Ø	0.0	0.0	0.0	0.0

Ø,t,d	−0.6	0.2	1.4
t,d,r	−1.0	1.6	−1.0
d,r,t	−0.5	−0.1	0.8
r,t,k	−3.6	0.3	0.3
t,k,g	−0.2	0.1	1.2
k,g,o	0.3	0.6	0.9
g,o,Ø	−0.5	−0.9	0.1

Other somewhat useful notions: dilation = 2

Ø	0.0	0.0	0.0	0.0
tentative	0.2	0.1	−0.3	0.4
deal	0.5	0.2	−0.3	−0.1
reached	−0.1	−0.3	−0.2	0.4
to	0.3	−0.3	0.1	0.1
keep	0.2	−0.3	0.4	0.2
government	0.1	0.2	−0.1	−0.1

Ø,t,d	−0.6	0.2	1.4
t,d,r	−1.0	1.6	−1.0
d,r,t	−0.5	−0.1	0.8
r,t,k	−3.6	0.3	0.3
t,k,g	−0.2	0.1	1.2
k,g,o	0.3	0.6	0.9
g,o,Ø	−0.5	−0.9	0.1

keep	0.2	−0.3	0.4	0.2
government	0.1	0.2	−0.1	−0.1
open	−0.4	−0.4	0.2	0.3
ø	0.0	0.0	0.0	0.0

Apply 3 **filters** of size 3

k,g,o	0.3	0.6	0.9
g,o,ø	−0.5	−0.9	0.1

2-max p	0.3	1.6	1.4
	−0.2	0.6	1.2

open	−0.4	−0.4	0.2	0.3
ø	0.0	0.0	0.0	0.0

Apply 3 **filters** of size 3

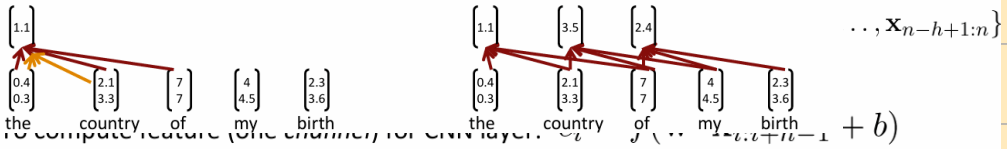
1,3,5	0.3	0.0
2,4,6		
3,5,7		

④ Example : Single Layer CNN for sentence Classification

① process

- Goal: Sentence classification:
  - Mainly positive or negative sentiment of a sentence
  - Other tasks like:
    - Subjective or objective language sentence
    - Question classification: about person, location, number, ...

- Word vectors: $\mathbf{x}_i \in \mathbb{R}^n$
- Sentence: $\mathbf{x}_{1:n} = \mathbf{x}_1 \oplus x_2 \oplus \cdots \oplus \mathbf{x}_n$ (vectors are concatenated)
- Filter applied to concatenation of words in range: $\mathbf{x}_{i:i+j}$ (symmetric more common)
- Convolutional filter $\mathbf{w} \in \mathbb{R}^{hk}$ applied to all possible windows $\{\mathbf{x}_{1:h}, \mathbf{x}_{2:h+1}, \ldots, \mathbf{x}_{n-h+1:n}\}$
  - Filter is done as a long vector over window of *h* words
  - Filter could be of size *h* = 2, 3, or 4 words
- To compute feature (one *channel*) for CNN layer: $c_i = f(\mathbf{w}^T \mathbf{x}_{i:i+h-1} + b)$
- Result is a feature map: $\mathbf{c} = [c_1, c_2, \ldots, c_{n-h+1}] \in \mathbb{R}^{n-h+1}$



- Result is a feature map: $\mathbf{c} = [c_1, c_2, \ldots, c_{n-h+1}] \in \mathbb{R}^{n-h+1}$

Pooling, channels, and classification

- Pooling: max-over-time pooling layer
- Idea: capture most important activation (maximum over time)
- Use multiple filter weights **w** (i.e., multiple channels)
- From feature map $\mathbf{c} = [c_1, c_2, \ldots, c_{n-h+1}] \in \mathbb{R}^{n-h+1}$
- Pooled single number: $\hat{c} = \max\{\mathbf{c}\}$
  - Because of max pooling $\hat{c} = \max\{\mathbf{c}\}$, length of **c** can be variable

- One convolution layer, followed by one max-pooling
  - To obtain final feature vector: (assuming *m* filters **w**) $\mathbf{z} = [\hat{c}_1, \ldots, \hat{c}_m]$
  - Used 100 feature maps each of sizes 3, 4, 5

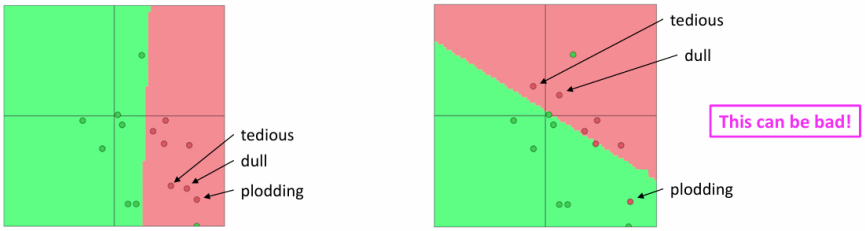
- Simple final softmax layer : $y = softmax\left(W^{(S)} z + b\right)$

② A problem experimentalist met

A pitfall when fine-tuning word vectors

- **Setting:** We are training a model for movie review sentiment building on word vectors
- In the training data we have “tedious”, “dull”; in the testing data we have “plodding”
- The pre-trained word vectors have all three similar:

- **Question: What happens when we update the word vectors?**
- **Answer:** Words in the training data move around; other words stay where they were



the classification boundary has moved!

Because Backward propagation has been done, these which in training data will move, But some words just won't be in your training data, and they're going to stay exactly where they were in word2vec vectors

A solution: Channel doubling multi-channel input idea

- Initialize model with pre-trained word vectors (e.g., word2vec or Glove)
- Start with two copies
- Backprop into only one set, keep other "static"
  - Fine-tuning should be useful for improving word vectors for task
  - But there is a problem that words in pre-training (and maybe runtime data) but not in training data will not move. So, it also makes sense to leave all word vectors where they are and to only update the parameters above the word vectors
  - Having two copies is an attempt to get the best of both worlds
- Both channel sets are added to c_i before max-pooling

(3) Summary

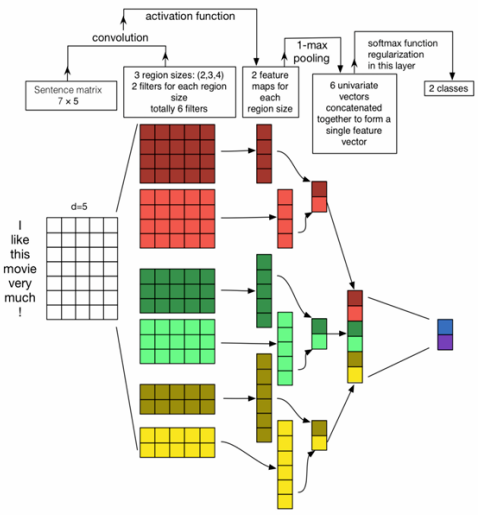
Kim (2014)

From:

Zhang and Wallace (2015) A Sensitivity Analysis of (and Practitioners' Guide to) Convolutional Neural Networks for Sentence Classification

https://arxiv.org/pdf/1510.03820.pdf

(follow on paper, not famous, but a nice picture)



(5) Model comparison

4. Model comparison: Our growing toolkit

- **Bag of Vectors**: Surprisingly good baseline for simple classification problems
  - Especially if followed by a few ReLU layers! (See paper: Deep Averaging Networks)
- **Window Model**: Good for single word classification for problems that do not need wide context. E.g., POS, NER
- **CNNs:** good for classification, need zero padding for shorter phrases, somewhat implausible/hard to interpret, **easy to parallelize on GPUs;** efficient and versatile
- **Recurrent Neural Networks**: Cognitively plausible (reading from left to right), not best for classification (if just use last state), much slower than CNNs, good for sequence tagging and classification, good for language models, better with attention
- **Transformers:** Great for language models, great for sentence calculations; in general, still the best thing since sliced bread for all NLP problems
  - "Vision Transformers" are taking over in vision but some papers argue that CNNs

and transformers have complementary advantages, and you can usefully use both

⑥ Batch Normalization

Batch Normalization (BatchNorm)

[Ioffe and Szegedy. 2015. Batch normalization: Accelerating deep network training by reducing internal covariate shift. arXiv:1502.03167.]

- Often used in CNNs
- Transform the convolution output of **a batch** by scaling the activations to have zero mean and unit variance
  - Again, like the familiar Z-transform of statistics
  - Related to LayerNorm, which is standard in Transformers, but crucially different:
    - LayerNorm calculates statistics across all feature dimensions for each instance independently
    - BatchNorm normalizes across all elements and items in a batch for each feature independently
- Use of BatchNorm also makes models **much** less sensitive to parameter initialization, since outputs are automatically rescaled
  - It also tends to make tuning of learning rates simpler
- PyTorch: nn.BatchNorm1d

⑦ 1×1 convolution

Size 1 Convolutions

[Lin, Chen, and Yan. 2013. Network in network. arXiv:1312.4400.]

- **Does this concept make sense?!? Yes.**
- Size 1 convolutions ("1x1"), a.k.a. Network-in-network (NiN) connections, are convolutional kernels with kernel_size=1
- A size 1 convolution gives you a fully connected linear layer across channels!
- It can be used to map from many channels to fewer channels
- Size 1 convolutions add additional neural network layers with very few additional parameters
  - Unlike Fully Connected (FC) layer across data item which adds **tons** of parameters
  - This is similar to the per-position feed-forward layers in transformers

also, compare with 3×3 Cov, 1×1 doesn't integrate spatial information

⑧ Example 2: very Deep Convolutional networks for Text classification

① background

- Conneau, Schwenk, Lecun, Barrault. EACL 2017.
- Starting point: sequence models (LSTMs) had been very dominant in NLP
  - Also CNNs, Attention, etc., but all the models were basically not very deep – not like the deep models in Vision
- What happens when we build a vision-like system for NLP?
- Model works up from the character level
  - Desire for "NLP from scratch" [raw signal]

② architecture

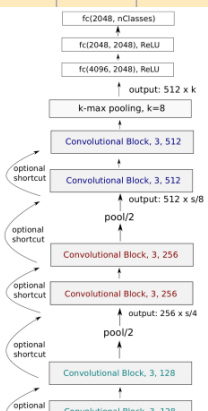
VD-CNN architecture

The system very much looks like a vision system in its design, similar to VGGnet or ResNet

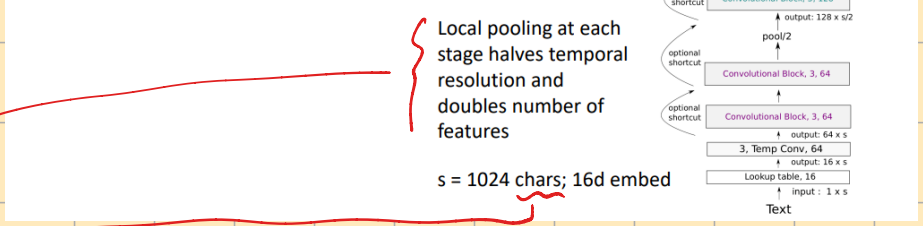
It looks unlike then typical Deep Learning NLP systems

- It looks a bit more like a Transformer?

Result is constant size, since text is truncated or padded



← full connection + ReLU



Why? Intuition is that if you are "compressing", you must "widen"

Representation $\in \mathbb{R}^{T \times C}$

↳ information content $= T \times C$

$\downarrow$

$\frac{T}{2} \times C$

$\downarrow$

$\frac{T}{2} \times 2C = T \times C$

Like pixels in Vision, here are individual characters

Tree recursive neural networks (TreeRNN)

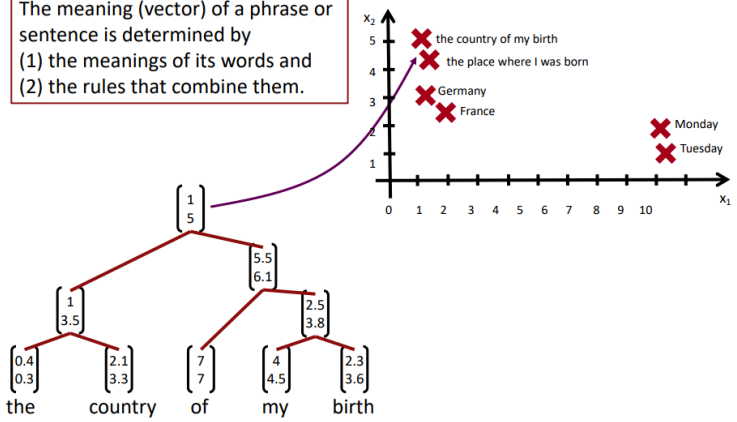
↘ not recurrent

①

How should we map phrases into a vector space?

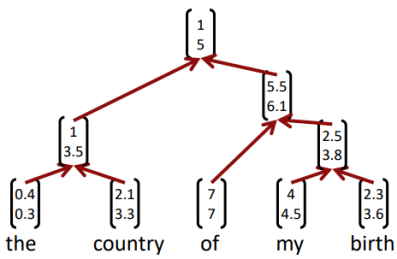
Socher, Manning, and Ng. ICML, 2011

Use principle of compositionality
The meaning (vector) of a phrase or sentence is determined by (1) the meanings of its words and (2) the rules that combine them.

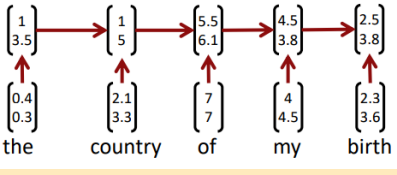


Recursive vs. recurrent neural networks

- Recursive neural nets provide representations for linguistic phrases
- But they require a tree structure

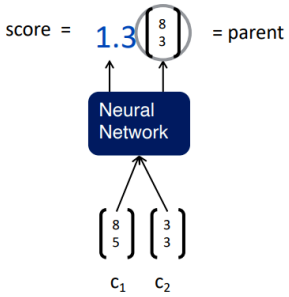


- Recurrent neural nets cannot capture phrases without prefix context
- They often capture too much of last words in "phrase" vector



② Defination

Simple Tree Recursive Neural Network Definition



score = $U^T p$

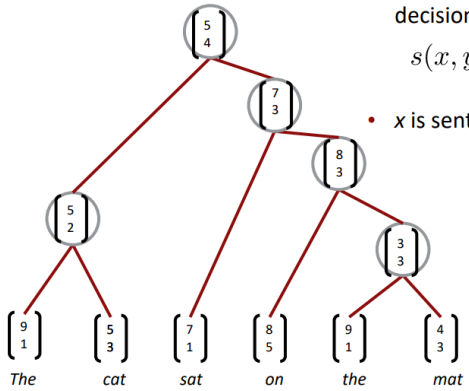
$p = \tanh(W\begin{bmatrix} c_1 \\ c_2 \end{bmatrix} + b)$,

Same W parameters at all nodes of the tree → like RNNs

U^T: another vector of learned parameters,
By dot producting with the representation of father vector,
to score whether it's a good constituent.

③ parsing (greedily)

Parsing a sentence



- The score of a tree is computed by the sum of the parsing decision scores at each node:

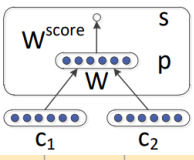
$$s(x, y) = \sum_{n \in nodes(y)} s_n$$

- x is sentence; y is parse tree

④

Discussion: Simple TreeRNN

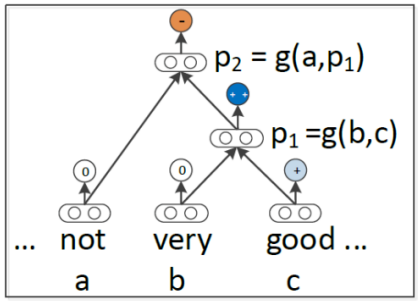
- We got some decent results with a single layer TreeRNN like this!
  - [Socher, Manning, and Ng. ICML, 2011] got a best paper award!
- A single weight matrix TreeRNN could capture some things but not more complex, higher order composition and parsing long sentences
- There is no real interaction between the input words
- And the composition function is the same for all syntactic categories, punctuation, etc.

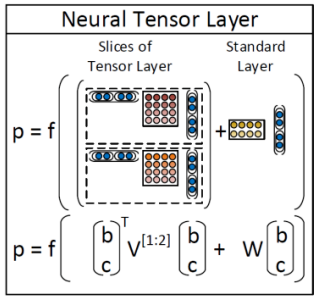


→ just sort of uniformly computing things

⑤ Example: Recursive Neural Tensor Networks

- Allows two word or phrase vectors to interact multiplicatively





- Not today, but see also Tai, Socher, Manning [2015]: TreeLSTMs
  - Work even better

people use it in Sentiment detection

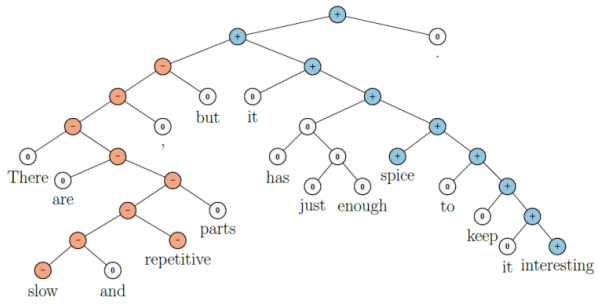
why use it in Sentiment detection?

↳ even today, quite a few people's Sentiment analysis systems are essentially just keyword matching

order to deal with complex sentence with complex sentiment, people camp up with the Stanford Sentiment Treebank

Experimental Results on Treebank

- RNTN can capture constructions like *X but Y*
- RNTN accuracy of 72%, compared to MV-RNN (65%), biword NB (58%) and RNN (54%)



↳ it seems would like to see better sentence understanding

So one of the things that neural networks, when looking at language, have often been faulted for and still faulted for to this day using Transformer models, is you often find the result that neural network models just don't pay attention to negation

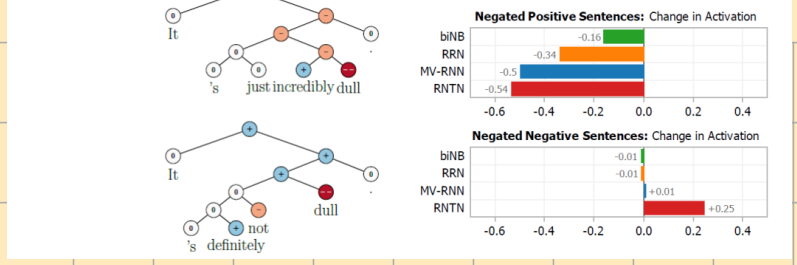
eg: The drug causes cancer
The drug does not cause cancer

↳ for human, the meaning of these two sentences are completely opposite

But for Transformer, share almost all tokens in two sentences, vector representation is very close in embedding / hidden state

Negation Results

When negating negatives, positive activation should increase!



(6) Conclusion

The reason that TreeRNN became uncompetitive is because they just didn't allow the kind of associations and information flow that we have in Transformer. Its only information flow was tree structured following the context free backbone. But on the other hand, to the extent that you actually want to model the sort of semantics of human language, in some sense, these model were more right.