Improved Semantic RepresentationsFrom Tree-Structured LongShort-TermMemory Networks

1. Long Short-Term Memory Networks

1.1 Overview

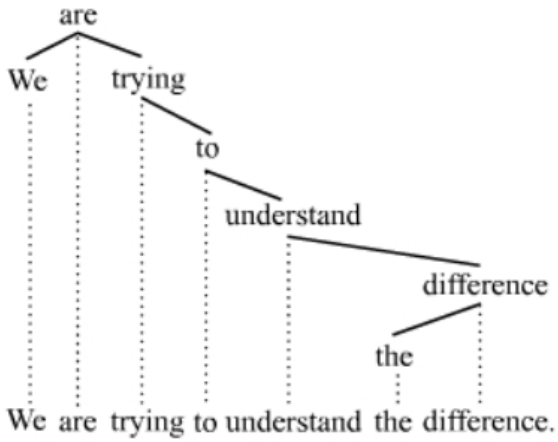
1.1.1 LSTM

- LSTM은 R^d vector들의 collection이다. 여기서 d는 LSTM의 memory dimension이다.
- input gate, forget gate, output gate, memory cell, hidden state가 있다.

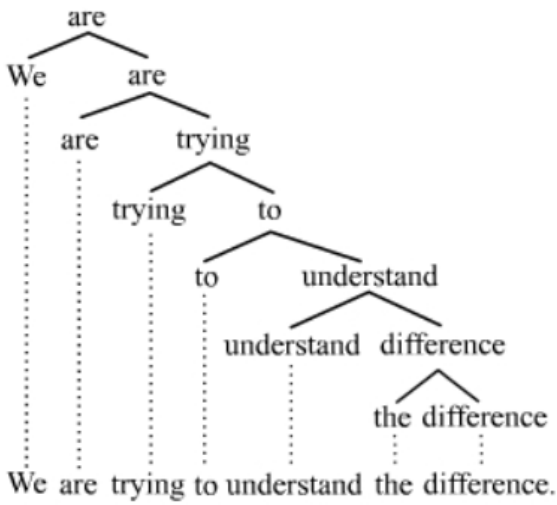
$$
\begin{aligned}
i_t &= \sigma\left(W^{(i)}x_t + U^{(i)}h_{t-1} + b^{(i)}\right), \\
f_t &= \sigma\left(W^{(f)}x_t + U^{(f)}h_{t-1} + b^{(f)}\right), \\
o_t &= \sigma\left(W^{(o)}x_t + U^{(o)}h_{t-1} + b^{(o)}\right), \\
u_t &= \tanh\left(W^{(u)}x_t + U^{(u)}h_{t-1} + b^{(u)}\right), \\
c_t &= i_t \odot u_t + f_t \odot c_{t-1}, \\
h_t &= o_t \odot \tanh(c_t),
\end{aligned}
$$

- input gate, forget gate, output gate는 0에서 1사이의 값을 갖는다.
- forget gate는 이전 memory cell을 얼마나 잊을 지를 결정한다.
- input gate는 각각의 unit을 얼마나 update할 것인지 결정한다.
- output gate는 내부 memory state를 얼마만큼 이용할 건지를 결정한다.

1.1.2 Dependency vs. constituency



Dependency

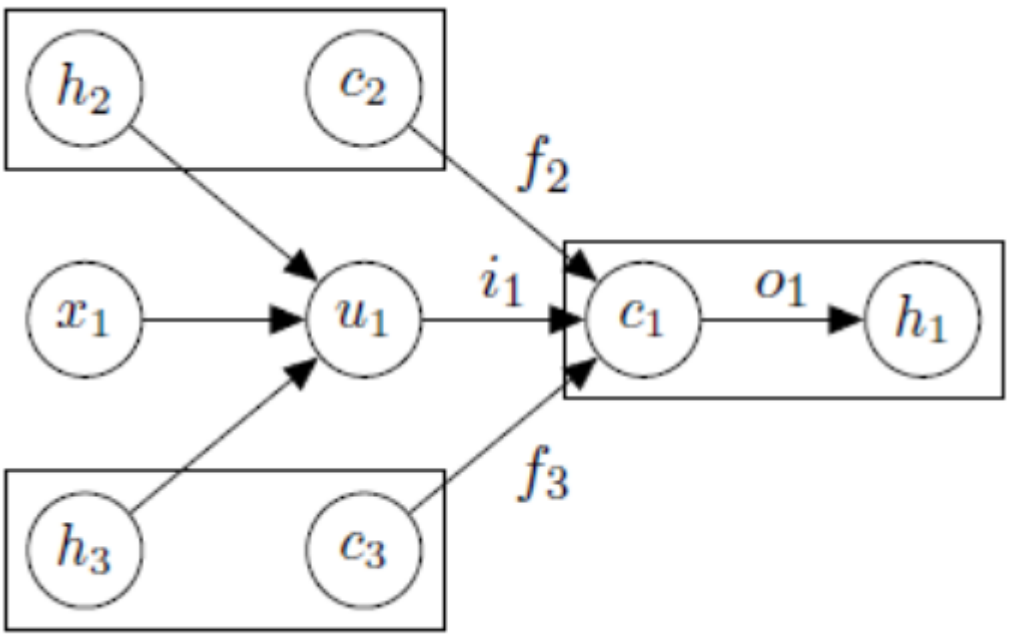


Constituency (BPS)

- dependency grammar는 단어를 중심으로 하고, constituency grammar는 문장 성분을 중심으로 한다. 즉, dependency grammar에서는 특정 단어가 어떤 단어에 의존하는지(수식하는지)에 관심이 있고, constituency grammar에서는 구나 단어가 문장 내에서 어떤 역할(명사구, 동사구 등등)을 하는지에 관심이 있다.

2. Tree-Structured LSTMs

- basic LSTM 구조의 두 가지 응용인 the $\boldsymbol{Child-SumTree-LSTM}$, the $\boldsymbol{N-aryTree-LSTM}$을 소개하겠다.
- 두 구조는 각각의 LSTM unit이 multiple child unit의 정보를 합칠 수 있게 해준다.

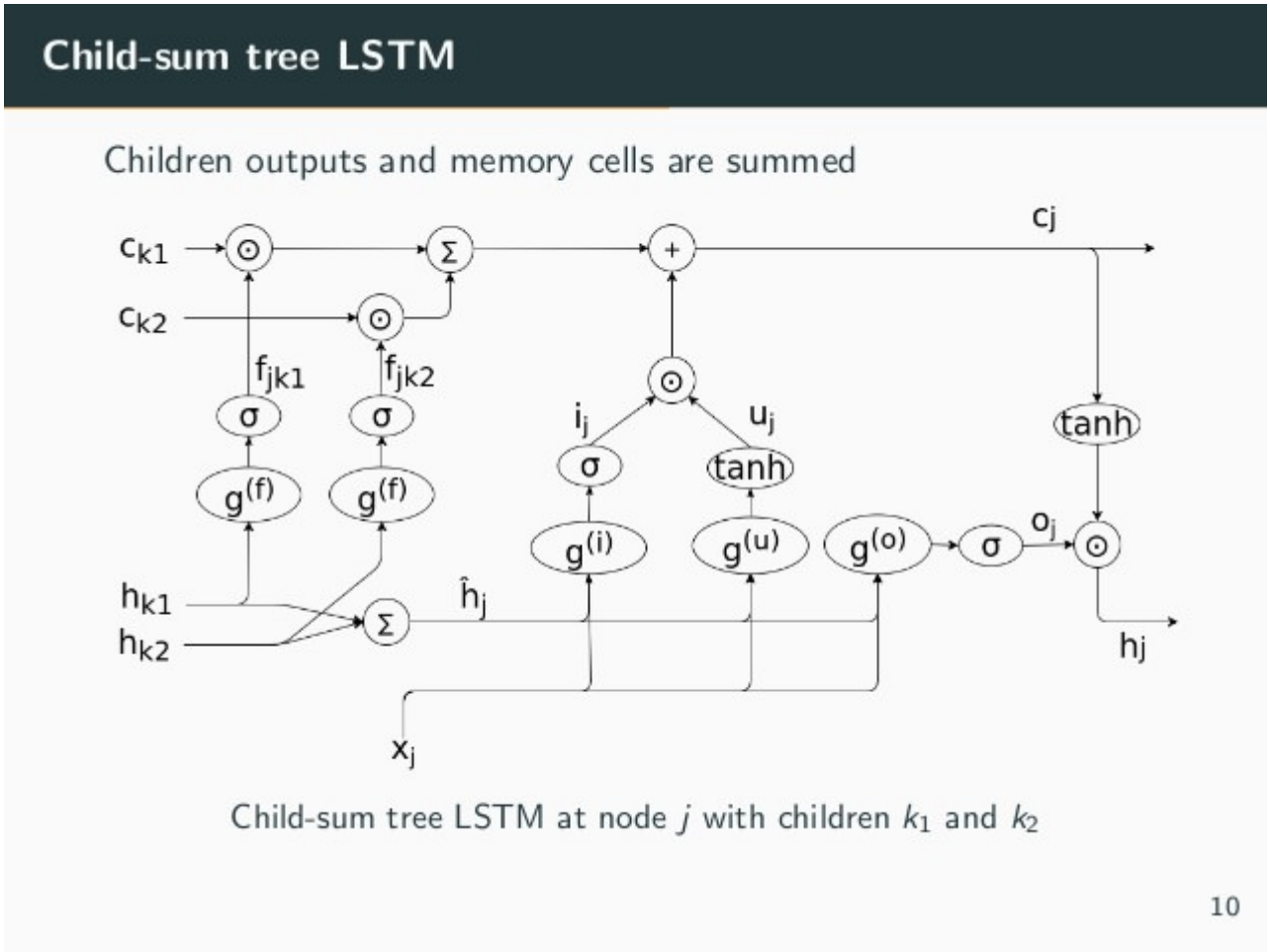


- 일반적인 LSTM unit과 Tree-LSTM이 다른 점이 있다면, gating vector들과 memory cell의 update가 많은 child unit들에 의존한다는 것이다. 그리고 forget gate는 각각의 child k만큼 있게 된다.
- 위와 같은 구조로 Tree-LSTM은 문장에서 중심적인 의미를 가진 단어(semantic head)를 강조하는 것을 배우게 된다. 그리고 감성 분류에서 sentiment를 잘 담고 있는 children의 representation을 보존하는 것도 배울

수 있다.

- 논문에서 각각의 x_j는 문장 내에서 단어의 vector 표현이다.

2.1 Child-Sum Tree-LSTMs



$$\tilde{h}_j = \sum_{k \in C(j)} h_k, \tag{2}$$

$$i_j = \sigma\left(W^{(i)} x_j + U^{(i)} \tilde{h}_j + b^{(i)}\right), \tag{3}$$

$$f_{jk} = \sigma\left(W^{(f)} x_j + U^{(f)} h_k + b^{(f)}\right), \tag{4}$$

$$o_j = \sigma\left(W^{(o)} x_j + U^{(o)} \tilde{h}_j + b^{(o)}\right), \tag{5}$$

$$u_j = \tanh\left(W^{(u)} x_j + U^{(u)} \tilde{h}_j + b^{(u)}\right), \tag{6}$$

$$c_j = i_j \odot u_j + \sum_{k \in C(j)} f_{jk} \odot c_k, \tag{7}$$

$$h_j = o_j \odot \tanh(c_j), \tag{8}$$

*$C(j)$는 j번째 node의 children을 가리킨다.

- 각각의 parameter matrix(W)를 Tree-LSTM unit의 component vector들과 input, unit의 children의 hidden states 사이의 상관 관계를 encoding한 거로 이해하면 된다. 예를 들면 동사와 같이 의미적으로 중요한 word가 input으로 주어지면 $W^{(i)}$의 값이 1에 가깝게 된다.
- Child-Sum Tree-LSTM의 unit은 child hidden states의 합을 참고해 components의 값을 결정하기 때문에 각 node의 children의 수(branching factor)가 많거나 children에 문장의 순서가 반영되지 않았을 때 적절하다.

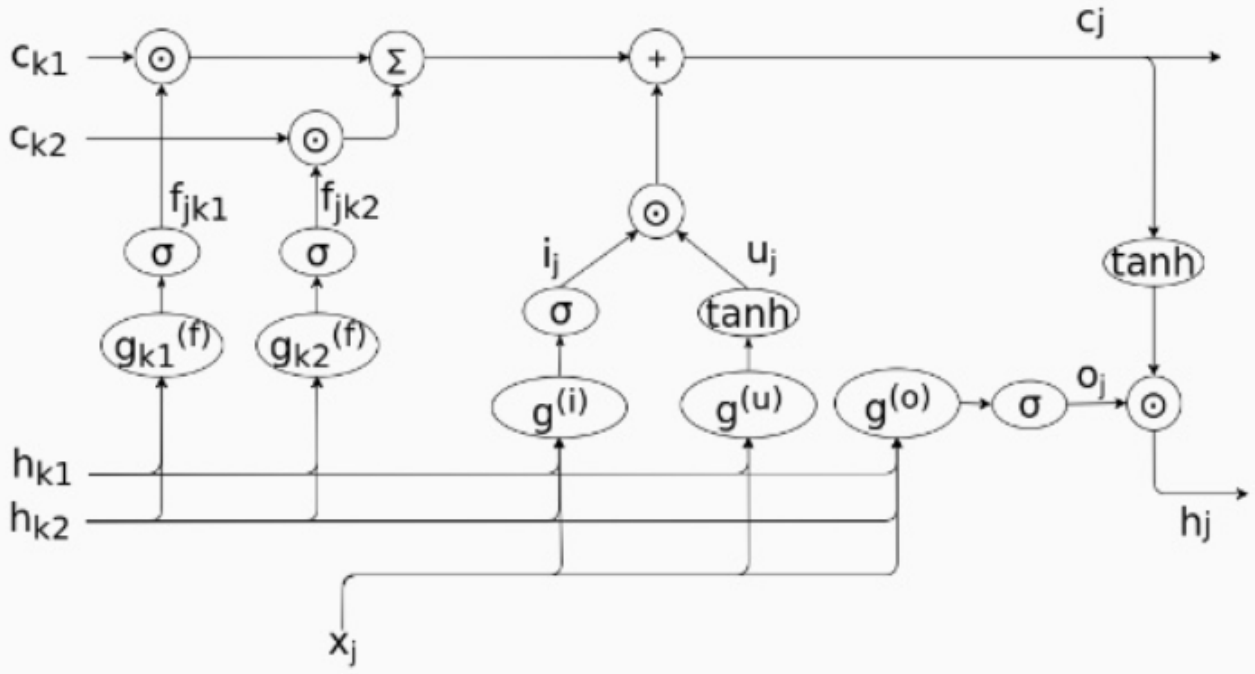
*head란 구의 문법적 범주를 결정하는 단어를 말한다. 예를 들어 big red dog라는 구가 있을 때 명사 구란 것을 결정짓는 것은 dog이기 때문에 dog가 head이고 나머지 big과 red는 dog를 수식하기 때문에 dog의 dependent이다.

- dependency tree에 적용된 Child-Sum Tree-LSTM를 *Dependency Tree-LSTM*라고 칭하겠다.

2.2 N-ary Tree-LSTMs

- branching factor가 최대 N개이고 children이 순서대로 정렬되어 있을 때 사용된다.

Given $g_k^{(n)}(x_t, h_{l_1}, \cdots, h_{l_N}) = W^{(n)}x_t + \sum_{l=1}^{N} U_{kl}^{(n)} h_{jl} + b^{(n)}$



Binary tree LSTM at node j with children k_1 and k_2

$$i_j = \sigma\left(W^{(i)}x_j + \sum_{\ell=1}^{N} U_{\ell}^{(i)} h_{j\ell} + b^{(i)}\right), \quad (9)$$

$$f_{jk} = \sigma\left(W^{(f)}x_j + \sum_{\ell=1}^{N} U_{k\ell}^{(f)} h_{j\ell} + b^{(f)}\right), \quad (10)$$

$$o_j = \sigma\left(W^{(o)}x_j + \sum_{\ell=1}^{N} U_{\ell}^{(o)} h_{j\ell} + b^{(o)}\right), \quad (11)$$

$$u_j = \tanh\left(W^{(u)}x_j + \sum_{\ell=1}^{N} U_{\ell}^{(u)} h_{j\ell} + b^{(u)}\right), \quad (12)$$

$$c_j = i_j \odot u_j + \sum_{\ell=1}^{N} f_{j\ell} \odot c_{j\ell}, \quad (13)$$

$$h_j = o_j \odot \tanh(c_j), \quad (14)$$

- Child-Sum Tree-LSTM과 다른 점은 딱 형광펜으로 강조한 부분 밖에 없다. child의 순서를 고려하는게 유일한 차이인 것 같다.
- 각각의 child에 대해 별개의 parameter matrix를 설정해주는 것은 model이 unit의 children의 state에 대해 더 잘 값을 설정할 수 있게 해준다. 예를 들면 왼쪽 child가 명사구이고 오른쪽 child가 동사구라고 하면, 동사구를 강조해 주는 게 더 좋다. 따라서 명사구에 해당하는 forget gate의 components를 0에 가깝게 설정되도록 하고 동사구에 해당하는 forget gate는 반대로 설정되도록 학습시켜주면 된다.
- child를 왼쪽과 오른쪽으로 구분할 수 있기 때문에, 일반적으로 *Binary Tree-LSTM* unit을 *binarized constituency trees*에 적용한다.
- *Binary Tree-LSTM*의 적용을 *Constituency Tree-LSTM*이라고 하겠다.
- 적용에서의 주요한 차이는 compositional parameter이다. Dependency Tree-LSTM에서는 **dependent와 head**, Constituency Tree-LSTM에서는 **left child와 right child**가 되겠다.

3. Models

3.1 Tree-LSTM Classification

- 별개의 class인 y를 예측하는 작업을 한다.
- 각각의 node j에서 $\{x\}_j$에 해당하는 label $\hat{y}_j$를 맞추기 위해 softmax classifier를 사용할 것이다.
- classifier는 해당 node의 hidden state h_j를 input으로 받을 것이다.

$$\hat{p}_\theta(y \mid \{x\}_j) = \text{softmax}\left(W^{(s)} h_j + b^{(s)}\right)$$

$$\hat{y}_j = \arg\max_y \hat{p}_\theta\left(y \mid \{x\}_j\right).$$

- cost function은 true class label $y^{(k)}$의 negative log-likelihood이다.

$$J(\theta) = -\frac{1}{m} \sum_{k=1}^{m} \log \hat{p}_\theta\left(y^{(k)} \mid \{x\}^{(k)}\right) + \frac{\lambda}{2}\|\theta\|_2^2,$$

m: training set에서 labeling된 node의 개수

k: k번째 labeled node

λ : L2 regularization의 hyperparameter

- 해당 논문에서는 sentiment classification을 했다.

3.2 Semantic Relatedness of Sentence Pairs

- [1, K]의 범주를 가지는 sentence pair의 유사도를 구한다. K는 정수이다.
- 높은 점수는 높은 유사도를 가지는 걸 나타낸다.
- 일단 Tree-LSTM model을 사용해서 h_L, h_R sentence representation을 만든다.
- (h_L, h_R)의 사이의 거리와 각을 이용해 유사도를 예측한다.

$$h_\times = h_L \odot h_R,$$

$$h_+ = |h_L - h_R|,$$

$$h_s = \sigma\left(W^{(\times)} h_\times + W^{(+)} h_+ + b^{(h)}\right),$$

$$\hat{p}_\theta = \text{softmax}\left(W^{(p)} h_s + b^{(p)}\right),$$

$$\hat{y} = r^T \hat{p}_\theta,$$

*r^T = [1 2 ... K]

- h_X는 input representation의 elementwise comparison이다.
- 예측 분포인 $\hat{p}_\theta$와 r^T 내적한 값인 $\hat{y}$가 y와 비슷하게 나오도록 해야했다.
- 그래서 $y = r^T p$를 만족하는 sparse한 target distribution인 p를 다음과 같이 정의했다.

$$p_i = \begin{cases} y - \lfloor y \rfloor, & i = \lfloor y \rfloor + 1 \\ \lfloor y \rfloor - y + 1, & i = \lfloor y \rfloor \\ 0 & \text{otherwise} \end{cases}$$

- r^T가 점수 class이므로 p는 해당 점수가 나올 확률이 된다. 그러므로 r^T와 p를 곱했을 때, 해당 class의 값이 가장 크게 나와야 한다.
- 예를 들어 유사도가 10.7이라고 해보자. 그렇다면 p_{11}이 가장 큰 값을 가져야 한다. 이때 $p_{11} = 0.7, p_{10} = 0.3$이 된다. 따라서 10.7과 가장 가까운 정수값인 11이 내적에 큰 영향을 주게되는 것이다.
- cost 함수는 p와 $\hat{p}_\theta$사이의 정규화된 KL-divergence이다.

$$J(\theta) = \frac{1}{m} \sum_{k=1}^{m} \mathrm{KL}\left(p^{(k)} \,\middle\|\, \hat{p}_\theta^{(k)} \right) + \frac{\lambda}{2} \|\theta\|_2^2,$$

m: training pair의 개수

k: k번째 sentence pair

4. Results

4.1 Sentiment Classification

Method	Fine-grained	Binary
RAE (Socher et al., 2013)	43.2	82.4
MV-RNN (Socher et al., 2013)	44.4	82.9
RNTN (Socher et al., 2013)	45.7	85.4
DCNN (Blunsom et al., 2014)	48.5	86.8
Paragraph-Vec (Le and Mikolov, 2014)	48.7	87.8
CNN-non-static (Kim, 2014)	48.0	87.2
CNN-multichannel (Kim, 2014)	47.4	**88.1**
DRNN (Irsoy and Cardie, 2014)	49.8	86.6
LSTM	46.4 (1.1)	84.9 (0.6)
Bidirectional LSTM	49.1 (1.0)	87.5 (0.5)
2-layer LSTM	46.0 (1.3)	86.3 (0.6)
2-layer Bidirectional LSTM	48.5 (1.0)	87.2 (1.0)
Dependency Tree-LSTM	48.4 (0.4)	85.7 (0.4)
Constituency Tree-LSTM		
– randomly initialized vectors	43.9 (0.6)	82.0 (0.5)
– Glove vectors, fixed	49.7 (0.4)	87.5 (0.8)
– Glove vectors, tuned	**51.0** (0.5)	88.0 (0.3)

*Fine-grained: 5-class sentiment classification.

*Binary: positive/negative sentiment classification.

- constituency Tree-LSTM이 가장 성적이 좋았다.
- training 동안 word representation을 update하는 것(tuned)은 fine-grained에서는 좋은 결과를 내는 데 많은 기여를 했지만, binary classification에서는 미미한 영향을 끼쳤다.

4.2 Semantic Relatedness

Method	Pearson's r	Spearman's ρ	MSE
Illinois-LH (Lai and Hockenmaier, 2014)	0.7993	0.7538	0.3692
UNAL-NLP (Jimenez et al., 2014)	0.8070	0.7489	0.3550
Meaning Factory (Bjerva et al., 2014)	0.8268	0.7721	0.3224
ECNU (Zhao et al., 2014)	0.8414	–	–
Mean vectors	0.7577 (0.0013)	0.6738 (0.0027)	0.4557 (0.0090)
DT-RNN (Socher et al., 2014)	0.7923 (0.0070)	0.7319 (0.0071)	0.3822 (0.0137)
SDT-RNN (Socher et al., 2014)	0.7900 (0.0042)	0.7304 (0.0076)	0.3848 (0.0074)
LSTM	0.8528 (0.0031)	0.7911 (0.0059)	0.2831 (0.0092)
Bidirectional LSTM	0.8567 (0.0028)	0.7966 (0.0053)	0.2736 (0.0063)
2-layer LSTM	0.8515 (0.0066)	0.7896 (0.0088)	0.2838 (0.0150)
2-layer Bidirectional LSTM	0.8558 (0.0014)	0.7965 (0.0018)	0.2762 (0.0020)
Constituency Tree-LSTM	0.8582 (0.0038)	0.7966 (0.0053)	0.2734 (0.0108)
Dependency Tree-LSTM	**0.8676** (0.0030)	**0.8083** (0.0042)	**0.2532** (0.0052)

*pearson's r, spearmam's p, mse는 evaluation matrics

*처음 두 metrics는 사람의 평가와 비교했을 때 상관 관계

- Tree-LSTM 모델이 어떤 feature engineering이 없이 다른 system보다 성능이 좋았다.
- 제일 좋은 결과는 dependency Tree-LSTM을 사용했을 때였다.
- sentiment classification task와 달리 true label값(supervision)을 tree의 root에서만 받았다.

Reference

- Introduction to Tree-LSTMs (해당 논문에 대한 presentation)
  https://www.slideshare.net/tuvistavie/tree-lstm
- https://en.wikipedia.org/wiki/Head_(linguistics)
- http://taweb.aichi-u.ac.jp/tmgross/DG.html