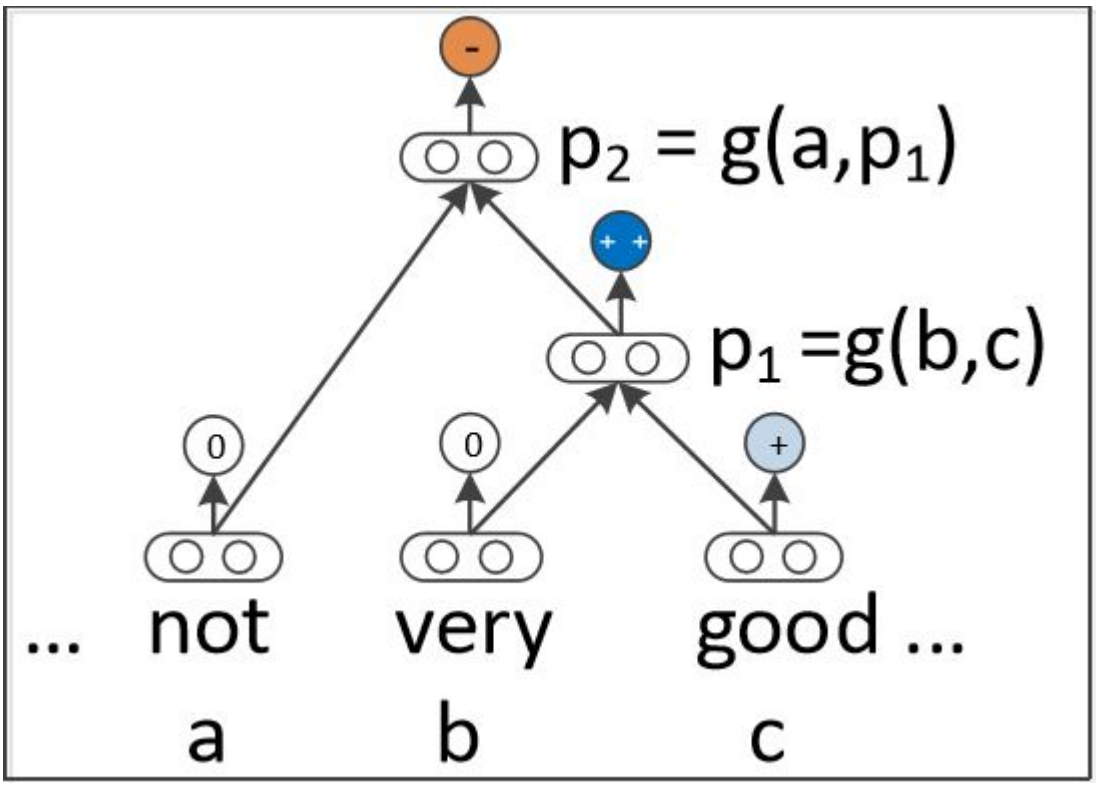
RNTN을 이용한 Sentimental Semantic 분석

Recursive Neural Models



- n-gram이(n은 단어 갯수) compositional model에 주어졌을 때, vector로 표현되는 단어들이 binary tree와 vector로 표현되는 각각의 leaf node로 parsing 된다.
- 모든 단어는 uniform distribution으로 랜덤하게 초기화된다.

*uniform distribution: 특정 범위에서 균일하게 나타나는 분포.

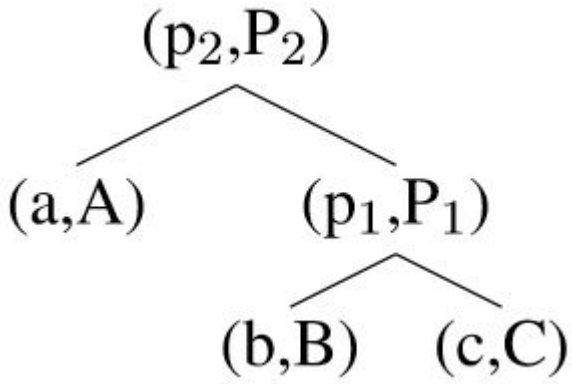
- Recursive neural models은 다른 input(자식 vector)을 가지는 함수 g를 이용해 parent vector를 계산한다. parent vector는 또 다른 함수의 input, 즉 feature로 제공된다.
- 이처럼 자식 vector가 합쳐져 부모 vector가 되고, 다시 자식 vector가 되어 또 다른 자식 vector와 합쳐지기 때문에 **Recursive Neural Models**라고 하는 것이다.

이전 모델들

Recursive Neural Network

- 부모 vector는 자식 vector를 가지고 있다. 즉, 부모 단어는 자식 단어들의 특성을 띄고 있는 것이다.
- RNN의 기본적인 모델이라고 생각하면 될 것 같다.

MV-RNN: Matrix-Vector RNN



- 모든 단어와 긴 구문을 vector와 matrix로 된 parse tree로 나타내는 것이 주된 아이디어이다.
- 각각의 n-gram은 (vector, matrix) pair인 리스트로 나타나진다.

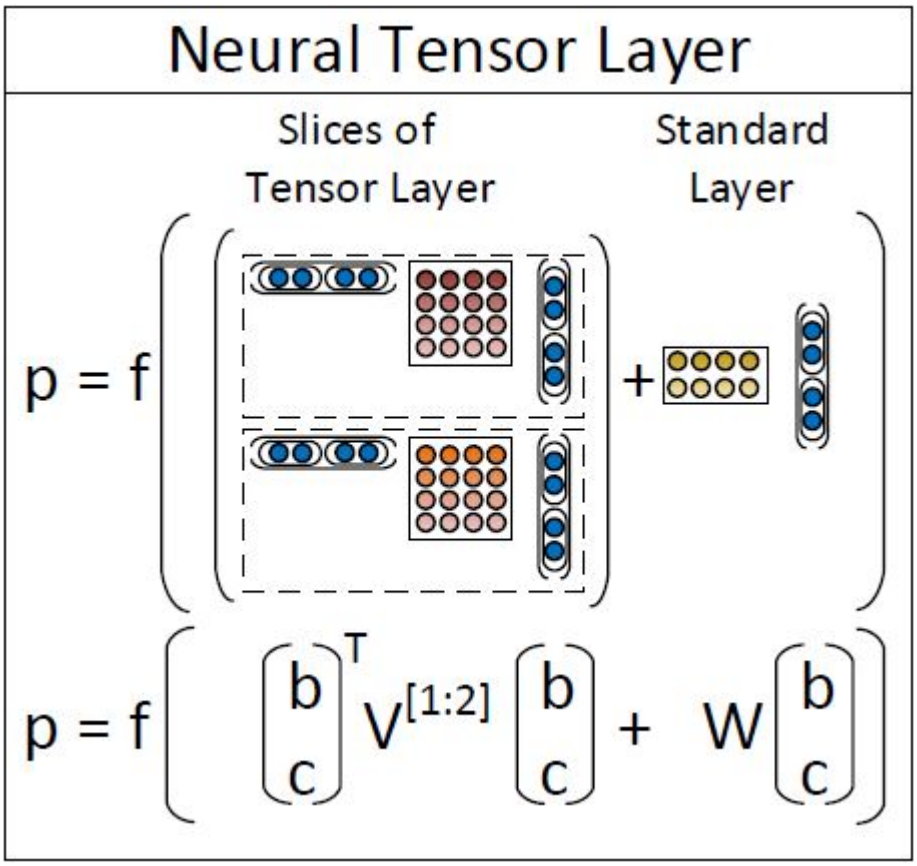
$$p_1 = f\left(W\begin{bmatrix} Cb \\ Bc \end{bmatrix}\right), P_1 = f\left(W_M\begin{bmatrix} B \\ C \end{bmatrix}\right),$$

(소문자로 된 것은 vector, 대문자로 된 것은 matrix)

- 단어 혹은 구문인 두 개의 구성요소가 결합될 때, 구성요소를 나타내는 matrix는 다른 구성요소의 vector와 곱해진다. (Cb, Bc)
- 각각의 matrix는 dxd의 matrix로 초기화되며, WM matrix는 dx2d의 matrix로서 곱해지면 다시 dxd가 된다. (matrix가 dxd로 유지.)

RNTN: Recursive Neural Tensor Network

- 기존의 standard RNN, MV-RNN에 문제가 있으므로, 더 단순하고 좋은 composition function이 있을까에 대한 의문에서 출발
- 아이디어는 모든 노드에 tensor 기반의 composition function을 쓰는 것이다.



(위 그림과 아래 그림에 나오는 a, b, c는 Recursive Neural Models에 있는 것을 가리킨다.)

- V는 2dx2d를 d개 가지는 즉, [2dx2dxd]인 tensor이다. V는 composition의 특정 타입을 담고있다.

  (1차원 = vector, 2차원 = matrix, 3차원 = tensor)
- 단순히 W[b c]만 하면 단어 간의 복잡한 관계를 표현하지 못한다. 그래서 파라미터를 추가해 주기 위해 V를 넣은 것이다. 즉, W[b c]만 하면 [b c]에 대한 1차 식이지만, [b c]^T*V*[b c] 를 하면 [b c]끼리 곱해지게 되는 것이다. 이는 [b c]에 대한 2차 식으로 나타나진다. (쉽게 이해하자면 x1 과 x2에 대한 1차식에서 2차식으로 변화시켜주기 위한 것이다.)

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

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

- p1과 p2의 W는 같다. p1은 또 다른 인풋으로 들어가게 된다.
- V가 0으로 설정됐을 때, V는 직접적으로 input vector와 관련을 맺는다. (standard layer 값만 남게 되므로. 즉, 기존 모델과 똑같다.)

Tensor Backprop through Structure

- target vector와 predicted vector는 one-hot encoding으로 돼있다.
- 각각의 노드는 softmax classifier를 갖는다.

- target vector와 predicted vector의 차이를 줄여야 한다. 즉, KL-divergence(Kullback Leibler divergence)를 최소화시켜야 한다.
- 각각의 노드는 weights V,W을 반복적으로 사용해 backprop한다.

$$\delta^{i,s} = \left(W_s^T(y^i - t^i)\right) \otimes f'(x^i)$$

- 노드 i에서의 softmax error vector는 위와 같이 구해진다.

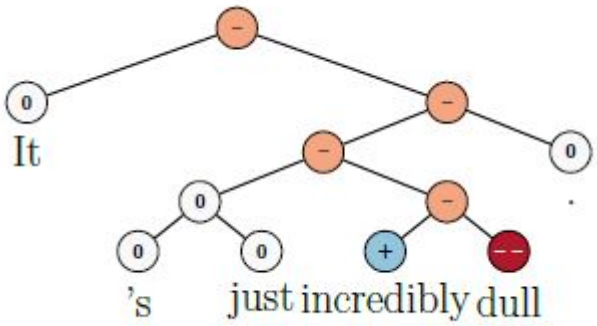
1. 노드 i의 predicted vector와 target vector를 빼준 결과를, softmax하고 transpose한 W와 곱해준다.
2. 노드 i의 vector(a,b,c,p1,p2)를 tanh인 f에 파라미터로 주고 미분한다.
3. 1의 결과와 2의 결과를 Hadamard product해준다.

   *Hadamard product: 두 행렬의 element중 행과 열이 모두 같은 것 끼리 더해주는 것. 예를 들면 A행렬의 1행 1렬은 B행렬의 1행 1렬과 곱해주는 것이다.

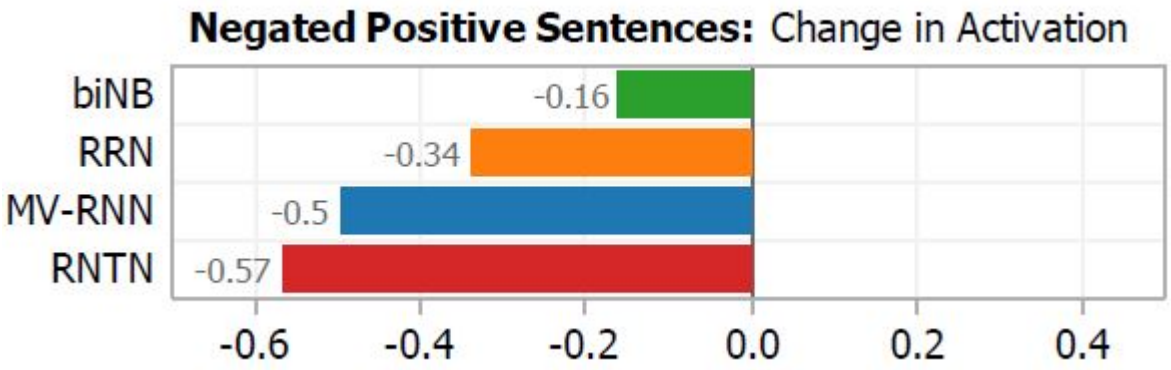
- 나머지 미분은 top-down 방식으로만 계산될 수 있다.
- V와 W에 대한 미분은 각각의 노드의 미분의 합이다.
- top 노드(부모 노드)는 top 노드의 softmax로부터만 에러를 받는다. 즉, top 노드의 에러는 top노드의 softmax error인 것이다.
- leaf 노드(자식 노드)는 error를 계산할 때, 자신의 softmax error와 부모 노드의 error 절반을 더한다. 만약 부모 노드의 왼쪽에 위치해 있다면 절반의 첫번째를 더하고, 오른쪽에 위치해 있다면 절반의 두번째를 더한다. (아마 slice의 윗부분과 아랫부분 아닐까 생각함...)

Model Analysis: High Level Negation

Set 1: Negating Positive Sentences.

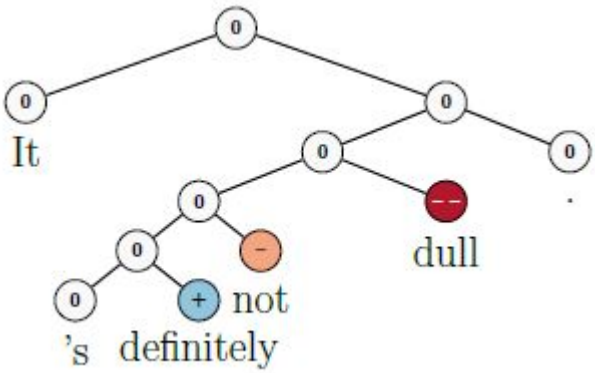


- "It's not good."과 같이 긍정적인 문장들과 그 문장들의 부정이 포함되어 있었다. 부정이 전반적인 sentiment를 긍정에서 부정으로 바꿔 놓았다.
- RNTN은 'least'와 같은 덜 명확한 표현이 있을 때도 분류할 수 있었다.
- 즉, 하나의 단어를 제외하고 모든 단어가 긍정을 나타날 때도, 그 하나의단어가 부정적인 단어라면 문장의 sentiment가 부정으로 바뀌었다.

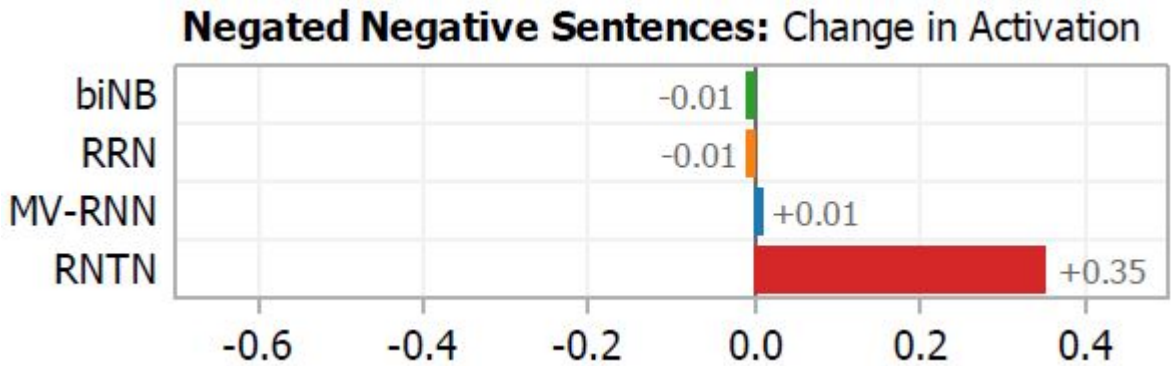


- 긍정적인 문장이 부정적인 단어로 인해 부정적인 문장으로 변할 때, RNTN이 좀 더 positive sentiment를 감소시켰다.

Set 2: Negating Negative Sentences.



- "It's not bad."와 같이 부정적인 문장과 그 문장들의 부정이 포함되어 있었다.
- RNTN을 이용한 sentiment treebank에서는 이러한 문장들을 덜 부정적인 문장으로 처리했다. 부정적이지 않다고 해서 긍정적인 문장은 아니기 때문이다.



- RNTN만 정확하게 부정적인 문장이 부정 표현으로 인해 덜 부정적인 문장으로 바뀌는 것을 감지해낼 수 있었다. neutral activation과 positive activation을 증가시키는 것이다.

Conclusion

- sentimental semantic을 분석하는 데는 RNTN이 현재 짱짱이다.
- 긍정과 부정표현을 구분해낼 뿐만 아니라, 긍정 표현이 부정 표현이 되는 것과 부정 표현이 중립 표현이 되는 것 또한 정확히 포착해낼 수 있다.

Reference

- Recursive Deep Models for Semantic Compositionality Over a Sentiment Treebank Socher et al. 2013. Introduces Recursive Neural Tensor Network. Uses a parse tree.
- Yoav Goldberg. 2015. A Primer on Neural Network Models for Natural Language Processing. 5, October

Q&A

1. 단어를 어떻게 벡터와 매트릭스로 나타내는가?

- 다양한 방법이 있다. 해당 문장에 단어가 있다면 1, 그렇지 않다면 0 이런 식으로도 할 수 있다. 하지만 대부분의 경우 word2vec이라는 것을 사용한다. word2vec이 어떻게 작동하는지 자세하게 알고 싶다면 다음의 링크를 참조하기 바란다.

http://khanrc.tistory.com/entry/TensorFlow-7-word2vec-Implementation

- 해당 논문에서는 새로운 코퍼스에서 supervised word representation이 이루어졌다고 한다. 단어를 label 존재하는 지도 학습을 통해 벡터화 시킨 것 같다.

2. 단어에 따로 감정 레이블이 붙어있는가?

- stanford sentiment treebank를 이용해, 수천 개의 문장에서 문법적으로 그럴 듯한 모든 phrase에 긍정 부정의 label이 붙어있다.