Deep Learning

NLP & Syntax Net

2016.07.11

이우진

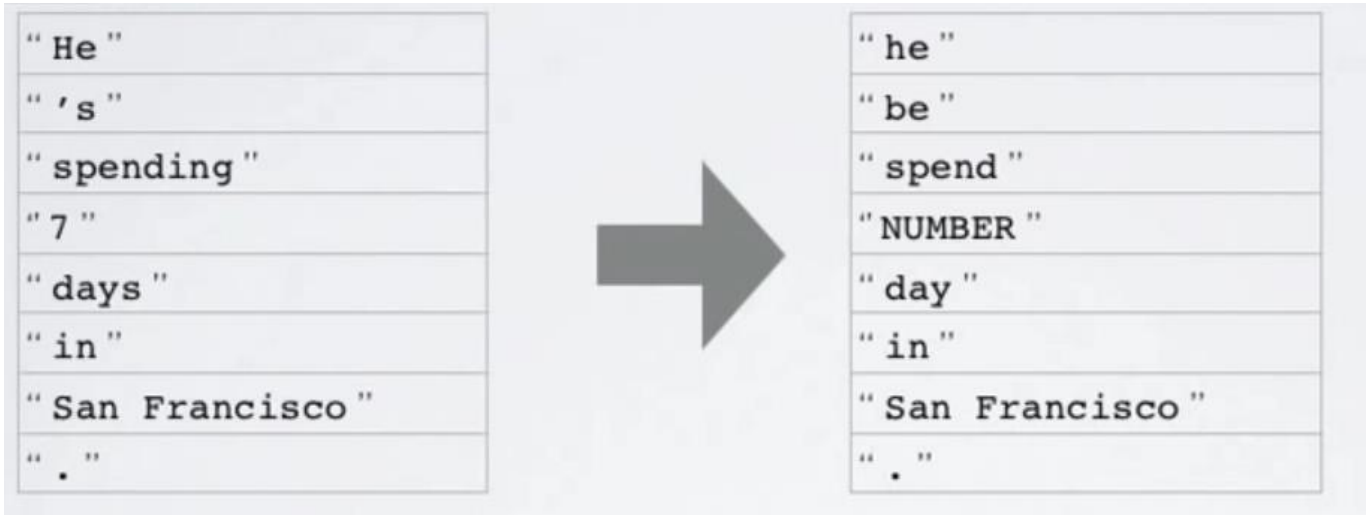
Natural Language Processing

- **NLP**
  - Language data를 다루는 task와 관련.
  - Text Data
- **Much like for computer vision, we can design neural networks specifically adapted to processing of text data**
  - High Dimensional

Preprocessing

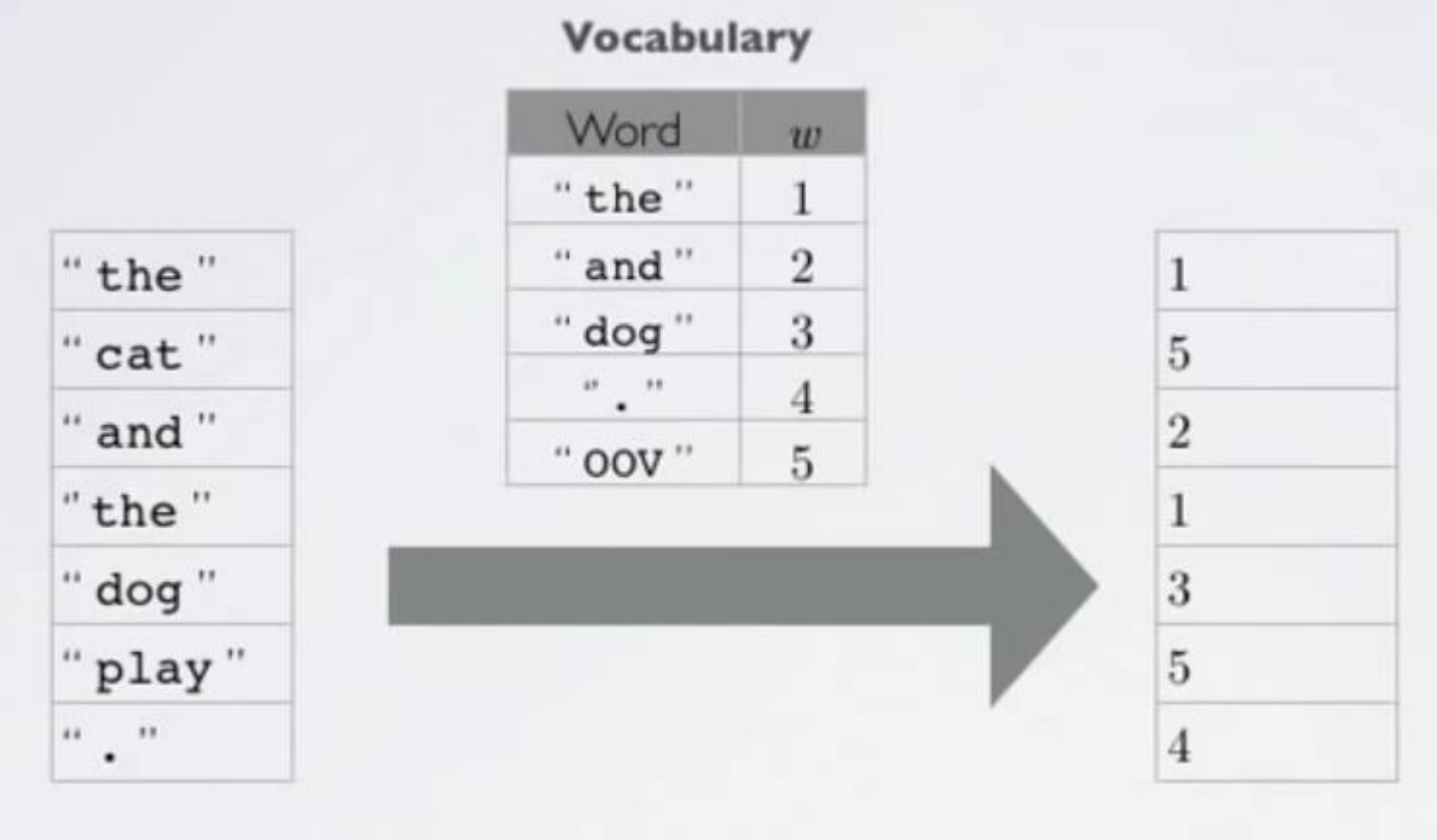
- **Tokenization**
  - Tokenize text.
  - Long string -> list of token strings

- **Lemmatization**
  - Standard form



Preprocessing

- **Vocabulary**
  - Unique ID
  - 단어마다 특정 ID를 부여한다.
  - 1000-250000개의 단어
  - The, a 같은것들은 무시
  - OOV = out of vocabulary
  - w로 단어를 표시



One-hot encoding

- **One-hot encoding of the ID**
  - ID를 제외하고는 모두 0으로 하고, ID만 1로 채운다.

ex.: for vocabulary size $D=10$, the one-hot vector of word ID $w=4$ is

$$e(w) = [\,0\;0\;0\;1\;0\;0\;0\;0\;0\;0\,]$$

  - 그러나 similarity에 대한 개념은 없다.

$$||e(w) - e(w')||^2 = 0 \text{ if } w = w'$$

$$||e(w) - e(w')||^2 = 2 \text{ if } w \neq w'$$

  - High dimensionality
  - Overfitting에 취약하다
  - 계산이 복잡하다
  - Easy but Poor

Word Representations

- **Continuous word representation**
  - 모든 단어 w가 실수 벡터 C(w)로 표현이 된다.
  - 벡터를 learning을 해야한다.
  - Distance가 similarity를 의미한다.
  - 10개의 단어를 표현하려면 벡터를 합친다.

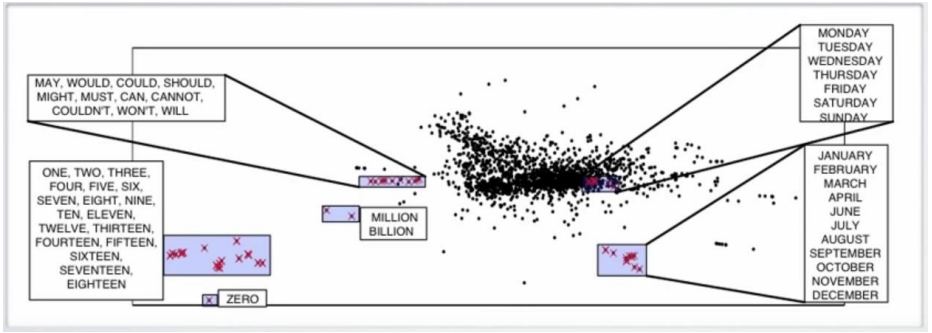
$$\mathbf{x} = [C(w_1)^\top, \ldots, C(w_{10})^\top]^\top$$

Word	w	$C(w)$
"the"	1	[0.6762, -0.9607, 0.3626, -0.2410, 0.6636]
"a"	2	[0.6859, -0.9266, 0.3777, -0.2140, 0.6711]
"have"	3	[0.1656, -0.1530, 0.0310, -0.3321, -0.1342]
"be"	4	[0.1760, -0.1340, 0.0702, -0.2981, -0.1111]
"cat"	5	[0.5896, 0.9137, 0.0452, 0.7603, -0.6541]
"dog"	6	[0.5965, 0.9143, 0.0899, 0.7702, -0.6392]
"car"	7	[-0.0069, 0.7995, 0.6433, 0.2898, 0.6359]

  - Representation을 gradient descent를 통해 학습

$$C(w) \Longleftarrow C(w) - \alpha \nabla_{C(w)} l$$

  - 뉴럴넷의 파라미터 뿐만 아니라 representation을 업데이트한다.
  - C라는 matrix에 단어 e(w)을 projection를 해서 C(w)를 구한다.



Language Modeling

- **Language Model**
  - Probabilistic model that assigns probabilities to any sequence of words.
  - 마르코브 가정을 기반으로 한다.

$$p(w_1, \ldots, w_T) = \prod_{t=1}^{T} p(w_t \mid w_{t-(n-1)}, \ldots, w_{t-1})$$

  - T번째 단어는 기본의 n-1단어들로 만들어 졌다.

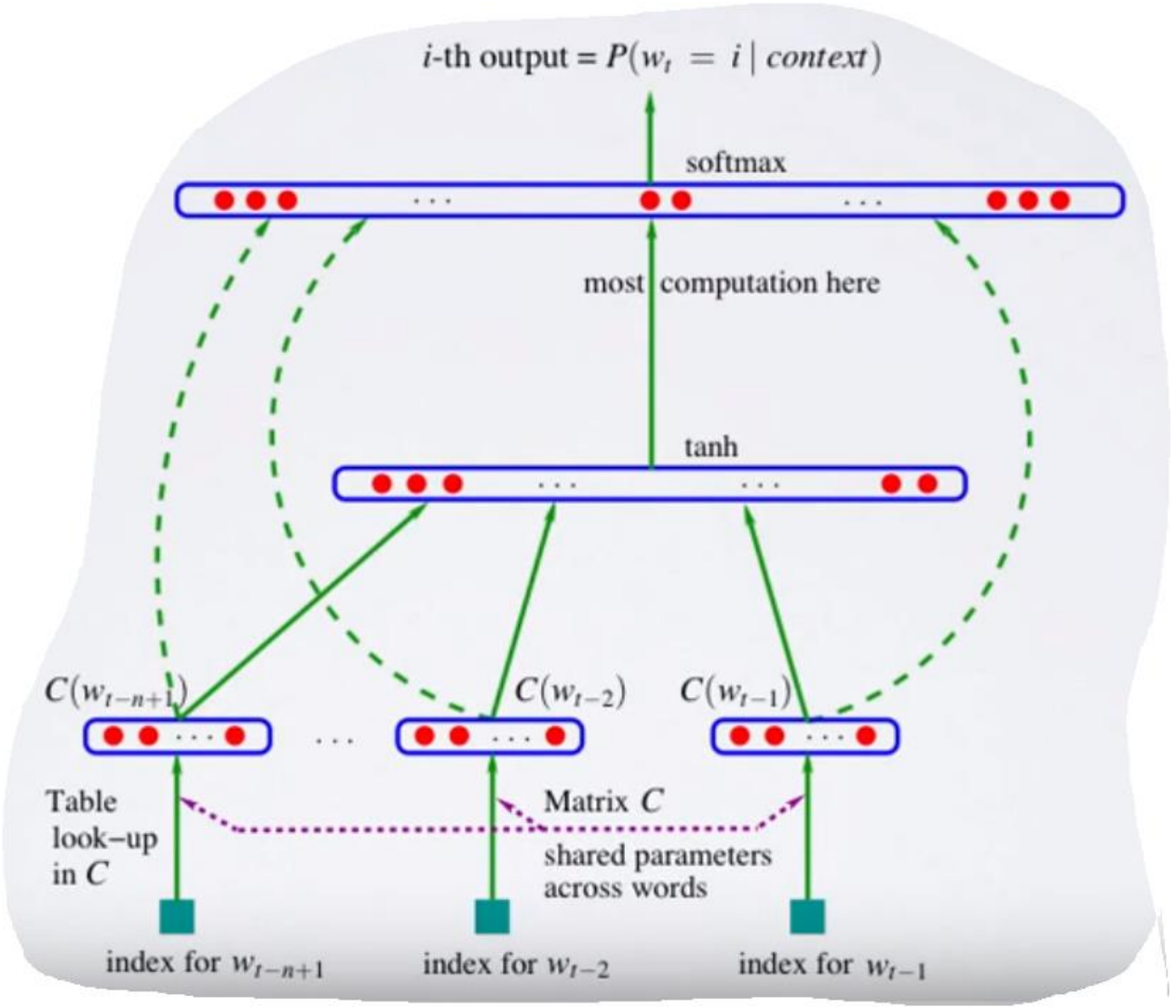
- **N-gram model**
  - N개의 단어들의 sequence
  - Conditional probability를 구한다.

- unigrams (n=1): "is", "a", "sequence", etc.
- bigrams (n=2): ["is", "a"], ["a", "sequence"], etc.
- trigrams (n=3): ["is", "a", "sequence"], ["a", "sequence", "of"], etc.

$$p(w_t \mid w_{t-(n-1)}, \ldots, w_{t-1}) = \frac{\text{count}(w_{t-(n-1)}, \ldots, w_{t-1}, w_t)}{\text{count}(w_{t-(n-1)}, \ldots, w_{t-1}, \cdot)}$$

Neural Network Language Model

- **Neural Network Language Model**
  - Model the conditional with neural network
  - 1. context에서 id를 얻는다.
  - 2. Extract vector representation of each ID
  - 3. 각각의 C(w)를 tanh안에 넣어서 softmax
  - Bengio,Ducharme,Vincent and Jauvin 2003



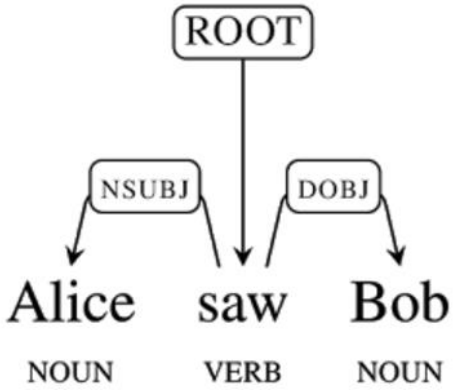
SyntaxNet

- **SyntaxNet 공개**
  - Open source으로 공개된 인공 신경망 프레임워크
  - National Language Understanding(자연어 이해)기능을 TensorFlow를 이용해서 구현
  - English parser가 training이 되어서 나오기 때문에 별도의 training 없이 사용 가능
  - 세계에서 가장 정확도가 높은 모델이라고 자부

SyntaxNet

- **SyntaxNet 원리**
  - 입력 값으로 문장(Sentence)이 주어지면 각각을 어절(part of speech)단위로 구분하여 각 단어의 문법적인 역할을 파악하고 해당 tag를 부여
  - dependency parse tree로 표현

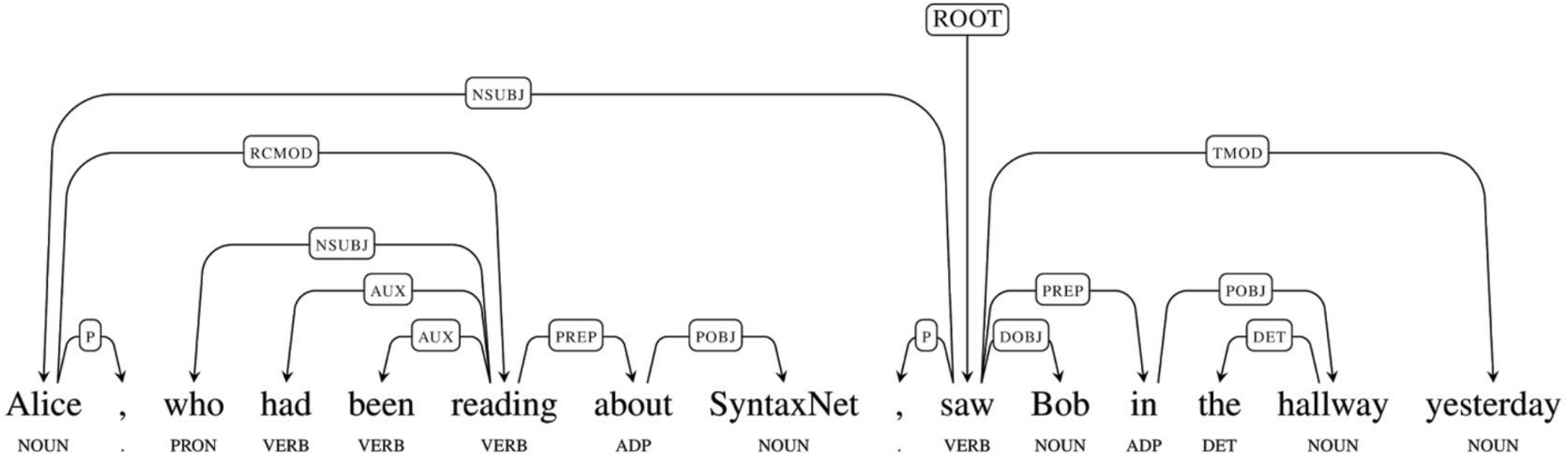
  - 주어진 문장을 구성하고 있는 각 단어들의 문법적인 관계를 결정



- Parsey Mcparseface로 분석한 결과
- 주요 동사인 saw가 문장의 Root에서 최상위 노드
- Alice와 Bob은 명사(noun), saw는 동사(verb)로 분류
- Alice는 주어(subject)이고 Bob은 직접목적어(direct object)이다.

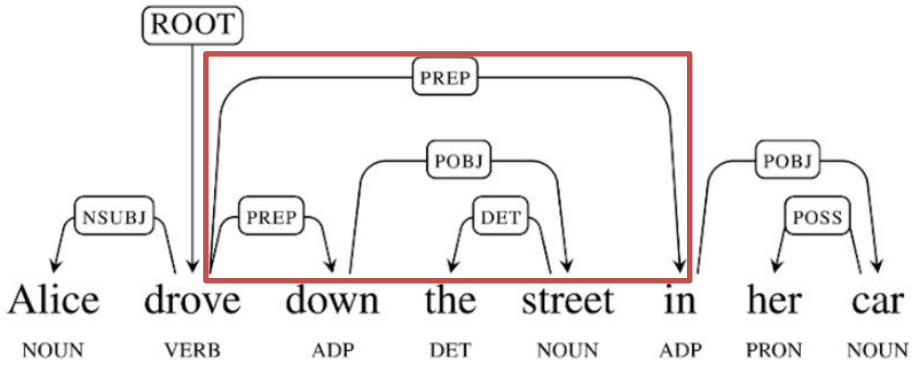
SyntaxNet

- **SyntaxNet 원리**
  - 조금 더 복잡한 문장을 표현
  - Alice는 관계대명사 who와 함께 새로운 동사 reading의 수식을 받고 있고, Saw는 시간을 나타내고 있는 명사 yesterday의 영향을 받고 있다.

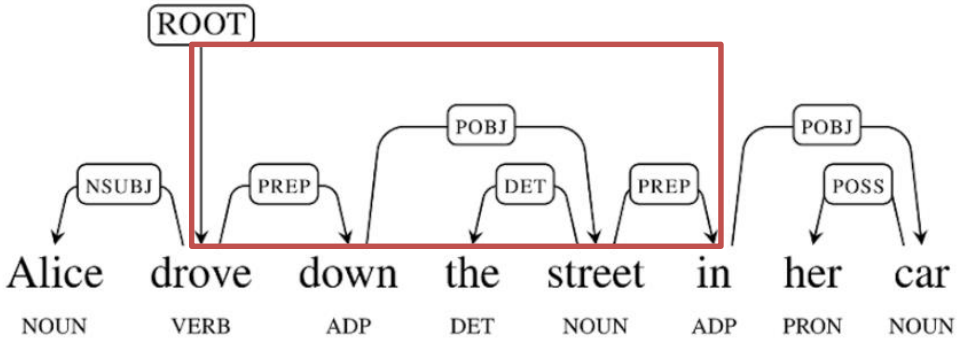


SyntaxNet

- **Why is Parsing So Hard For Computers to Get Right**
  - 인간의 언어는 ambiguity를 가지기 때문에 구문분석이 쉽지 않다.
  - 영어로 대화할 때 보통 20~30 단어를 구사하며, 이를 해석할 수 있는 문법적 경우의 수는 수만 가지
  - Parser가 모든 후보군을 일일이 확인하고, 가장 그럴듯한 것을 찾아내야 한다.
  - Alice drove down the street in her car.



엘리스는 자동차를 타고 거리를 운전했다.

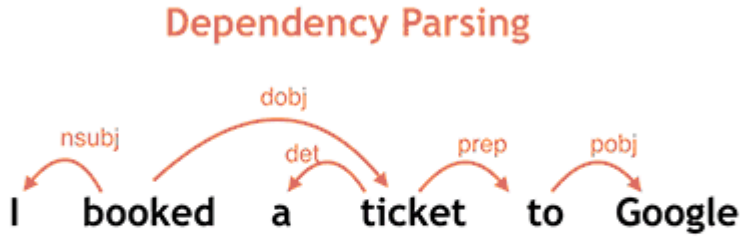


엘리스는 자동차 안에 있는 거리를 운전했다.

- 위의 예제의 경우 전치사 in이 drove를 수식하는지, street를 수식하는지에 따라 해석이 갈린다

SyntaxNet

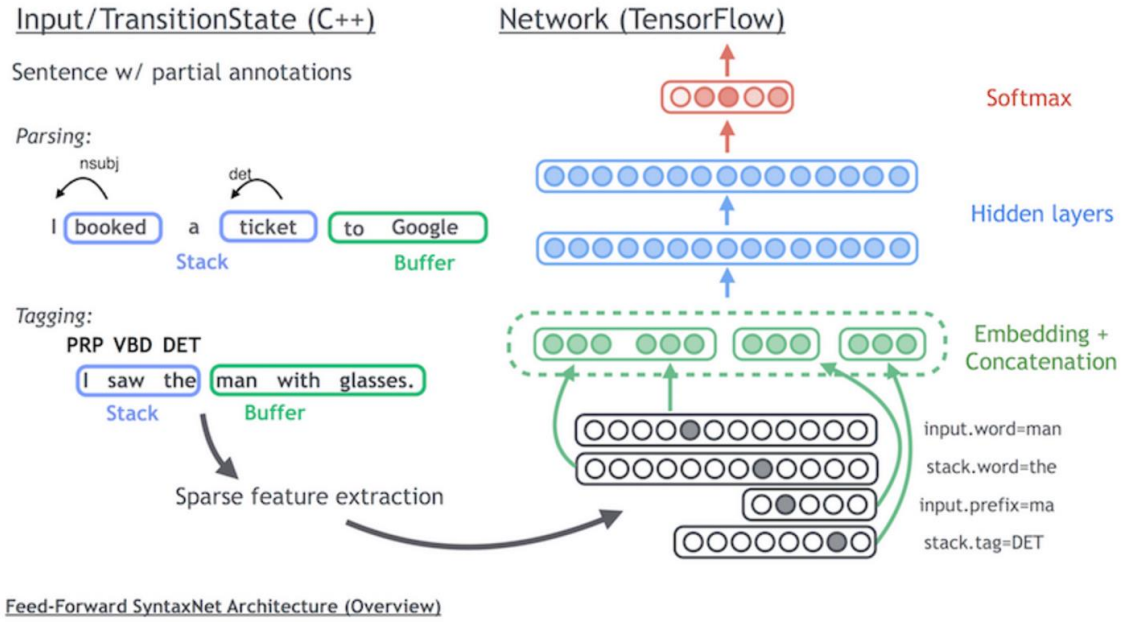
- **SyntaxNet의 문제 해결**
  - 컴퓨터는 인간과는 달리 문법적으로는 오류가 없는 중의적 해석을 처리하기가 쉽지 않다.
  - SyntaxNet은 **인공신경망**을 적용해서 해결
  - 입력된 문장을 각 단어들 사이의 의존성을 고려하여 점진적으로 왼쪽에서부터 오른쪽으로 처리.
  - Neural Network를 사용해서 그럴듯한 방향으로 해석이 가능하도록 결정



SyntaxNet

POS Tagger Training

- POS tag을 하기 위해서 단어 구조의 영향을 고려한다.
  - Ex) the 뒤에는 adjectives or nouns
- POS tag을 위해서 단어에서 features를 뽑아낸 다음, 이것을 인공신경망의 input으로 입력한다.
- 왼쪽에서 오른쪽으로 decision이 진행이 되기 때문에 prior decision도 feature로 입력한다.
  - Ex) the previous predicted tag was a noun.



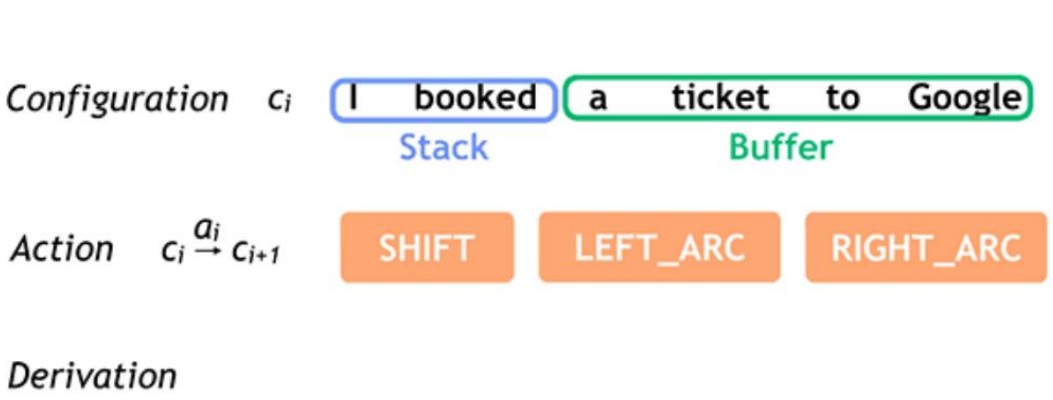
Feed-Forward SyntaxNet Architecture (Overview)

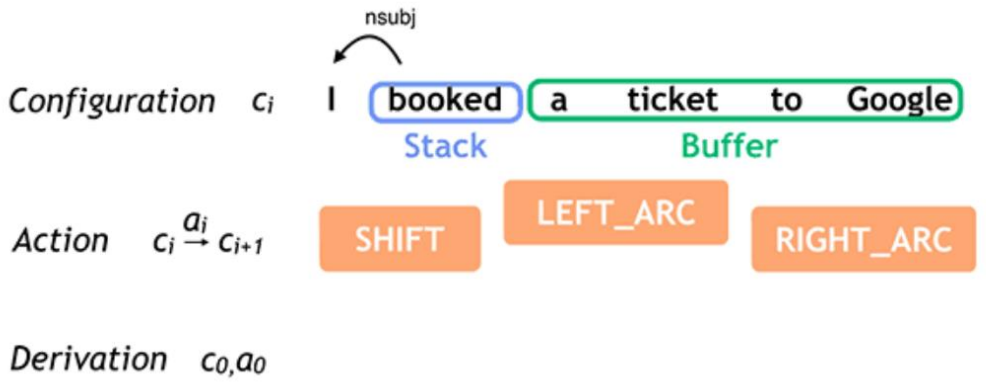
Stack = words we have already tagged
Feature spec = words, suffixes, prefixes

SyntaxNet

Dependency Parsing

- 문장구조에서 서로 어떠한 관계인지 밝혀내야 한다.
- Transition based dependency parser를 이용
- 모두 unprocessed input (buffer)로 시작한 다음 아래 parser가 아래 3가지중 하나를 결정한다.
  - **SHIFT:** Push another word onto the top of the stack, i.e. shifting one token from the buffer to the stack
  - **LEFT_ARC:** Pop the top two words from the stack. Attach the second to the first, creating an arc pointing to the **left**. Push the **first** word back on the stack
  - **RIGHT_ARC:** Pop the top two words from the stack. Attach the second to the first, creating an arc point to the **right**. Push the **second** word back on the stack

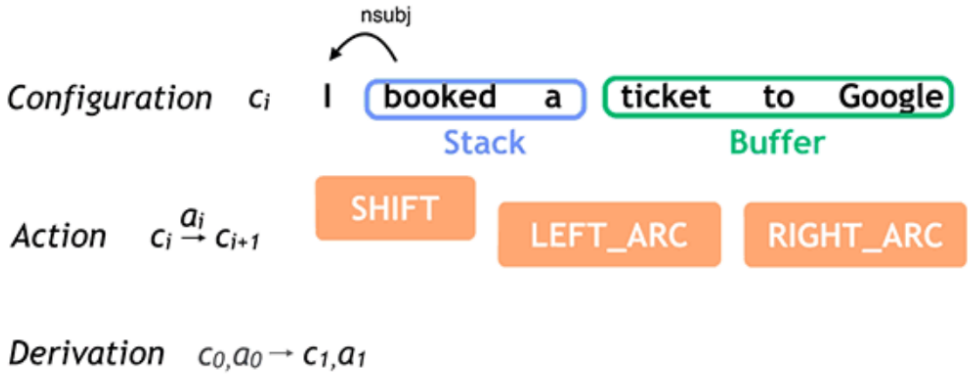


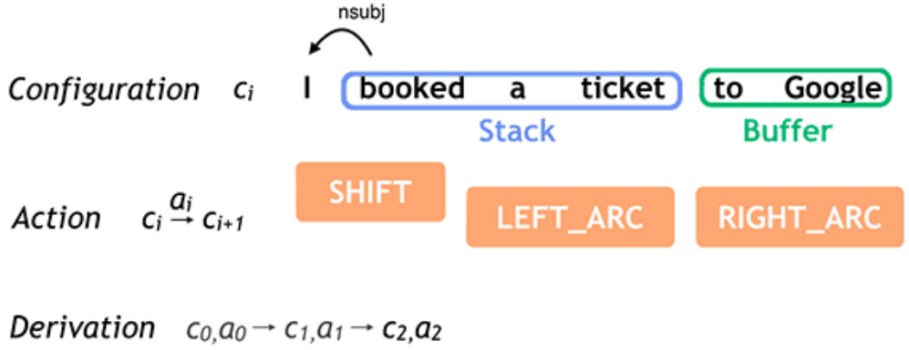


SyntaxNet

Dependency Parsing

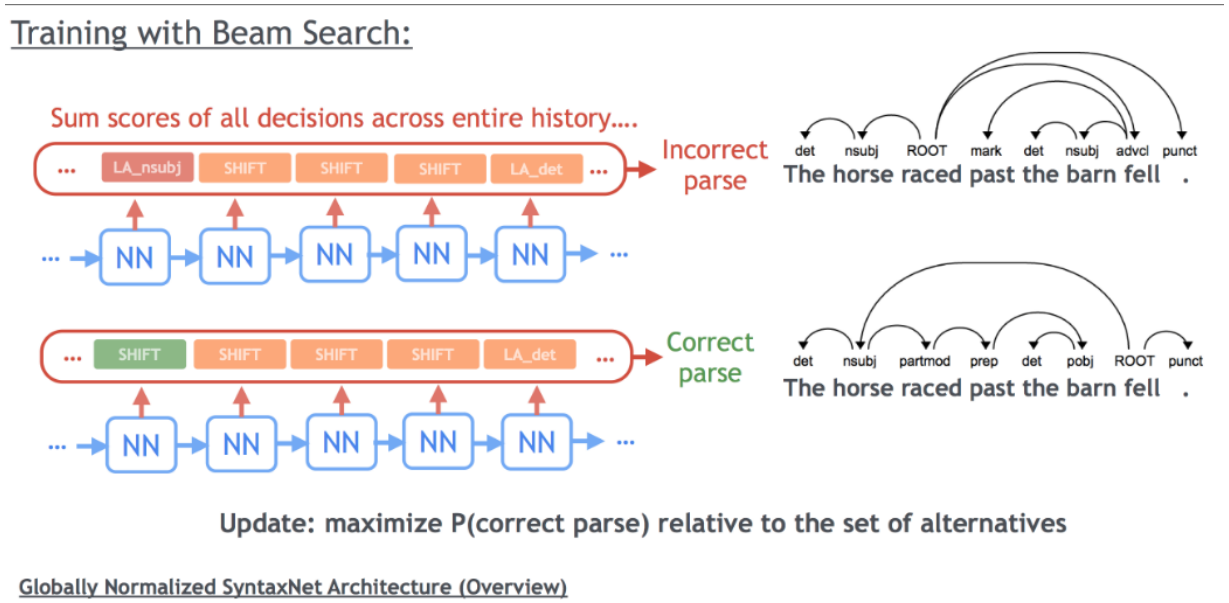
- 문장구조에서 서로 어떠한 관계인지 밝혀내야 한다.
- Transition based dependency parser를 이용
- 모두 unprocessed input (buffer)로 시작한 다음 아래 parser가 아래 3가지중 하나를 결정한다.
  - **SHIFT:** Push another word onto the top of the stack, i.e. shifting one token from the buffer to the stack
  - **LEFT_ARC:** Pop the top two words from the stack. Attach the second to the first, creating an arc pointing to the **left**. Push the **first** word back on the stack
  - **RIGHT_ARC:** Pop the top two words from the stack. Attach the second to the first, creating an arc point to the **right**. Push the **second** word back on the stack





SyntaxNet

- **Training**
  - Global Training
  - Label-bias problem: the model doesn't learn what a good parse like, only what action to take given a history of gold decisions
  - **Beam Search** 사용
    - 다양한 가능성을 유지시키다가 명백한 결격사유가 등장 했을 때 후보 군에서 탈락
    - 8개정도의 후보를 놓고 마지막에 모든 hypothesis를 더해서 최적의 decision 탐색
    - Beam 이 많아질수록 정확하지만, 학습 시간이 느려지는 효과



SyntaxNet

- **SyntaxNet의 성능**
  - 영어 뉴스에서 무작위로 문장을 뽑아내서 이것을 얼마나 잘 처리하는지 판단
  - Parsey MacParseface는 94%이상 파악
  - 기존의 모든 연구성과보다 우수한 성능
  - 언어학에 능통한 사람을 모집하여 실험한 결과 96%

  - 그러나 정돈되지 않은 인터넷 웹에 있는 텍스트에 대한 분석 성능은 떨어짐.
  - 전치사의 접속 모호성을 판단하기 위해 실생활의 지식이 요구되는 경우나 전후 문맥을 파악해야 하는 경우에는 성능이 떨어진다.
  - 영어뿐만 아니라 세계의 모든 언어를 이해 할 수 있도록 진행 할 예정

SyntaxNet

SyntaxNet의 구현

- python 2.7:
  - python 3 support is not available yet
- bazel:
  - **versions 0.2.0 - 0.2.2b, NOT 0.2.3**
  - follow the instructions here
- swig:
  - `apt-get install swig` on Ubuntu
  - `brew install swig` on OSX
- protocol buffers, with a version supported by TensorFlow:
  - check your protobuf version with `pip freeze | grep protobuf`
  - upgrade to a supported version with `pip install -U protobuf==3.0.0b2`
- asciitree, to draw parse trees on the console for the demo:
  - `pip install asciitree`
- numpy, package for scientific computing:
  - `pip install numpy`

SyntaxNet

- **SyntaxNet의 구현**
  - 아래와 같이 training된 모델에서 적용을 할 수도 있고 이것을 수정하여 새롭게 우리가 training 할수도 있다.

```
echo 'Bob brought the pizza to Alice.' | syntaxnet/demo.sh

Input: Bob brought the pizza to Alice .
Parse:
brought VBD ROOT
 +-- Bob NNP nsubj
 +-- pizza NN dobj
 |   +-- the DT det
 +-- to IN prep
 |   +-- Alice NNP pobj
 +-- . . punct
```

SyntaxNet

- **Training**
  - Local Training
  - Gold-decision sequence를 Training data로 이용
  - Softmax layer를 training시켜서 correct action을 예측할 수 있게 설정

```
bazel-bin/syntaxnet/parser_trainer \
  --arg_prefix=brain_parser \
  --batch_size=32 \
  --projectivize_training_set \
  --decay_steps=4400 \
  --graph_builder=greedy \
  --hidden_layer_sizes=200,200 \
  --learning_rate=0.08 \
  --momentum=0.85 \
  --output_path=models \
  --task_context=models/brain_pos/greedy/$PARAMS/context \
  --seed=4 \
  --training_corpus=tagged-training-corpus \
  --tuning_corpus=tagged-tuning-corpus \
  --params=200x200-0.08-4400-0.85-4
```

SyntaxNet

- 실제 예시

```
Input: I study machine learning in SLCF lab .
Parse:
study VBP ROOT
 +-- I PRP nsubj
 +-- learning NN dobj
 |   +-- machine NN nn
 |   +-- in IN prep
 |       +-- lab NN pobj
 |           +-- SLCF NNP nn
 +-- . . punct
```

```
Input: My research focuses on the design and analysis of efficient algorithms for learning , optimizati
on , and inference problems , with a recent emphasis on their applications to data mining and business
analytics .
Parse:
focuses VBZ ROOT
 +-- research NN nsubj
 |   +-- My PRP$ poss
 +-- on IN prep
 |   +-- design NN pobj
 |       +-- the DT det
 |       +-- and CC cc
 |       +-- analysis NN conj
 |       +-- of IN prep
 |           +-- algorithms NNS pobj
 |               +-- efficient JJ amod
 |               +-- for IN prep
 |                   +-- learning NN pobj
 |                       +-- , , punct
 |                       +-- optimization NN conj
 |                       +-- and CC cc
 |                       +-- problems NNS conj
 |                           +-- inference NN nn
 +-- , , punct
 +-- with IN prep
 |   +-- emphasis NN pobj
 |       +-- a DT det
 |       +-- recent JJ amod
 |       +-- on IN prep
 |           +-- applications NNS pobj
 |               +-- their PRP$ poss
 |               +-- to IN prep
 |                   +-- mining NN pobj
 |                       +-- data NNS nn
 |                       +-- and CC cc
 |                       +-- analytics NNS conj
 |                           +-- business NN nn
 +-- . . punct
```

Syntax Net

- **Reference**

- Rastogi, P., Cotterell, R., & Eisner, J. (2016). Weighting finite-state transductions with neural context. In *Proc. of NAACL*.
- Nivre, J. (2006). *Inductive dependency parsing* (pp. 87-120). Springer Netherlands.
- Slav, P. (2016). Announcing SyntaxNet: The world's Most Accurate Parser Goes Open Source , Google Research Blog
- https://github.com/tensorflow/models/tree/master/syntaxnet