

Lecture 16. Coreference Resolution

I. What is Coreference Resolution?

Coreference란 텍스트 안에서 real word에 존재하는 entity를 모두 찾아내는 것을 의미한다. 여기서 entity라 하는 고유명사가 될 수도 있겠지만 고유 명사를 가리키는 대명사나 혹은 일반 명사도 다 entity가 될 수 있기 때문에 어렵다. 그리고 어떤 단어가 entity인지 아닌지가 항상 분명한 것은 아니기 때문에 어렵다. Coreference를 풀 때는 우선 고유 명사 찾아내기, 이 고유 명사를 가리키는 일반 명사나 대명사를 찾아야 하는데 실제로는 전자가 후자보다 더 어렵다.

- Mathematically, we want to maximize this probability:

$$\sum_{j=1}^{i-1} \mathbb{1}(y_{ij} = 1) p(m_j, m_i)$$

- Turning this into a loss function:

$$J = \sum_{i=2}^N -\log \left(\sum_{j=1}^{i-1} \mathbb{1}(y_{ij} = 1) p(m_j, m_i) \right)$$

Iterate over all the mentions
in the document

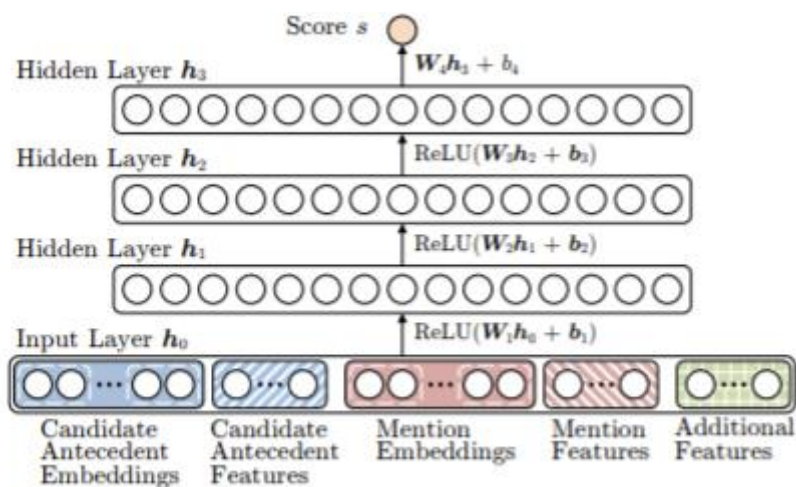
Usual trick of taking negative
log to go from likelihood to loss

II. probability 계산 방법

1. Non-neural statistical classifier(Non-Neural Coref Model : Features)

- Person/Number/Gender agreement
- Semantic compatibility
- Certain syntactic constraints
- More recently mentioned entities preferred for referenced
- Grammatical Role : Prefer entities in the subject position
- Parallelism

2. Simple neural network(Neural Coref Model)

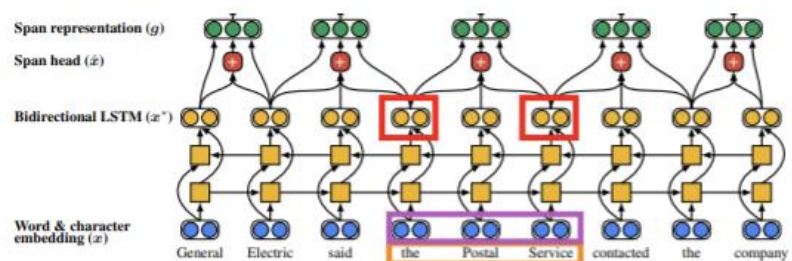


-Standard feed-forward neural network : Input layer(word embeddings and a few categorical features)

-Embeddings : 이전의 두개의 단어들, first word, last word, head word,...,of each mention

-Distance, Document genre, Speaker information 등

3. More advanced model using LSTMs, attention(End-to-end Model)



Span representation: $g_i = [\mathbf{x}_{\text{START}(i)}^*, \mathbf{x}_{\text{END}(i)}^*, \hat{\mathbf{x}}_i, \phi(i)]$

BILSTM hidden states for span's start and end

Attention-based representation (details next slide) of the words in the span

Additional features

Attention scores

$$\alpha_t = \mathbf{w}_\alpha \cdot \text{FFNN}_\alpha(\mathbf{x}_t^*)$$

dot product of weight vector and transformed hidden state

Attention distribution

$$a_{i,t} = \frac{\exp(\alpha_t)}{\sum_{k=\text{START}(i)}^{\text{END}(i)} \exp(\alpha_k)}$$

just a softmax over attention scores for the span

Final representation

$$\hat{\mathbf{x}}_i = \sum_{t=\text{START}(i)}^{\text{END}(i)} a_{i,t} \cdot \mathbf{x}_t$$

Attention-weighted sum of word embeddings

- Lastly, score every pair of spans to decide if they are coreferent mentions

$$s(i, j) = s_m(i) + s_m(j) + s_a(i, j)$$

Are spans i and j coreferent mentions? Is i a mention? Is j a mention? Do they look coreferent?

- Scoring functions take the span representations as input

$$s_m(i) = \mathbf{w}_m \cdot \text{FFNN}_m(\mathbf{g}_i)$$

$$s_a(i, j) = \mathbf{w}_a \cdot \text{FFNN}_a([\mathbf{g}_i, \mathbf{g}_j, \mathbf{g}_i \circ \mathbf{g}_j, \phi(i, j)])$$

include multiplicative interactions between the representations again, we have some extra features

-Current state-of-the-art model for coreference resolution

-Mention ranking model

-Improvements over simple feed-forward NN : LSTM 사용, attention 사용, mention detection과 coreference end-to-end를 한다.

1) word embedding matrix와 character-level CNN을 사용하여 문서 안의 단어들을 임베딩 한다.

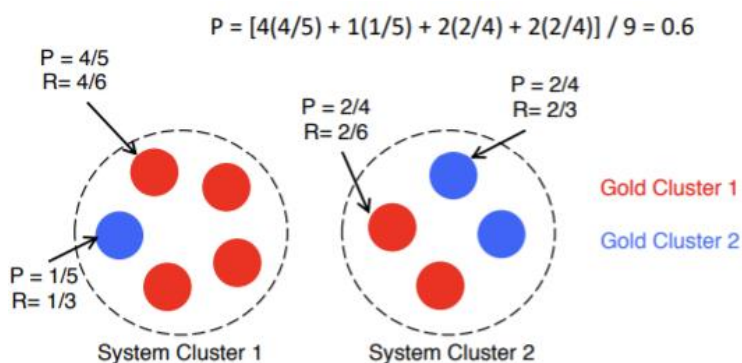
2) bidirectional LSTM을 한다.

3) vector로써 START(j)에서 END(j)까지 움직이는 text i 의 각각의 span을 표현한다.

4) \hat{x}_i 는 span안의 word embeddings의 attention-weighted average이다.


III. Coreference Evaluation

평가 지표는 precision과 recall이다.



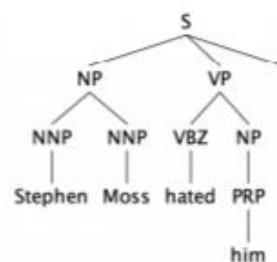
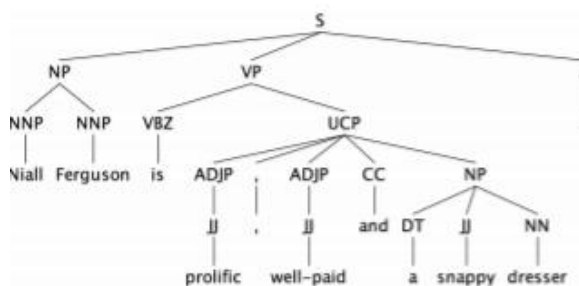
대용(anaphora) 관계에 있는 것이 같은 것을 가리키는 것 (corefer)은 아니기 때문에 둘은 다른 것이다.

IV. Coreference 전통적 방법 – Hobb's naïve algorithm

1. Begin at the NP immediately dominating the pronoun 
2. Go up tree to first NP or S. Call this X, and the path p.
3. Traverse all branches below X to the left of p, left-to-right, breadth-first. Propose as antecedent any NP that has a NP or S between it and X
4. If X is the highest S in the sentence, traverse the parse trees of the previous sentences in the order of recency. Traverse each tree left-to-right, breadth first. When an NP is encountered, propose as antecedent. If X not the highest node, go to step 5.
5. From node X, go up the tree to the first NP or S. Call it X, and the path p.
6. If X is an NP and the path p to X came from a non-head phrase of X (a specifier or adjunct, such as a possessive, PP, apposition, or relative clause), propose X as antecedent
(The original said "did not pass through the N' that X immediately dominates", but the Penn Treebank grammar lacks N' nodes....)
7. Traverse all branches below X to the left of the path, in a left-to-right, breadth first manner. Propose any NP encountered as the antecedent
8. If X is an S node, traverse all branches of X to the right of the path but do not go below any NP or S encountered. Propose any NP as the antecedent.
9. Go to step 4

4

아래 이미지에서 him을 찾는다고 하면 우선 가장 가까운 NP부터 찾되 이 NP와 현재 him 사이에는 다른 NP 혹은 S가 있어야 한다. 그 이유는 가장 가까운 NP와 NP는 주어와 목적어 관계에 있을 가능성이 높고 그렇다면 재귀 형태가 와야 하기 때문에 그 전 문장으로 recency에 따라 가서 알고리즘 대로 풀어간다.



하지만 이렇게 문장의 의미를 고려하지 않고 오직 syntactic features만을 이용해서 coreference를 접근하는 것은 한계가 있을 수 있다. 왜냐하면 같은 문장 구조를 지닌 문장들이라고 해도 문장의 의미에 따라 coreference 관계는 전혀 달라질 수 있기 때문이다. 그래서 이 한계를 극복하기 위해서 neural networks를 사용하는데 word vector를 사용하기 때문에 단어들의 semantic representation을 활용할 수 있다는 장점이 있다.