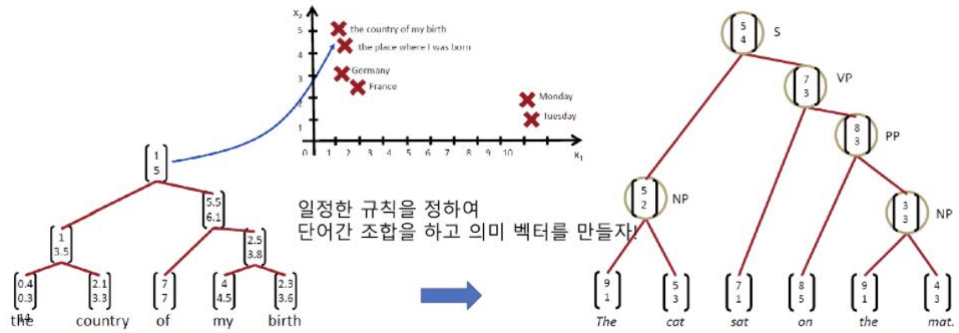
Lecture 18. Tree Recursive Neural Networks, Constituency Parsing, and Sentiment

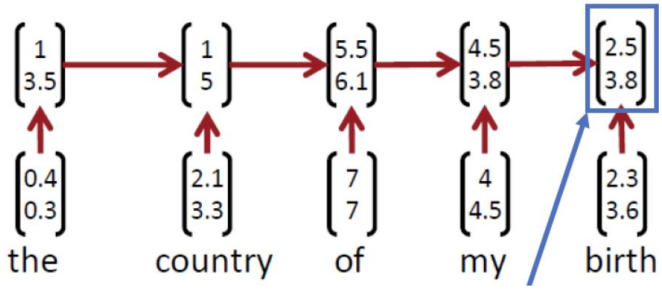
Ⅰ. Constituency Parsing

Compositionality : 각 단어의 조합을 통해 새로운 의미를 나타내고 단어의 조합으로부터 문장의 의미를 파악할 수 있음.

큰 구절을 vector space에 mapping시키는 방법은 우선 문장에서 각 단어의 조합을 TreeRNN을 통해 추출한 값을 저장하고 일정한 규칙을 통하여 문장의 의미 벡터를 추출하는 것이다.

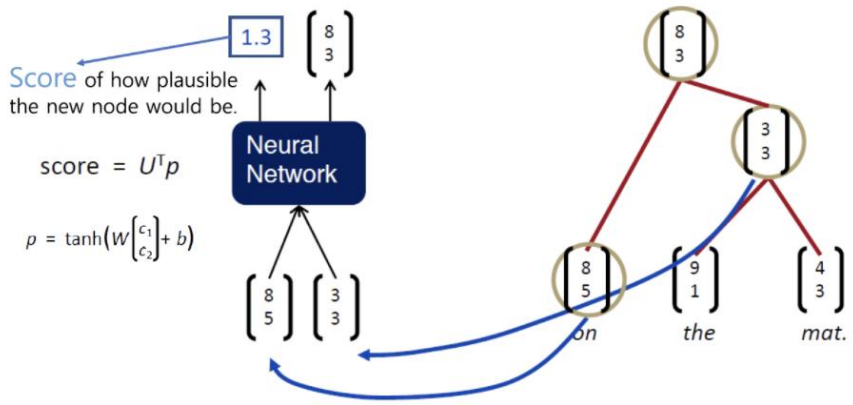


RNN을 통해 문장을 분석하는 경우 아래와 같이 인접 단어를 합친 단어의 의미를 충분히 반영하지 못한다. 또한 마지막 단어 벡터에 주목하는 경향이 있기 때문에 RNN보다 TreeRNN이 문장의 의미를 파악하는데 유용하다.

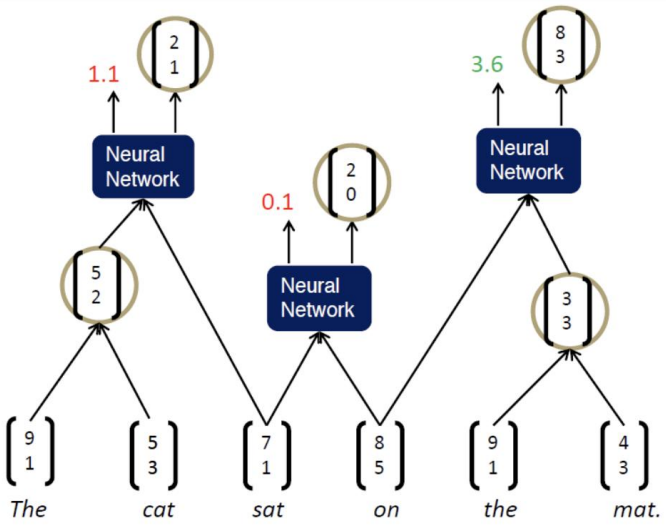


Ⅱ. Simple TreeRNN

'on the mat' 문장의 의미 벡터를 얻기 위해 'the'와 'mat' 조합 단어 벡터를 계산한 후 나머지 'on'과 조합하는 과정을 거쳤다. 먼저 봐야할 벡터는 부모 벡터값으로 아래 그림의 수식을 통해 벡터값을 계산하게 되며 W 행렬은 모든 TreeRNN에서 동일하다.



Score은 조합할 단어를 선택할 때 반영되는 값이다. 해당 단어가 얼마나 그럴 듯 한지를 의미하고 이후 score 값을 통해 문장의 단어들을 조합해간다. 해당 단어들의 조합으로 부모 단어의 벡터와 score값을 얻고 난 이후에는 greedy algorithm을 통해 다음 단어를 조합하게 된다.



Simple TreeRNN의 한계점은 아래와 같다.

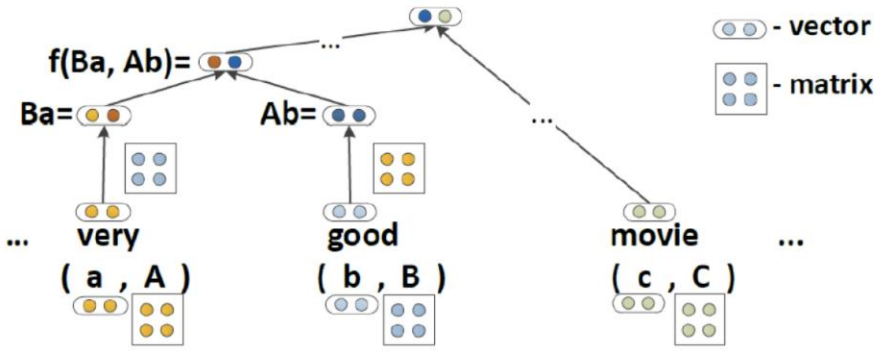
1. W가 모든 노드에서 동일한데 Simple TreeRNN은 일부 현상에선 적합할 수 있으나 더 복잡하고 고차 구성 및 긴 문장에서는 적절하지 못하다.
2. 인풋 단어 간에 실제 상호작용이 없다.
3. 조합 함수가 모든 경우에 대해서 동일하게 작용한다.

Ⅲ. Syntatically-United RNN

Simple RNN의 한계점을 개선한 모델이다. 모든 조합에 똑같이 사용되었던 행렬 W를 각기 다른 행렬로 설정한다. Probabilistic Context Free Grammar(PCFG)를 이용하여 조합될 단어의 확률을 계

산하고 TreeRNN에 적용한다.

1. Matrix-Vector RNN

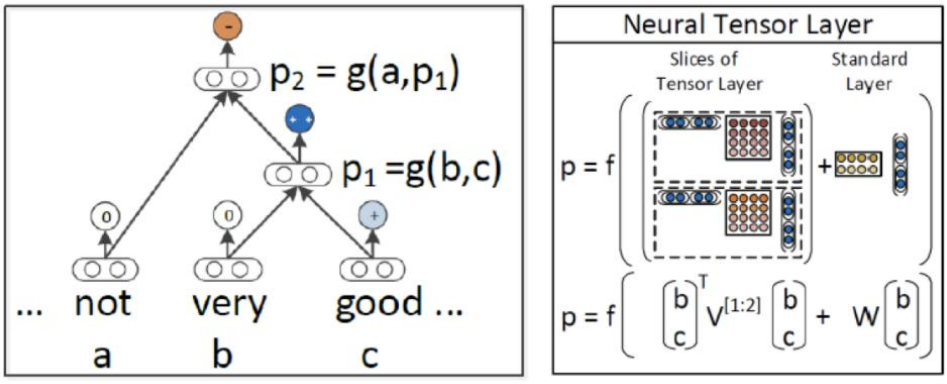


$$P = g(A, B) = W_M \begin{bmatrix} A \\ B \end{bmatrix}$$

이전 모델들과 다르게 단어들이 벡터 정보만 가지고 있는게 아닌 행렬에 대한 정보도 같이 가지고 있다. 해당 모델에서는 단어가 지니는 정보를 하나 더 가지게 함으로써 문장의 의미를 더욱 잘 파악할 수 있도록 한다. 'very'의 단어벡터와 'good'의 행렬 반대로 'good'의 단어벡터와 'very'의 행렬이 곱연산 되어 부모 노드로 전달된다.

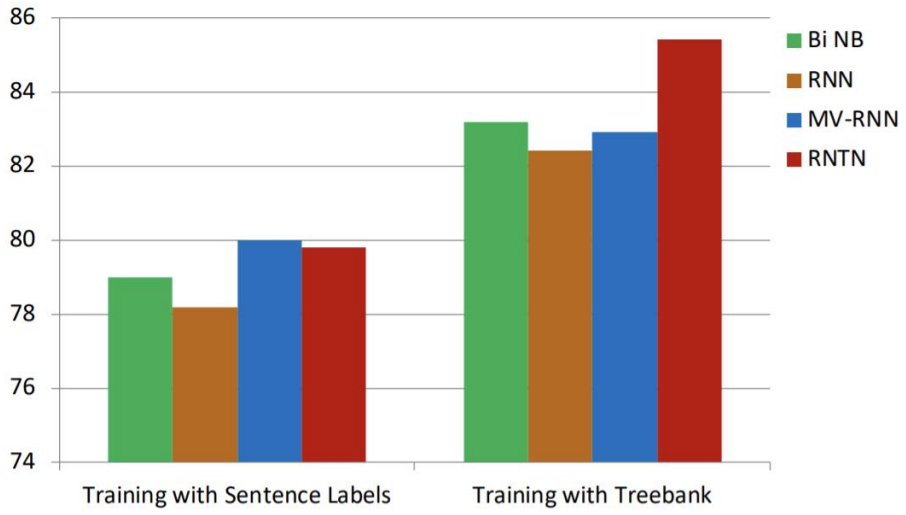
2. Recursive Neural Tensor Network

TreeRNN을 통해 단어, 문장을 분석하지 않고 Bag of Words를 통해서 임베딩하여 문장을 분석하여도 90%의 성능을 보인다. 하지만 'should have been'과 같은 경우에는 후회와 긍정은 부정을 의미하기에 해당 모델로는 단어의 감정을 분석하기 힘들다.



해당 모델은 MV-RNN보다 적은 파라미터를 가진다. MV-RNN은 모든 결합에 대하여 각 노드마다 행렬을 생성하고 계속 연산을 진행해야 하기 때문에 cost가 높아질 수 밖에 없다. 해당 모델에서는 파라미터 수를 확실히 줄임으로써 MV-RNN보다 cost를 낮추었고 단

어의 채널을 통해서 의미를 분석하는데 사용한다. 긍/부정을 분석한 결과 RNTN이 다른 모델에 비해 높은 성능을 보인다. Treebank 데이터는 구조적 분석에서 각 단어에 모두 라벨링이 되어있다. 그러면 제일 간단한 Bi NB모델의 성능이 RNN, MV-RNN보다 높다.



현실적으로는 GPU 연산이 힘들기 때문에 TreeRNN을 사용하기 힘들다. 병렬적으로 연산이 진행되어야 하는데 모든 task 마다 연산 구조가 동일하지 않고 tree 모양이 다르기 때문이다. 또한 데이터에 라벨링을 모두 거쳐야 하는데 이 데이터를 구축하는데에 어려움이 있다.