Forest-based Neural Machine Translation

Chunpeng Ma, Akihiro Tamura,

Masao Utiyama, Tiejun Zhao, Eiichiro Sumita

哈爾濱工業大學
HARBIN INSTITUTE OF TECHNOLOGY

愛媛大学
EHIME UNIVERSITY

国立研究開発法人
情報通信研究機構
National Institute of Information and Communications Technology

Motivation

Key point: Syntactic Information

- To use or not to use?
- How to use?
- To what extent?

Key point: Syntactic Information

- To use or not to use?
  - string-to-string model — × **Use implicitly**
  - tree/graph-to-string model — ◯ **Use explicitly**
- How to use?
- To what extent?

Key point: Syntactic Information

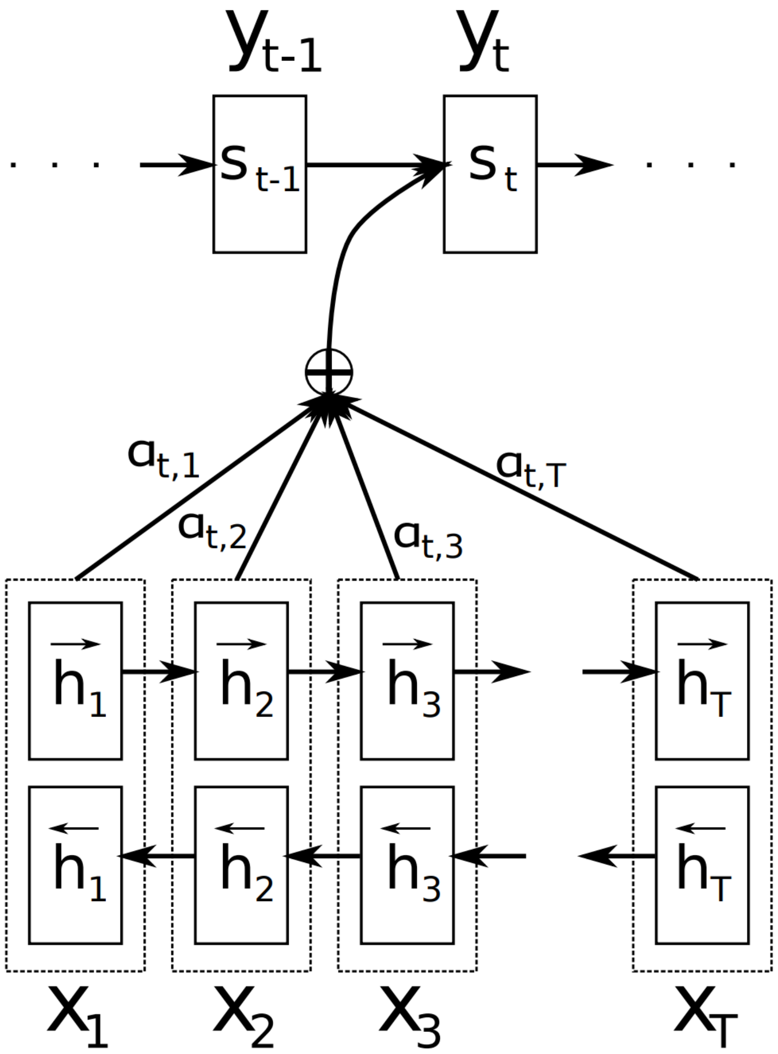
- To use or not to use?
  - string-to-string model — × **Use implicitly**
  - tree/graph-to-string model — ◯ **Use explicitly**
- How to use?
  - Change network structure — × **Complicated**
  - Change model input — ◯ **Simple**
- To what extent?

Key point: Syntactic Information

- To use or not to use?
  - string-to-string model — × Use implicitly
  - tree/graph-to-string model — ○ Use explicitly
- How to use?
  - Change network structure — × Complicated
  - Change model input — ○ Simple
- To what extent?
  - One parsing tree — × Less information
  - Multiple parsing trees — ○ More information

Background

Sequence-to-sequence Model with Attention Mechanism

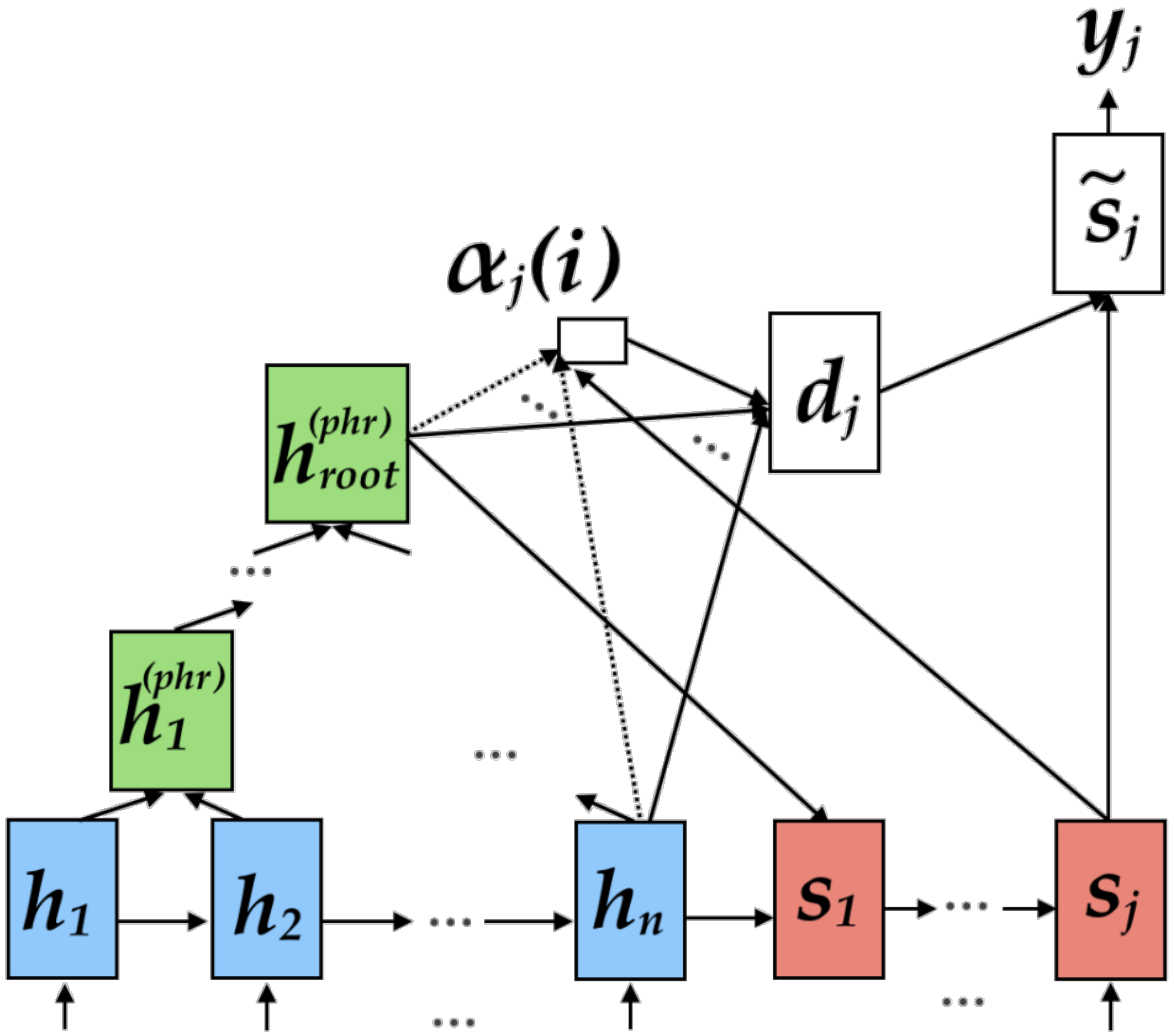


$$c_i = \sum_{j=0}^{T} \alpha_{ij} h_j,$$

$$\alpha_{ij} = \frac{\exp(a(s_{i-1}, h_j))}{\sum_{k=0}^{T} \exp(a(s_{i-1}, h_k))}$$

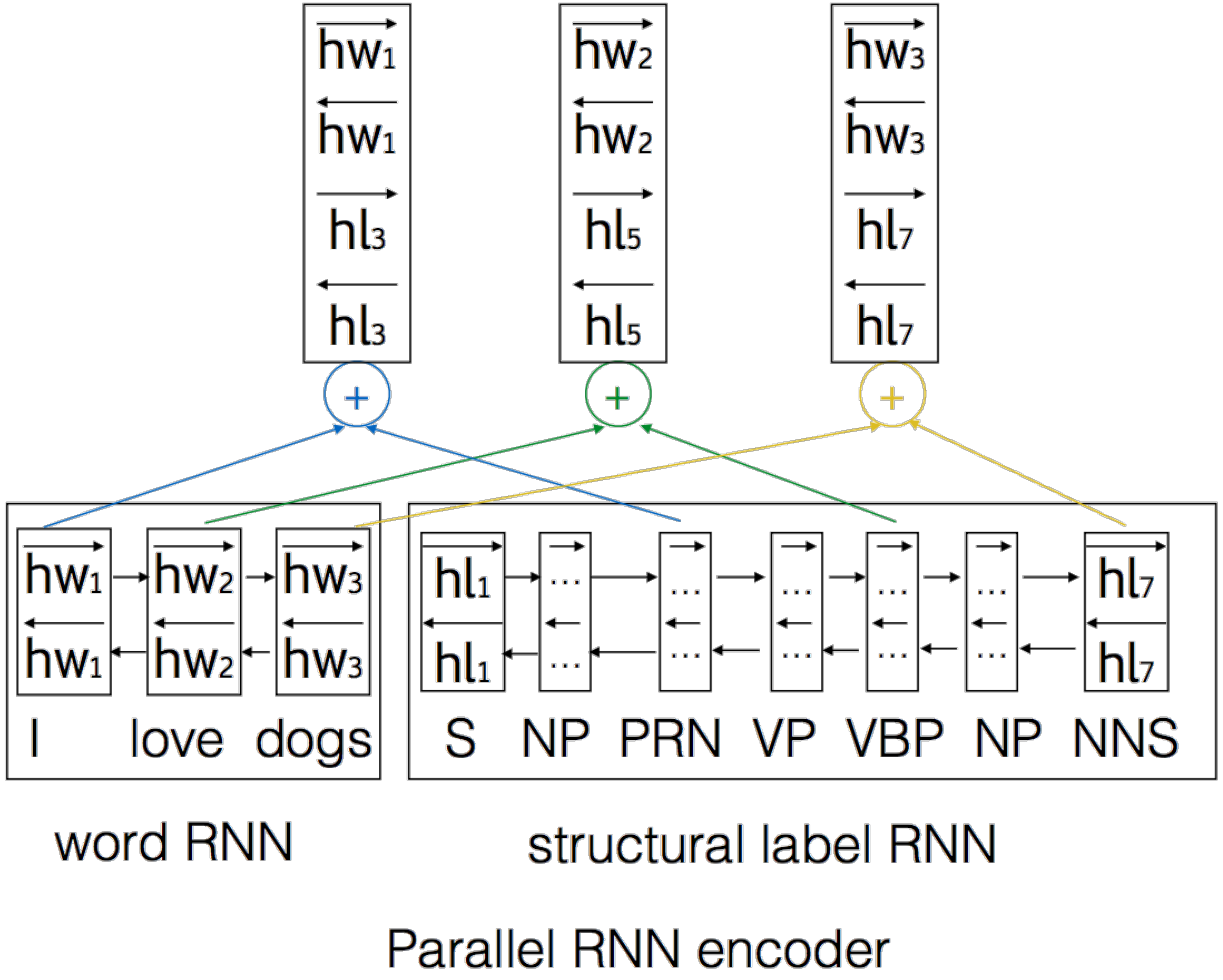
Tree-based NMT: Change network structure

- Eriguchi et al. (2016)



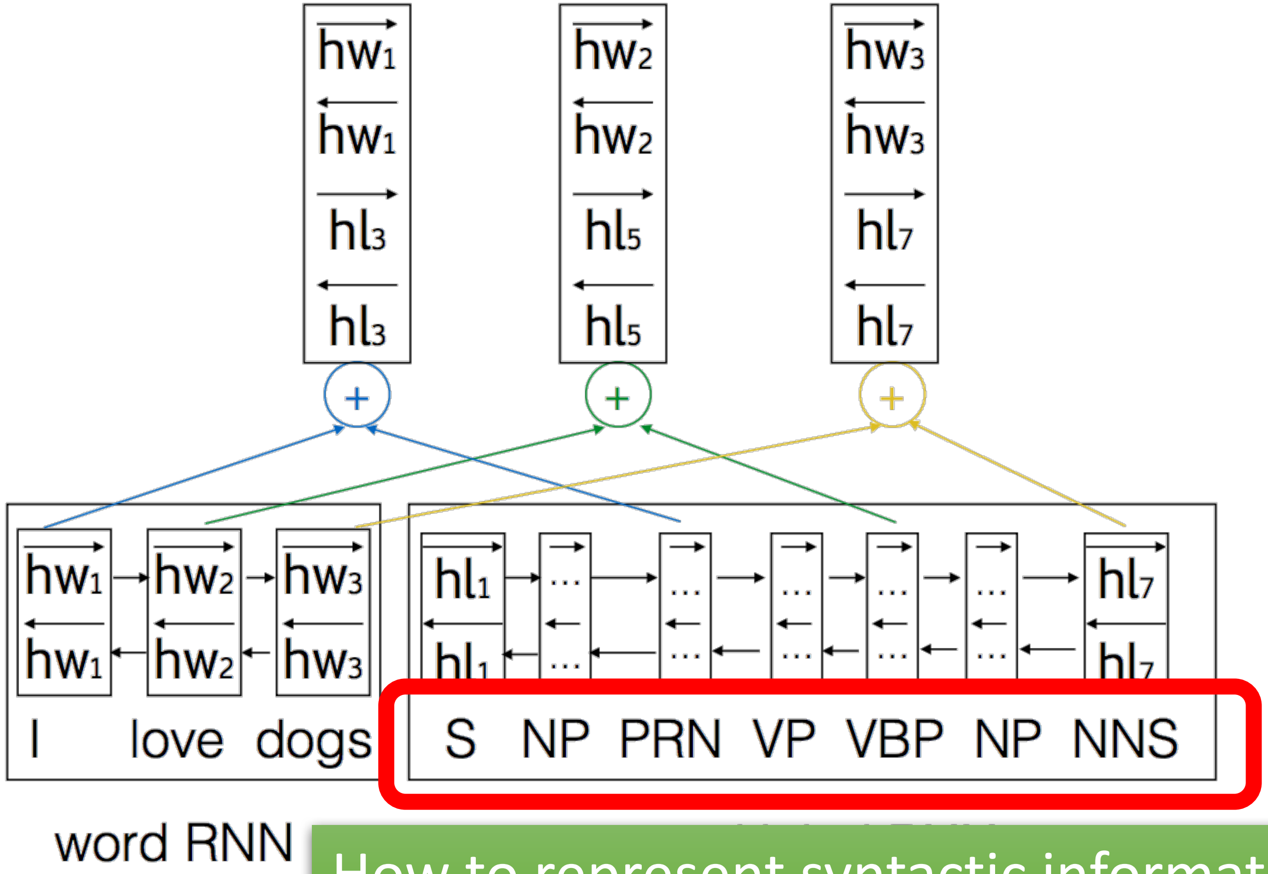
Tree-based NMT: Change model input

- Li et al. (2017)



Tree-based NMT: Change model input

- Li et al. (2017)

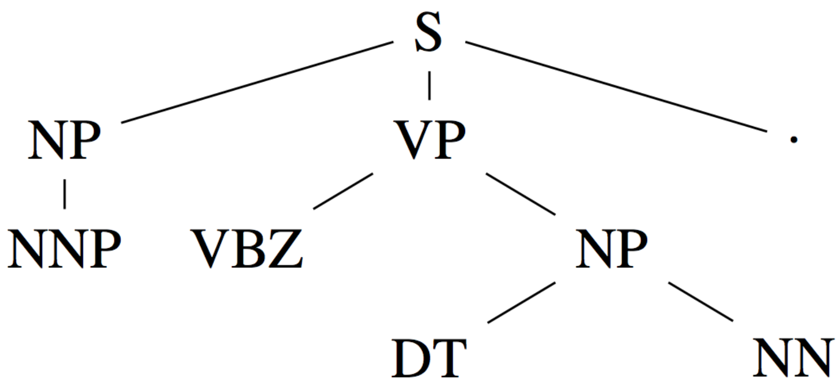


How to represent syntactic information appropriately using a sequence?

Tree Linearization

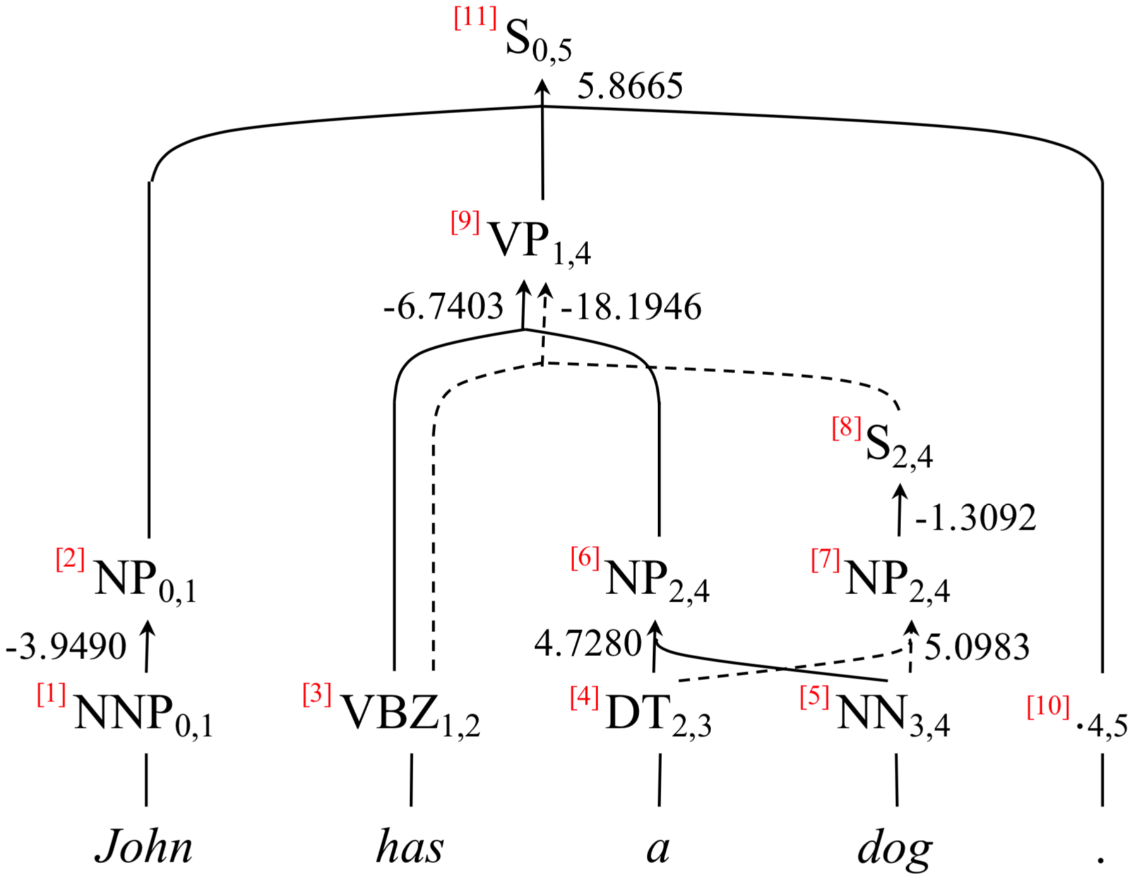
- Vinyals et al. (2015)

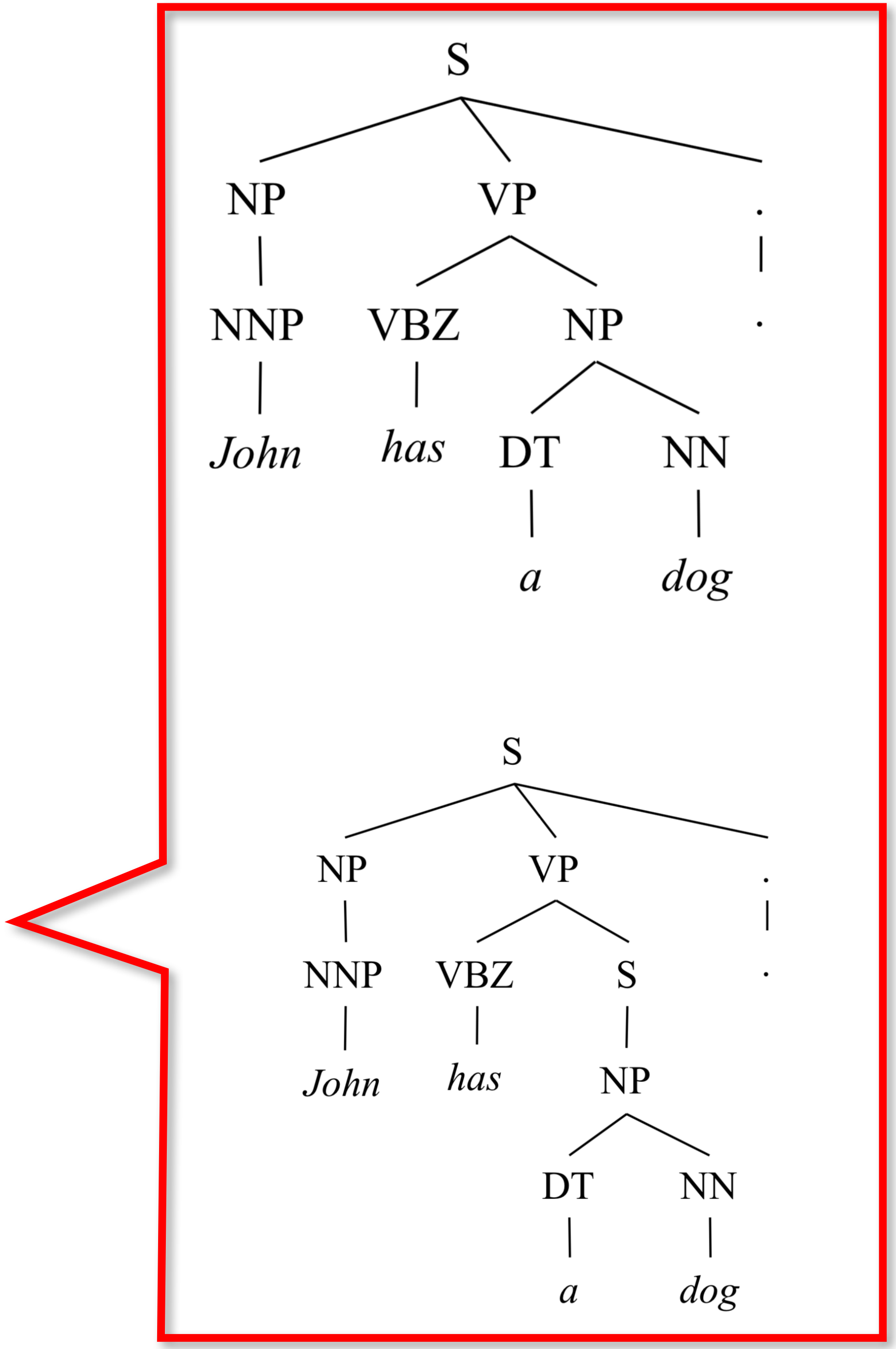
John has a dog . $\rightarrow$



John has a dog . $\rightarrow$ (S (NP NNP $)_{\mathrm{NP}}$ (VP VBZ (NP DT NN $)_{\mathrm{NP}}$ $)_{\mathrm{VP}}$. $)_{\mathrm{S}}$

Packed Forest





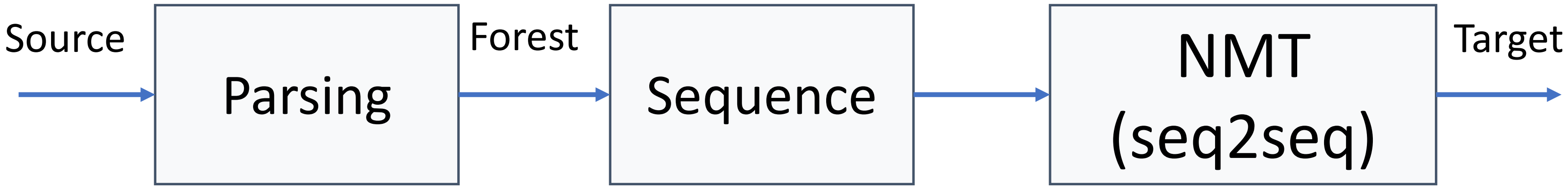
Our Method

Overview

- Use syntactic information explicitly
- Do not use tree/graph-structured network
- Use multiple parsing trees

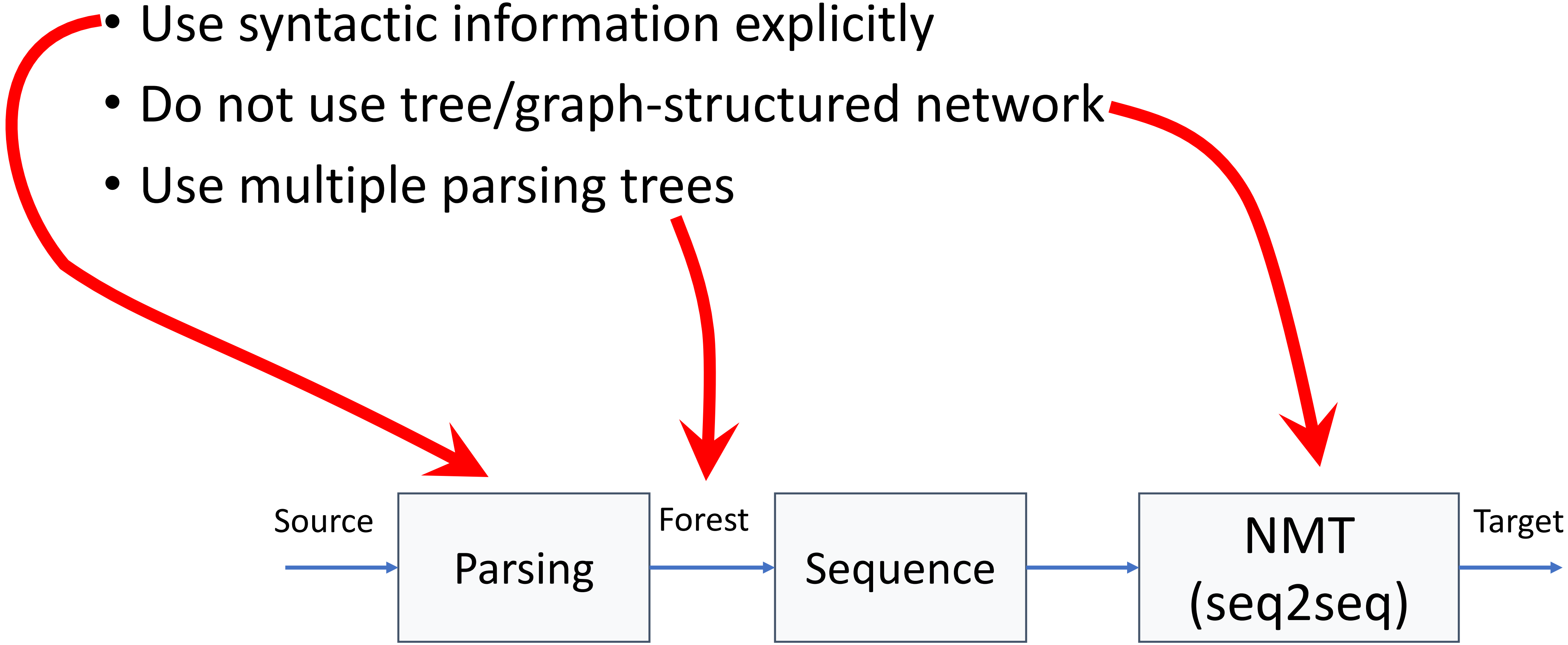
Overview

- Use syntactic information explicitly
- Do not use tree/graph-structured network
- Use multiple parsing trees



Overview

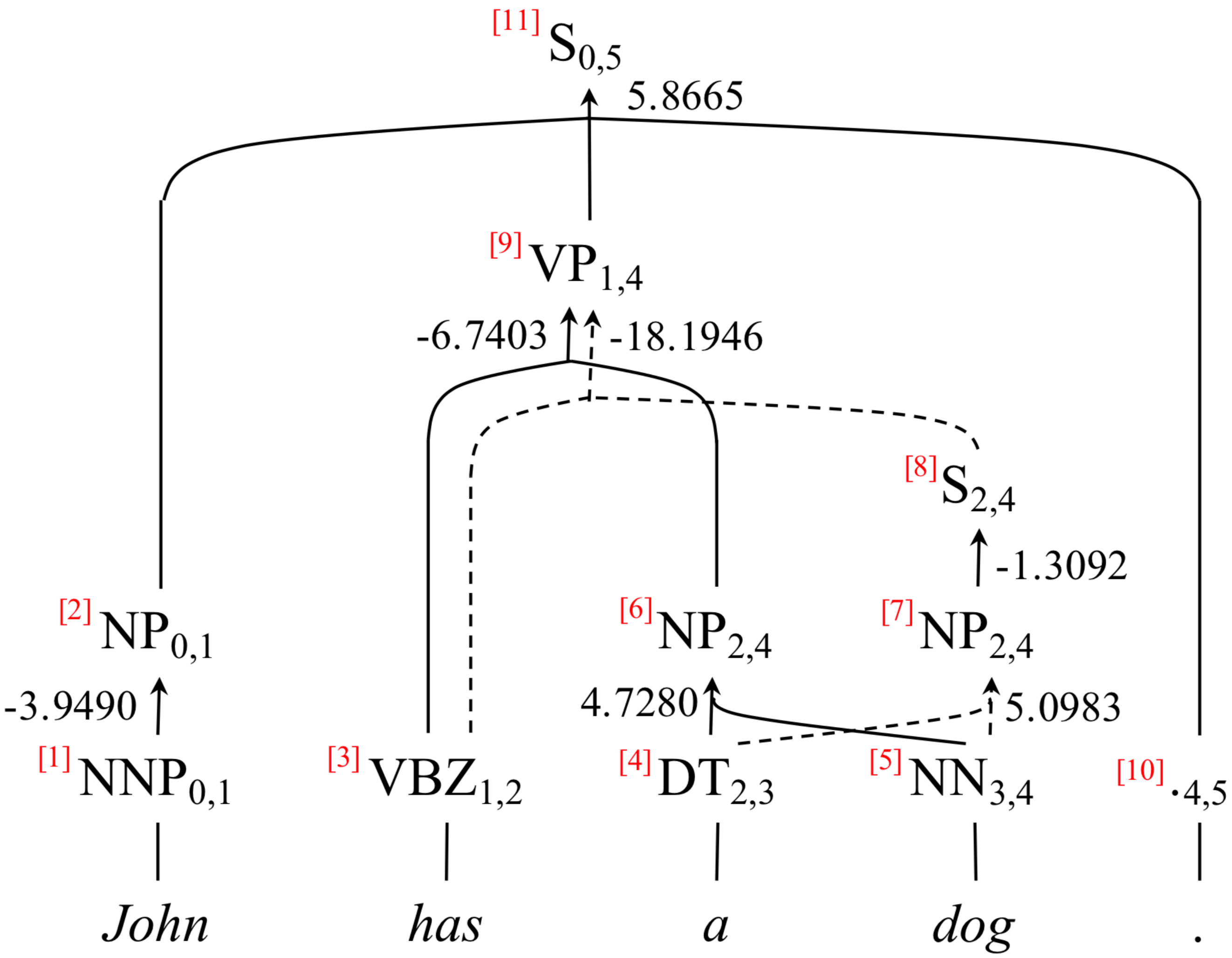
- Use syntactic information explicitly
- Do not use tree/graph-structured network
- Use multiple parsing trees



Forest Linearization

- Packed forest is directed acyclic graph, not tree
    - Fixed traversal order does not exist.

- Topological sort?
    - Outputs are not always optimal for MT.
    - Important information may be lost
        - Word sequential information
        - Parent-child information

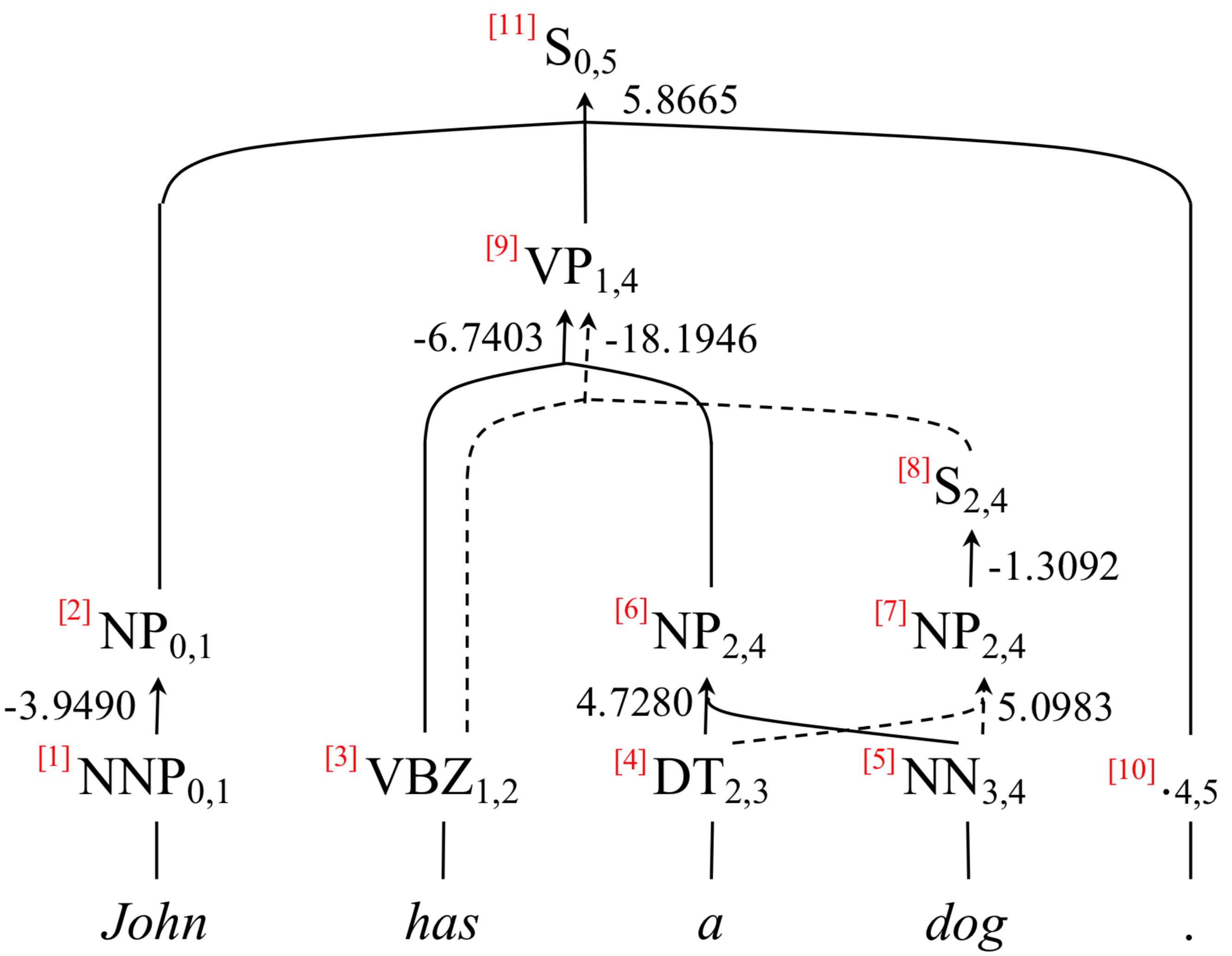


Word sequential information

[10]→[1]→[2]→…→[9]→[11]

Valid topological sort

Words are disordered



Parent-child information

[1]→[2]→…→[9]→[10]→[11]

Valid topological sort

Distances between [2][9][10] and [11] vary a lot.

```
function LinearizeForest(⟨V, E⟩, w)
    v ← FindRoot(V)
    r ← []
    ExpandSeq(v, r, ⟨V, E⟩, w)
    return r
```

```
function FindRoot(V)
    for v ∈ V do
        if v has no parent then
            return v
```

function LINEARIZEFOREST($\langle V, E \rangle, \mathbf{w}$)

  $v \leftarrow$ FINDROOT(V)

  $\mathbf{r} \leftarrow []$ ← Score Sequence

  EXPANDSEQ($v, \mathbf{r}, \langle V, E \rangle, \mathbf{w}$)

  return r

function FINDROOT(V)

  for $v \in V$ **do**

    if v has no parent **then**

      return v

procedure $\textsc{ExpandSeq}(v, \mathbf{r}, \langle V, E\rangle, \mathbf{w})$

    for $e \in E$ **do**

        if $head(e) = v$ **then**

            if $tails(e) \neq \emptyset$ **then**

                for $t \in \textsc{Sort}(tails(e))$ **do**

                    $\textsc{ExpandSeq}(t, \mathbf{r}, \langle V, E\rangle, \mathbf{w})$

                $l \leftarrow \textsc{LinearizeEdge}(head(e), \mathbf{w})$

                $\mathbf{r}$.append$(\langle l, \sigma(0.0)\rangle)$

                $l \leftarrow$ ©$\textsc{LinearizeEdges}(tails(e), \mathbf{w})$

                $\mathbf{r}$.append$(\langle l, \sigma(score(e))\rangle)$

            else

                $l \leftarrow \textsc{LinearizeEdge}(head(e), \mathbf{w})$

                $\mathbf{r}$.append$(\langle l, \sigma(0.0)\rangle)$

procedure $\text{EXPANDSEQ}(v, \mathbf{r}, \langle V, E \rangle, \mathbf{w})$

  for $e \in E$ **do**

    if $head(e) = v$ **then**

      if $tails(e) \neq \emptyset$ **then**

        for $t \in \text{SORT}(tails(e))$ **do**

          $\text{EXPANDSEQ}(t, \mathbf{r}, \langle V, E \rangle, \mathbf{w})$

        $l \leftarrow \text{LINEARIZEEDGE}(head(e), \mathbf{w})$

        $\mathbf{r}.\text{append}(\langle l, \sigma(0.0) \rangle)$

        $l \leftarrow \copyright\text{LINEARIZEEDGES}(tails(e), \mathbf{w})$

        $\mathbf{r}.\text{append}(\langle l, \sigma(score(e)) \rangle)$

      else

        $l \leftarrow \text{LINEARIZEEDGE}(head(e), \mathbf{w})$

        $\mathbf{r}.\text{append}(\langle l, \sigma(0.0) \rangle)$

Recursive Call

Terminate

procedure $\textsc{ExpandSeq}(v, \mathbf{r}, \langle V, E \rangle, \mathbf{w})$

  for $e \in E$ **do**

    if $head(e) = v$ **then**

      if $tails(e) \neq \emptyset$ **then**

        for $t \in \textsc{Sort}(tails(e))$ **do**

          $\textsc{ExpandSeq}(t, \mathbf{r}, \langle V, E \rangle, \mathbf{w})$

        $l \leftarrow \textsc{LinearizeEdge}(head(e), \mathbf{w})$

        $\mathbf{r}.\text{append}(\langle l, \sigma(0.0) \rangle)$

        $l \leftarrow \copyright\textsc{LinearizeEdges}(tails(e), \mathbf{w})$

        $\mathbf{r}.\text{append}(\langle l, \sigma(score(e)) \rangle)$

      else

        $l \leftarrow \textsc{LinearizeEdge}(head(e), \mathbf{w})$

        $\mathbf{r}.\text{append}(\langle l, \sigma(0.0) \rangle)$

Word sequential information

Recursive Call

Terminate

procedure EXPANDSEQ(v, $\mathbf{r}$, $\langle V, E\rangle$, $\mathbf{w}$)
    for $e \in E$ **do**
        if $head(e) = v$ **then**
            if $tails(e) \neq \emptyset$ **then**
                for $t \in$ SORT($tails(e)$) **do**
                    EXPANDSEQ(t, $\mathbf{r}$, $\langle V, E\rangle$, $\mathbf{w}$)
                $l \leftarrow$ LINEARIZEEDGE($head(e)$, $\mathbf{w}$)
                $\mathbf{r}$.append($\langle l, \sigma(0.0)\rangle$)
                $l \leftarrow$ ©LINEARIZEEDGES($tails(e)$, $\mathbf{w}$)
                $\mathbf{r}$.append($\langle l, \sigma(score(e))\rangle$)
            else
                $l \leftarrow$ LINEARIZEEDGE($head(e)$, $\mathbf{w}$)
                $\mathbf{r}$.append($\langle l, \sigma(0.0)\rangle$)

Word sequential information

Recursive Call

Parent-child information

Terminate

procedure $\textsc{ExpandSeq}(v, \mathbf{r}, \langle V, E\rangle, \mathbf{w})$
  for $e \in E$ **do**
    if $head(e) = v$ **then**
      if $tails(e) \neq \emptyset$ **then**
        for $t \in \textsc{Sort}(tails(e))$ **do**
          $\textsc{ExpandSeq}(t, \mathbf{r}, \langle V, E\rangle, \mathbf{w})$
        $l \leftarrow \textsc{LinearizeEdge}(head(e), \mathbf{w})$
        $\mathbf{r}.\text{append}(\langle l, \sigma(0.0)\rangle)$
        $l \leftarrow$ ⓒ$\textsc{LinearizeEdges}(tails(e), \mathbf{w})$
        $\mathbf{r}.\text{append}(\langle l, \sigma(score(e))\rangle)$
      else
        $l \leftarrow \textsc{LinearizeEdge}(head(e), \mathbf{w})$
        $\mathbf{r}.\text{append}(\langle l, \sigma(0.0)\rangle)$

Word sequential information

Recursive Call

Parent-child information

Children Mark

Terminate

Linearization of Leaf nodes

function LINEARIZEEDGE($X_{i,j}$, $\mathbf{w}$)
return $X \otimes (\bigodot_{k=i}^{j-1} w_k)$

function LINEARIZEEDGES($\mathbf{v}$, $\mathbf{w}$)
return $\bigoplus_{v \in \mathbf{v}}$ LINEARIZEEDGE(v, $\mathbf{w}$)

Linearization of Leaf nodes

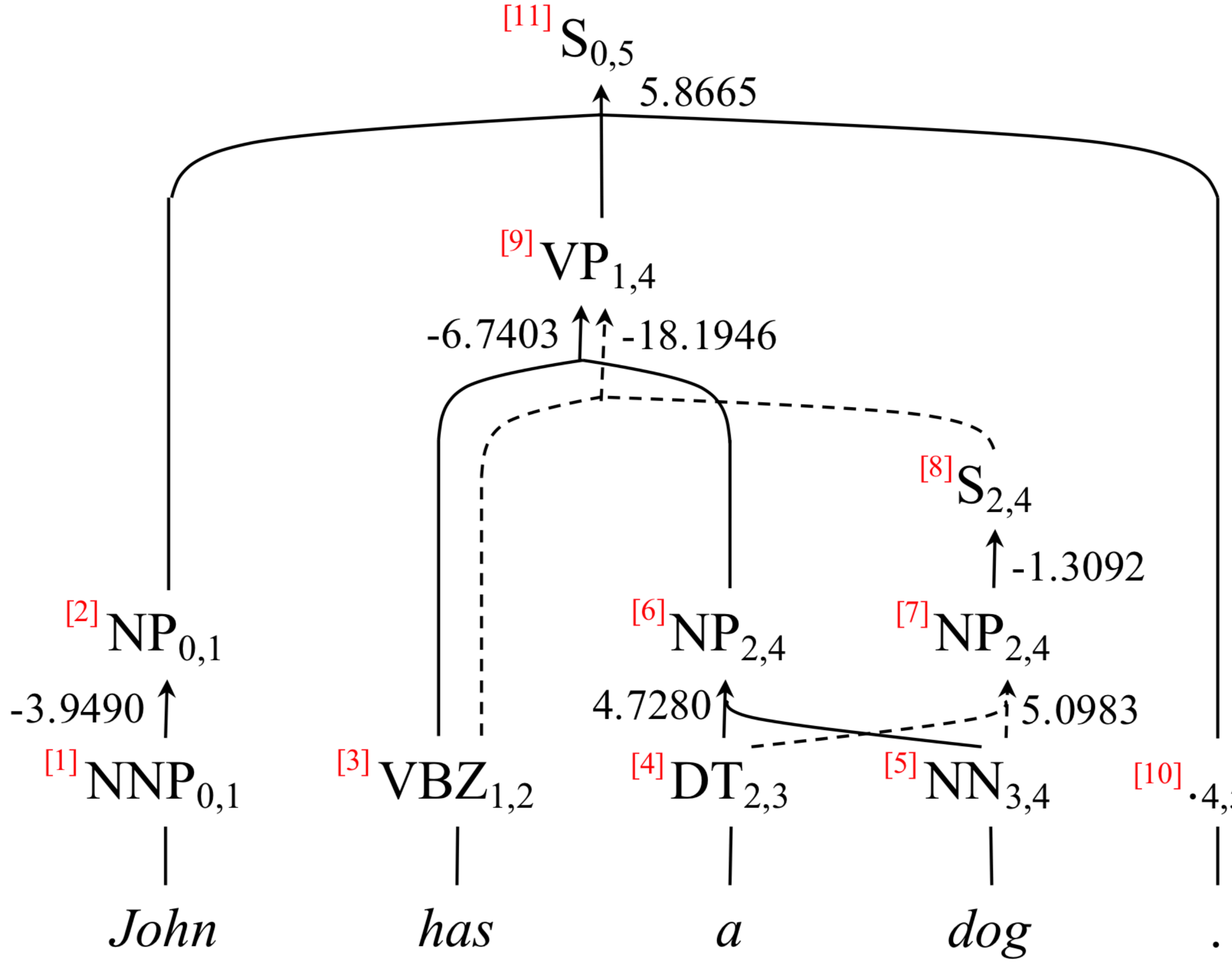
$$\textbf{function } \textsc{LinearizeEdge}(X_{i,j}, \mathbf{w})$$
$$\textbf{return } X \otimes \left(\bigodot_{k=i}^{j-1} w_k\right)$$

Connect phrase and constituent label

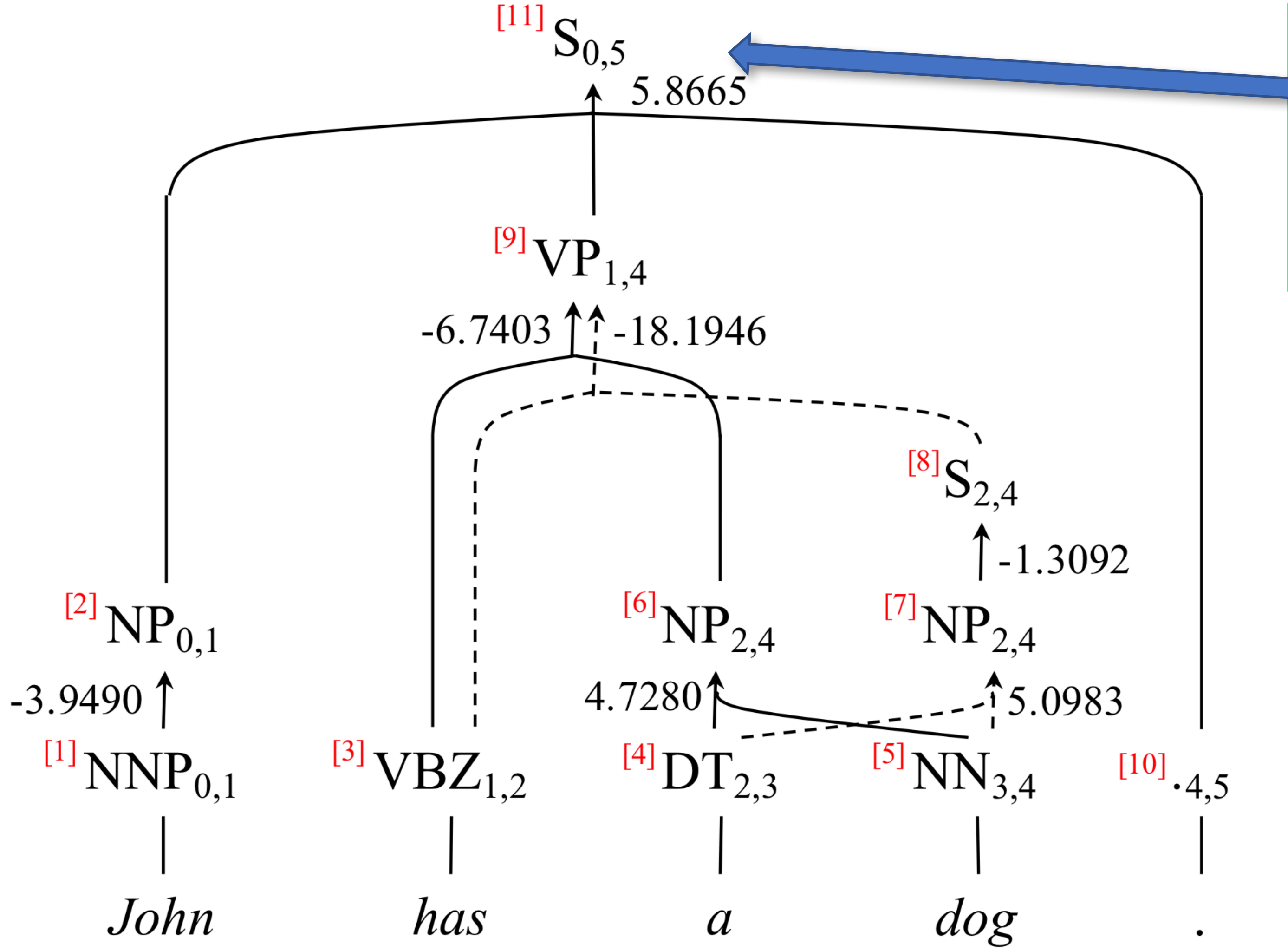
Connect words

$$\textbf{function } \textsc{LinearizeEdges}(\mathbf{v}, \mathbf{w})$$
$$\textbf{return } \bigoplus_{v \in \mathbf{v}} \textsc{LinearizeEdge}(v, \mathbf{w})$$

Connect hyperedges

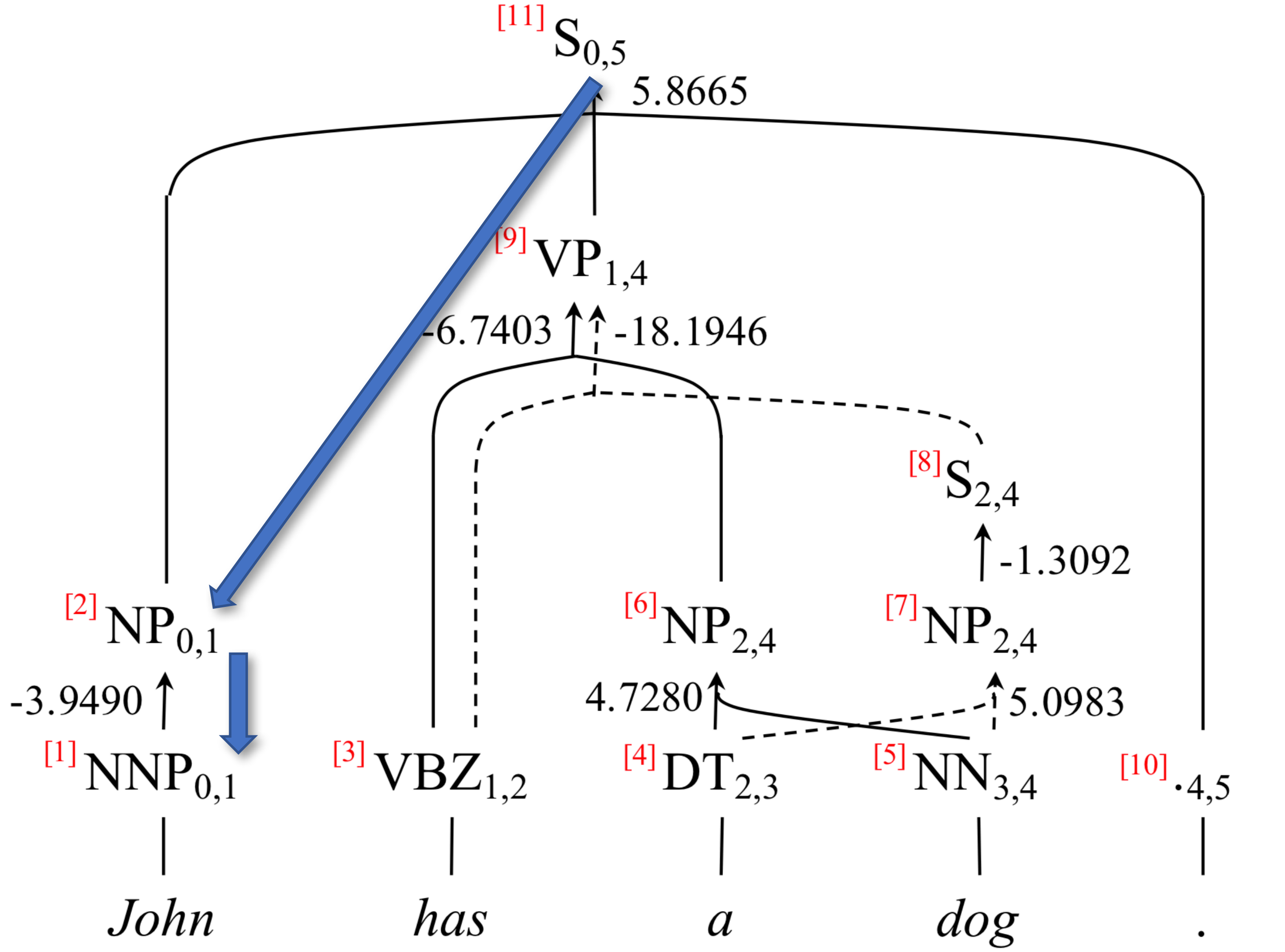


NNP⊗*John* / NP⊗*John* / ⓒNNP⊗*John* / VBZ⊗*has* / DT⊗*a* / NN⊗*dog* / NP⊗*a*⊙*dog* / ⓒDT⊗*a*⊕NN⊗*dog* / NP⊗*a*⊙*dog* / ⓒDT⊗*a*⊕NN⊗*dog* / S⊗*a*⊙*dog* / ⓒNP⊗*a*⊙*dog* / VP⊗*has*⊙*a*⊙*dog* / ⓒVBZ⊗*has*⊕NP⊗*a*⊙*dog* / ⓒVBZ⊗*has*⊕S⊗*a*⊙*dog* / .⊗. / S⊗*John*⊙*has*⊙*a*⊙*dog*⊙. / ⓒNP⊗*John*⊕VP⊗*has*⊙*a*⊙*dog*⊕.⊗.



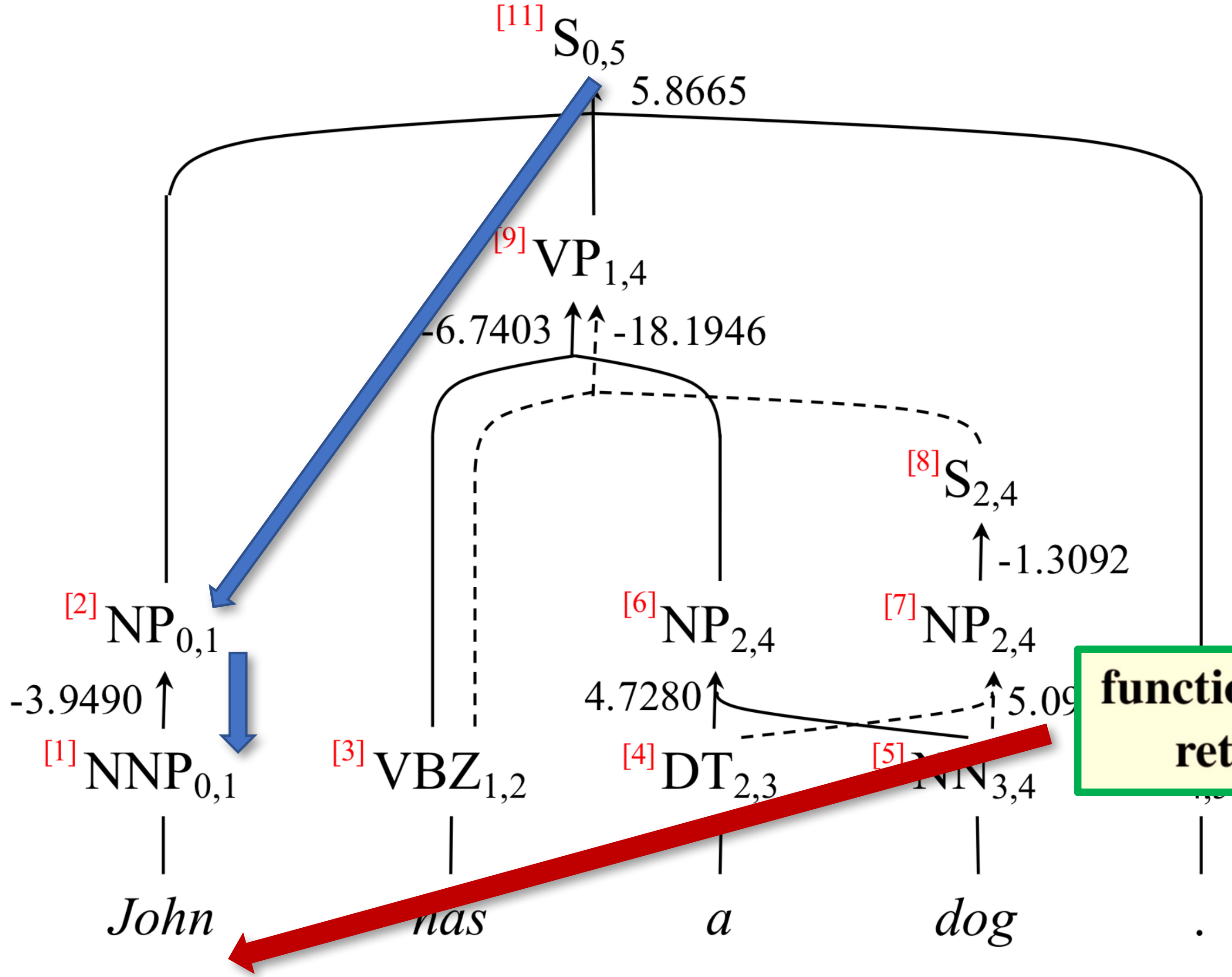
function LINEARIZEFOREST($\langle V, E\rangle$, $\mathbf{w}$)
    $v \leftarrow$ FINDROOT(V)
    $\mathbf{r} \leftarrow []$
    EXPANDSEQ(v, $\mathbf{r}$, $\langle V, E\rangle$, $\mathbf{w}$)
    return $\mathbf{r}$

NNP⊗*John* / NP⊗*John* / ⓒNNP⊗*John* / VBZ⊗*has* / DT⊗*a* / NN⊗*dog* / NP⊗*a*⊙*dog* / ⓒDT⊗*a*⊕NN⊗*dog* / NP⊗*a*⊙*dog* / ⓒDT⊗*a*⊕NN⊗*dog* / S⊗*a*⊙*dog* / ⓒNP⊗*a*⊙*dog* / VP⊗*has*⊙*a*⊙*dog* / ⓒVBZ⊗*has*⊕NP⊗*a*⊙*dog* / ⓒVBZ⊗*has*⊕S⊗*a*⊙*dog* / .⊗. / S⊗*John*⊙*has*⊙*a*⊙*dog*⊙. / ⓒNP⊗*John*⊕VP⊗*has*⊙*a*⊙*dog*⊕.⊗.




```
procedure ExpandSeq(v, r, ⟨V, E⟩, w)
    for e ∈ E do
        if head(e) = v then
            if tails(e) ≠ ∅ then
                for t ∈ Sort(tails(e)) do
                    ExpandSeq(t, r, ⟨V, E⟩, w)
                l ← LinearizeEdge(head(e), w)
                r.append(⟨l, σ(0.0)⟩)
                l ← ©LinearizeEdges(tails(e), w)
                r.append(⟨l, σ(score(e))⟩)
            else
                l ← LinearizeEdge(head(e), w)
                r.append(⟨l, σ(0.0)⟩)
```

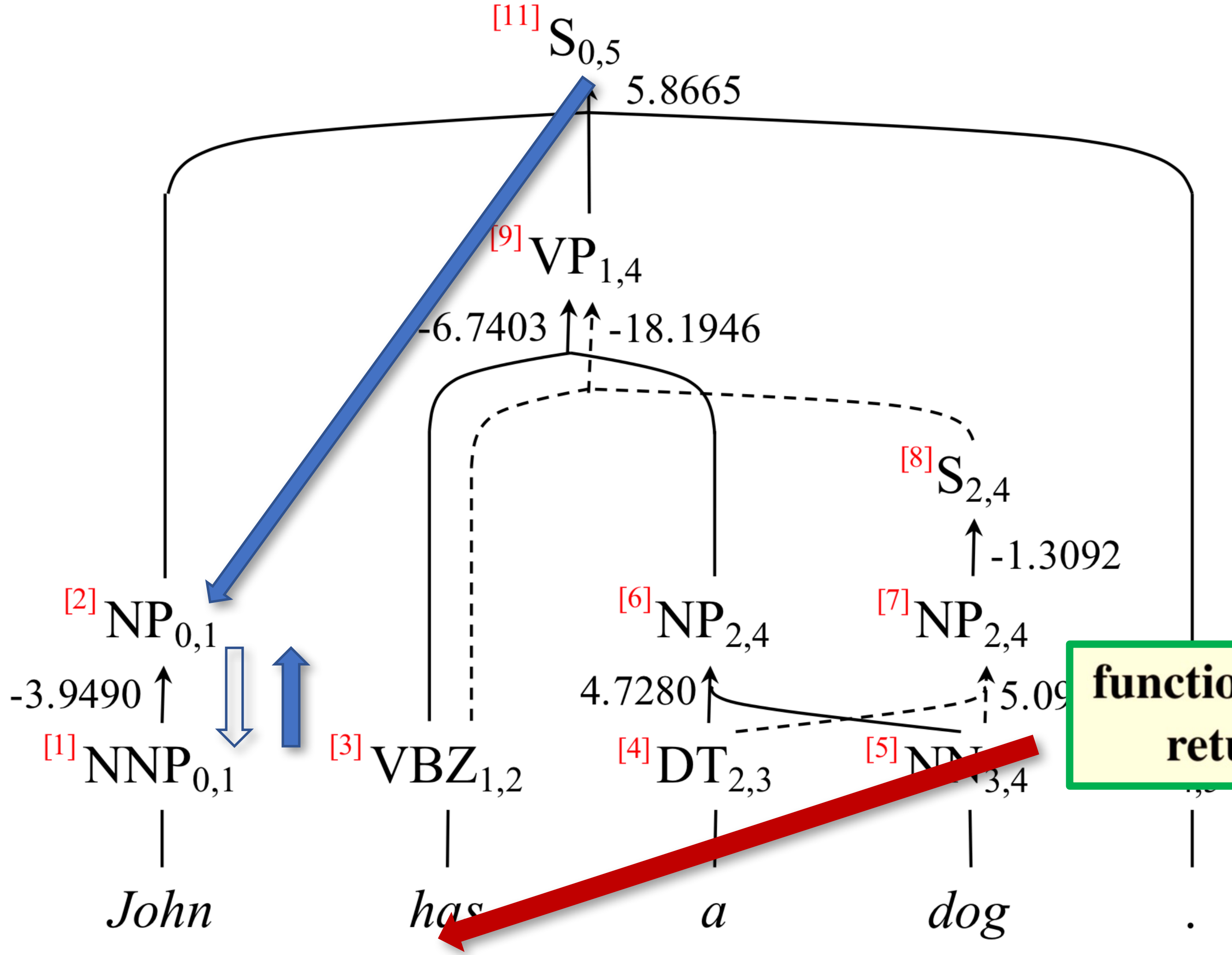
NNP⊗*John* / NP⊗*John* / ©NNP⊗*John* / VBZ⊗*has* / DT⊗*a* / NN⊗*dog* / NP⊗*a*⊙*dog* / ©DT⊗*a*⊕NN⊗*dog* / NP⊗*a*⊙*dog* / ©DT⊗*a*⊕NN⊗*dog* / S⊗*a*⊙*dog* / ©NP⊗*a*⊙*dog* / VP⊗*has*⊙*a*⊙*dog* / ©VBZ⊗*has*⊕NP⊗*a*⊙*dog* / ©VBZ⊗*has*⊕S⊗*a*⊙*dog* / .⊗. / S⊗*John*⊙*has*⊙*a*⊙*dog*⊙. / ©NP⊗*John*⊕VP⊗*has*⊙*a*⊙*dog*⊕.⊗.



procedure ExpandSeq($v, \mathbf{r}, \langle V, E\rangle, \mathbf{w}$)
  for $e \in E$ **do**
    if $head(e) = v$ **then**
      if $tails(e) \neq \emptyset$ **then**
        for $t \in$ Sort($tails(e)$) **do**
          ExpandSeq($t, \mathbf{r}, \langle V, E\rangle, \mathbf{w}$)
        $l \leftarrow$ LinearizeEdge($head(e), \mathbf{w}$)
        $\mathbf{r}$.append($\langle l, \sigma(0.0)\rangle$)
        $l \leftarrow$ ©LinearizeEdges($tails(e), \mathbf{w}$)
        $\mathbf{r}$.append($\langle l, \sigma(score(e))\rangle$)
      else
        $l \leftarrow$ LinearizeEdge($head(e), \mathbf{w}$)
        $\mathbf{r}$.append($\langle l, \sigma(0.0)\rangle$)

function LinearizeEdge($X_{i,j}, \mathbf{w}$)
  return $X \otimes (\odot_{k=i}^{j-1} w_k)$

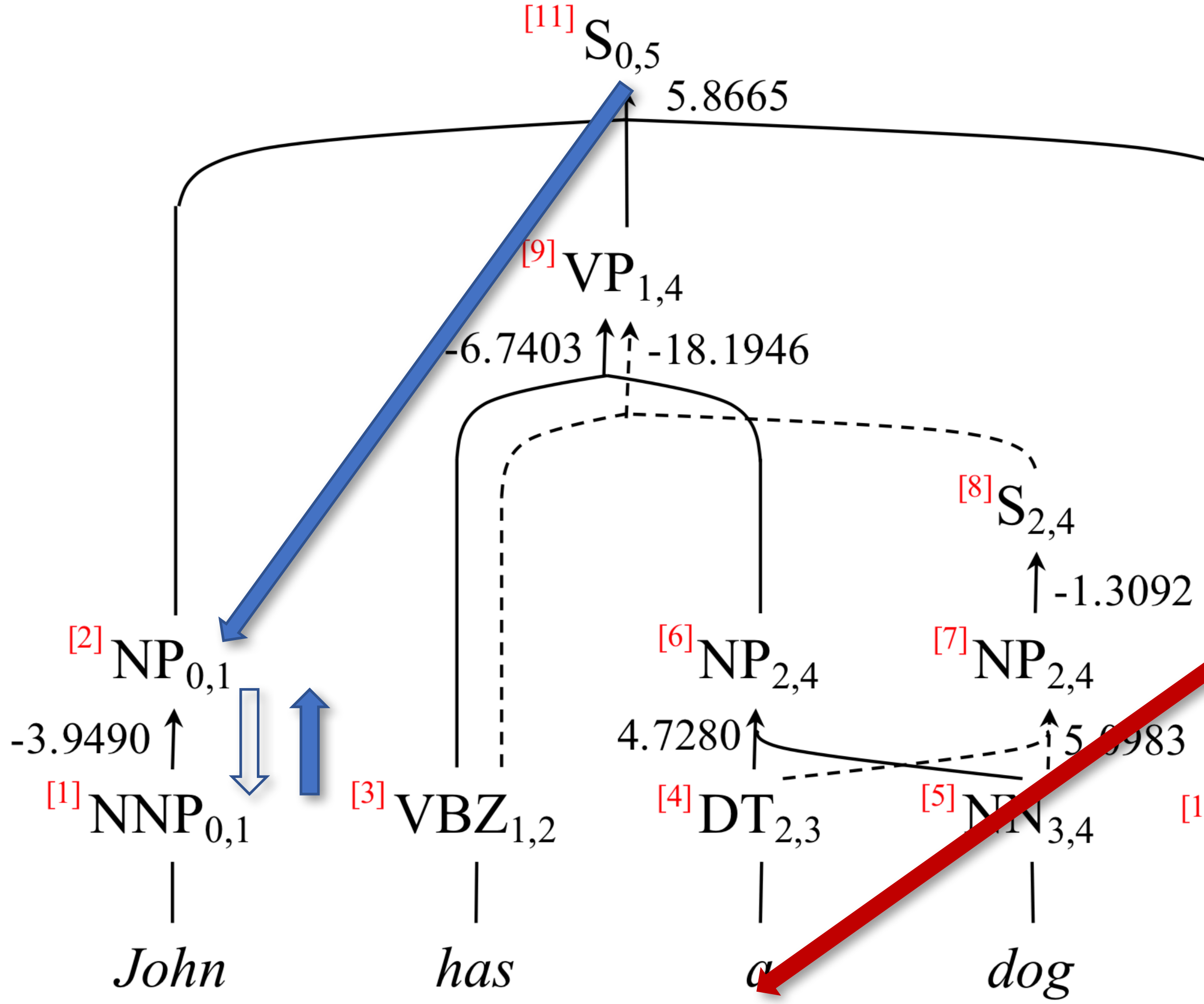
NNP⊗*John* / NP⊗*John* / ©NNP⊗*John* / VBZ⊗*has* / DT⊗*a* / NN⊗*dog* / NP⊗*a*⊙*dog* / ©DT⊗*a*⊕NN⊗*dog* / NP⊗*a*⊙*dog* / ©DT⊗*a*⊕NN⊗*dog* / S⊗*a*⊙*dog* / ©NP⊗*a*⊙*dog* / VP⊗*has*⊙*a*⊙*dog* / ©VBZ⊗*has*⊕NP⊗*a*⊙*dog* / ©VBZ⊗*has*⊕S⊗*a*⊙*dog* / .⊗. / S⊗*John*⊙*has*⊙*a*⊙*dog*⊙. / ©NP⊗*John*⊕VP⊗*has*⊙*a*⊙*dog*⊕.⊗.



procedure $\textsc{ExpandSeq}(v, \mathbf{r}, \langle V, E\rangle, \mathbf{w})$
    for $e \in E$ **do**
        if $head(e) = v$ **then**
            if $tails(e) \neq \emptyset$ **then**
                for $t \in \textsc{Sort}(tails(e))$ **do**
                    $\textsc{ExpandSeq}(t, \mathbf{r}, \langle V, E\rangle, \mathbf{w})$
                $l \leftarrow \textsc{LinearizeEdge}(head(e), \mathbf{w})$
                $\mathbf{r}.\text{append}(\langle l, \sigma(0.0)\rangle)$
                $l \leftarrow ⓒ\textsc{LinearizeEdges}(tails(e), \mathbf{w})$
                $\mathbf{r}.\text{append}(\langle l, \sigma(score(e))\rangle)$
            else
                $l \leftarrow \textsc{LinearizeEdge}(head(e), \mathbf{w})$
                $\mathbf{r}.\text{append}(\langle l, \sigma(0.0)\rangle)$

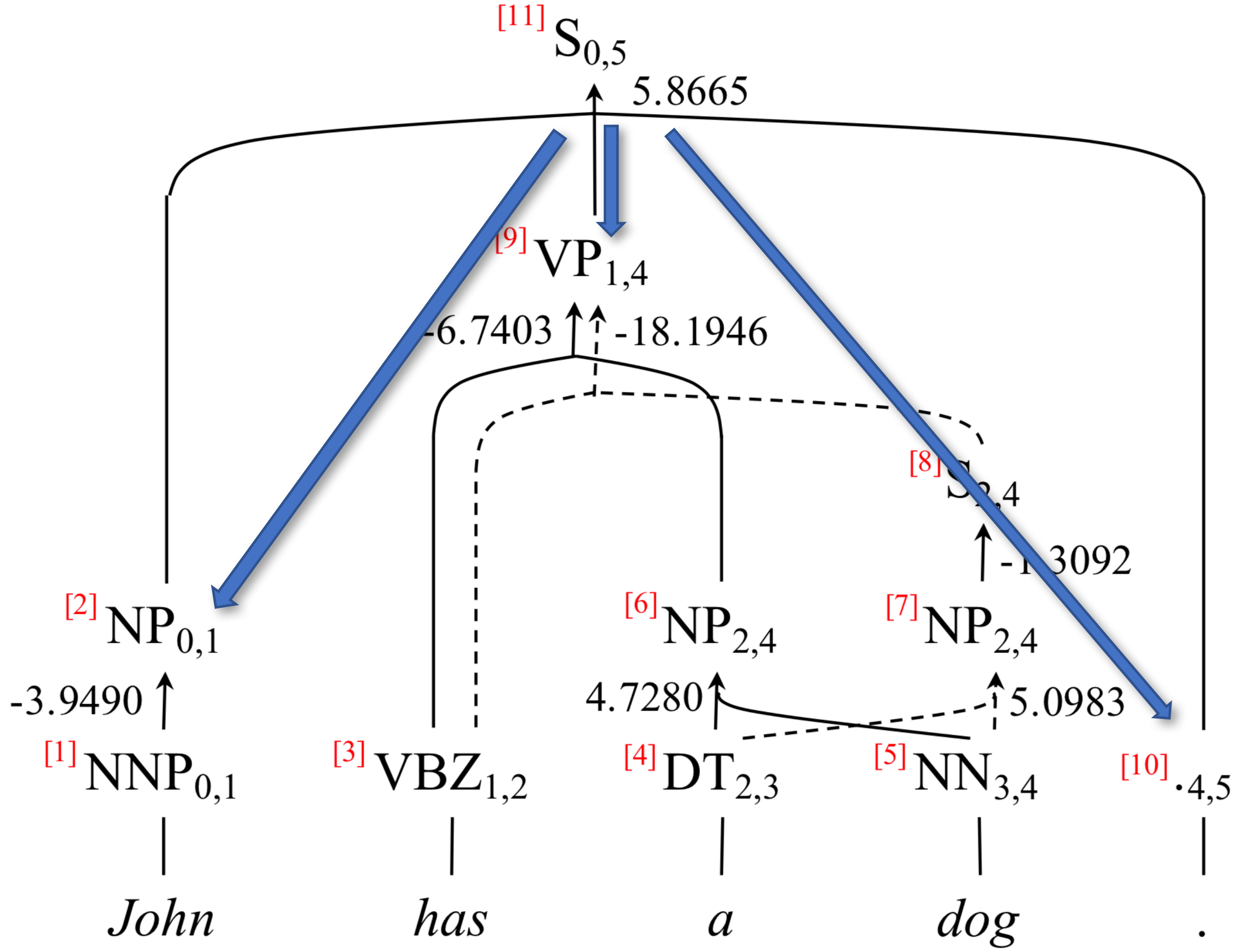
function $\textsc{LinearizeEdge}(X_{i,j}, \mathbf{w})$
    return $X \otimes (\odot_{k=i}^{j-1} w_k)$

NNP⊗*John* / NP⊗*John* / ⓒNNP⊗*John* / VBZ⊗*has* / DT⊗*a* / NN⊗*dog* / NP⊗*a*⊙*dog* / ⓒDT⊗*a*⊕NN⊗*dog* / NP⊗*a*⊙*dog* / ⓒDT⊗*a*⊕NN⊗*dog* / S⊗*a*⊙*dog* / ⓒNP⊗*a*⊙*dog* / VP⊗*has*⊙*a*⊙*dog* / ⓒVBZ⊗*has*⊕NP⊗*a*⊙*dog* / ⓒVBZ⊗*has*⊕S⊗*a*⊙*dog* / .⊗. / S⊗*John*⊙*has*⊙*a*⊙*dog*⊙. / ⓒNP⊗*John*⊕VP⊗*has*⊙*a*⊙*dog*⊕.⊗.



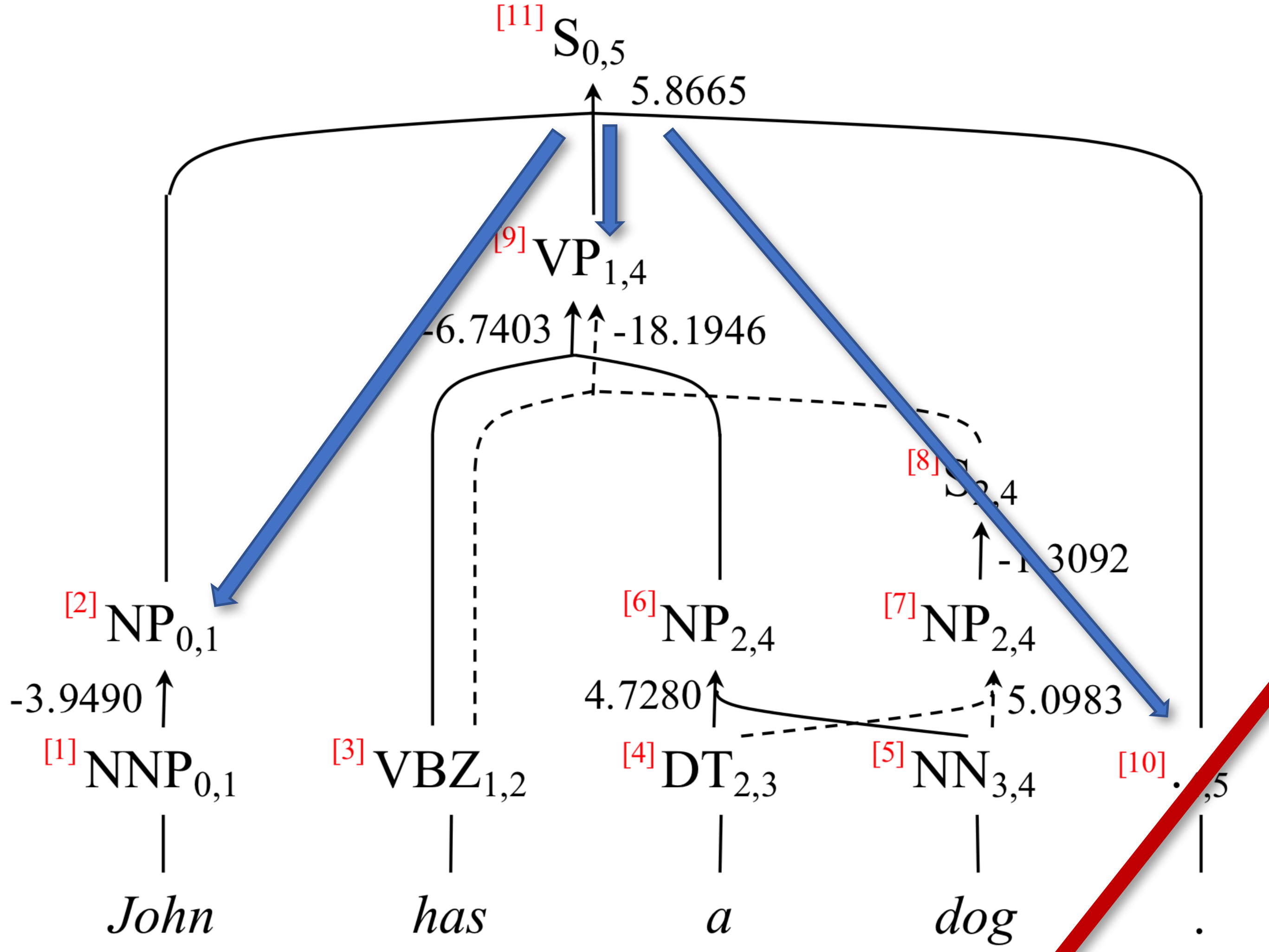

```
procedure ExpandSeq(v, r, ⟨V, E⟩, w)
    for e ∈ E do
        if head(e) = v then
            if tails(e) ≠ ∅ then
                for t ∈ Sort(tails(e)) do
                    ExpandSeq(t, r, ⟨V, E⟩, w)
                l ← LinearizeEdge(head(e), w)
                r.append(⟨l, σ(0.0)⟩)
                l ← ©LinearizeEdges(tails(e), w)
                r.append(⟨l, σ(score(e))⟩)
            else
                l ← LinearizeEdge(head(e), w)
                r.append(⟨l, σ(0.0)⟩)
```

NNP⊗*John* / NP⊗*John* / ©NNP⊗*John* / VBZ⊗*has* / DT⊗*a* / NN⊗*dog* / NP⊗*a*⊙*dog* / ©DT⊗*a*⊕NN⊗*dog* / NP⊗*a*⊙*dog* / ©DT⊗*a*⊕NN⊗*dog* / S⊗*a*⊙*dog* / ©NP⊗*a*⊙*dog* / VP⊗*has*⊙*a*⊙*dog* / ©VBZ⊗*has*⊕NP⊗*a*⊙*dog* / ©VBZ⊗*has*⊕S⊗*a*⊙*dog* / .⊗. / S⊗*John*⊙*has*⊙*a*⊙*dog*⊙. / ©NP⊗*John*⊕VP⊗*has*⊙*a*⊙*dog*⊕.⊗.




```
procedure EXPANDSEQ(v, r, ⟨V, E⟩, w)
    for e ∈ E do
        if head(e) = v then
            if tails(e) ≠ ∅ then
                for t ∈ SORT(tails(e)) do
                    EXPANDSEQ(t, r, ⟨V, E⟩, w)
                l ← LINEARIZEEDGE(head(e), w)
                r.append(⟨l, σ(0.0)⟩)
                l ← ©LINEARIZEEDGES(tails(e), w)
                r.append(⟨l, σ(score(e))⟩)
            else
                l ← LINEARIZEEDGE(head(e), w)
                r.append(⟨l, σ(0.0)⟩)
```

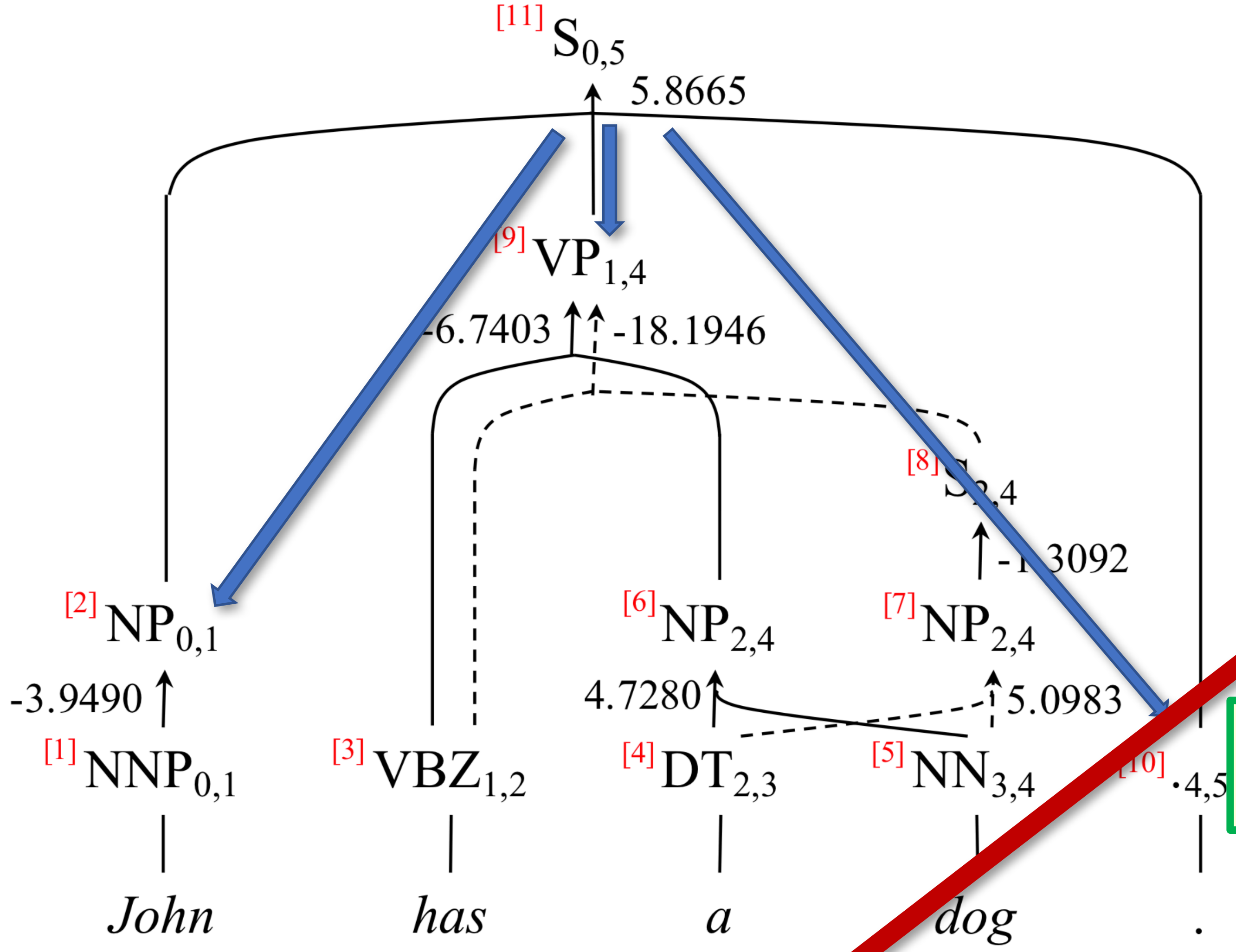
NNP⊗*John* / NP⊗*John* / ©NNP⊗*John* / VBZ⊗*has* / DT⊗*a* / NN⊗*dog* / NP⊗*a*⊙*dog* / ©DT⊗*a*⊕NN⊗*dog* / NP⊗*a*⊙*dog* / ©DT⊗*a*⊕NN⊗*dog* / S⊗*a*⊙*dog* / ©NP⊗*a*⊙*dog* / VP⊗*has*⊙*a*⊙*dog* / ©VBZ⊗*has*⊕NP⊗*a*⊙*dog* / ©VBZ⊗*has*⊕S⊗*a*⊙*dog* / .⊗. / S⊗*John*⊙*has*⊙*a*⊙*dog*⊙. / ©NP⊗*John*⊕VP⊗*has*⊙*a*⊙*dog*⊕.⊗.



procedure EXPANDSEQ($v, \mathbf{r}, \langle V, E\rangle, \mathbf{w}$)
  for $e \in E$ **do**
    if $head(e) = v$ **then**
      if $tails(e) \neq \emptyset$ **then**
        for $t \in$ SORT($tails(e)$) **do**
          EXPANDSEQ($t, \mathbf{r}, \langle V, E\rangle, \mathbf{w}$)
        $l \leftarrow$ LINEARIZEEDGE($head(e), \mathbf{w}$)
        $\mathbf{r}$.append($\langle l, \sigma(0.0)\rangle$)
        $l \leftarrow$ ⓒLINEARIZEEDGES($tails(e), \mathbf{w}$)
        $\mathbf{r}$.append($\langle l, \sigma(score(e))\rangle$)
      else
        $l \leftarrow$ LINEARIZEEDGE($head(e), \mathbf{w}$)
        $\mathbf{r}$.append($\langle l, \sigma(0.0)\rangle$)

function LINEARIZEEDGE($X_{i,j}, \mathbf{w}$)
  return $X \otimes (\odot_{k=i}^{j-1} w_k)$

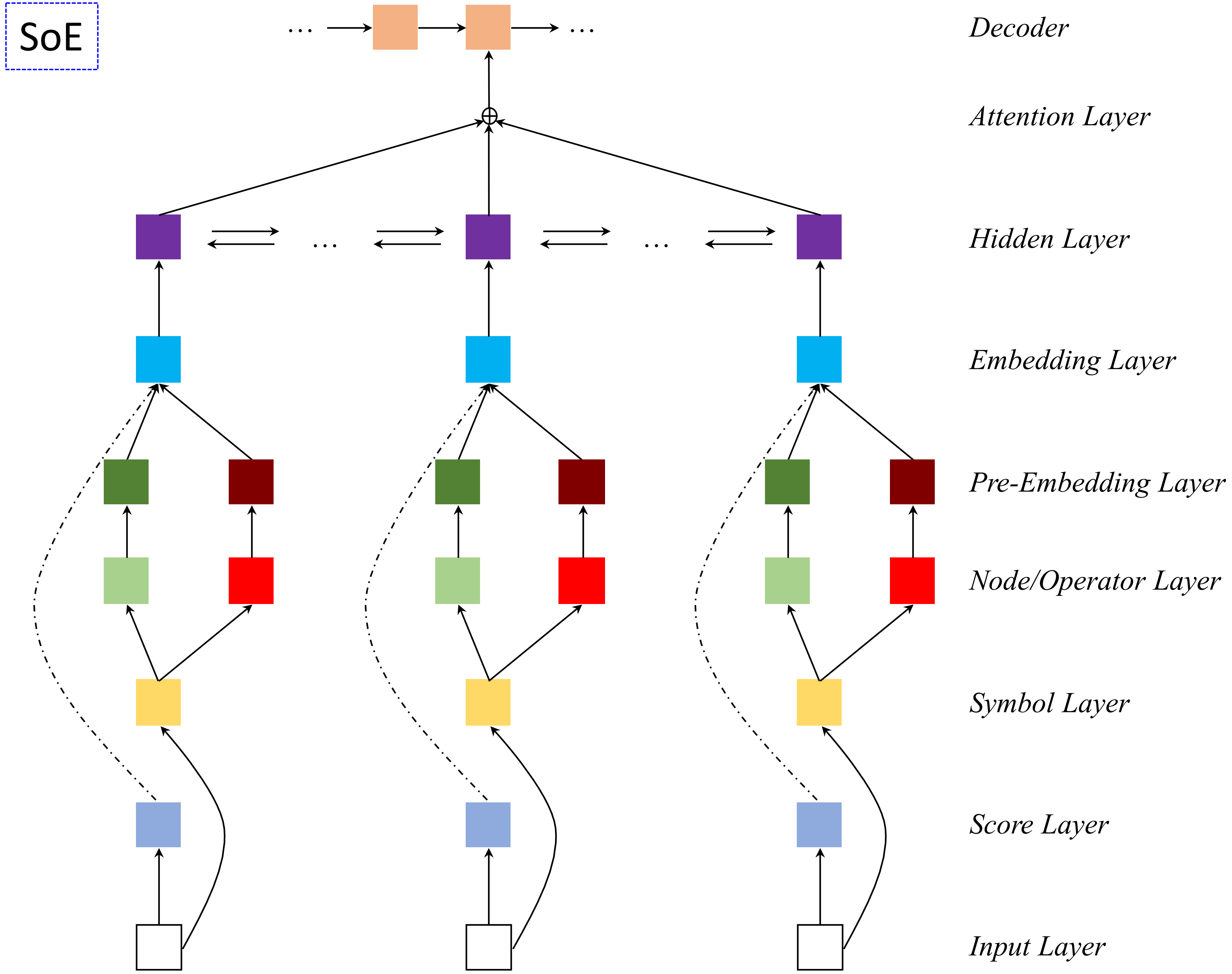
NNP⊗*John* / NP⊗*John* / ⓒNNP⊗*John* / VBZ⊗*has* / DT⊗*a* / NN⊗*dog* / NP⊗*a*⊙*dog* / ⓒDT⊗*a*⊕NN⊗*dog* / NP⊗*a*⊙*dog* / ⓒDT⊗*a*⊕NN⊗*dog* / S⊗*a*⊙*dog* / ⓒNP⊗*a*⊙*dog* / VP⊗*has*⊙*a*⊙*dog* / ⓒVBZ⊗*has*⊕NP⊗*a*⊙*dog* / ⓒVBZ⊗*has*⊕S⊗*a*⊙*dog* / .⊗. / S⊗*John*⊙*has*⊙*a*⊙*dog*⊙. / ⓒNP⊗*John*⊕VP⊗*has*⊙*a*⊙*dog*⊕.⊗.

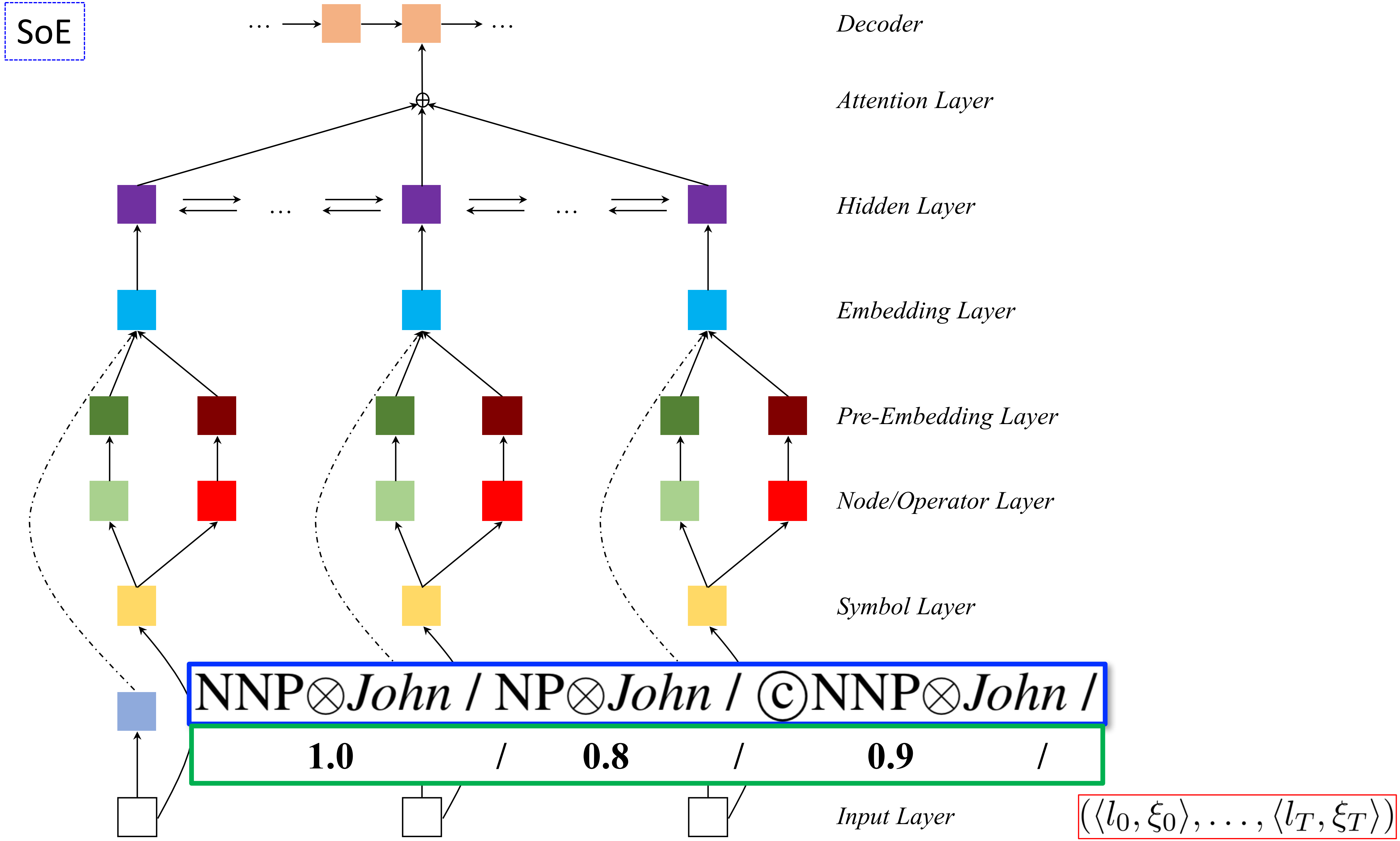


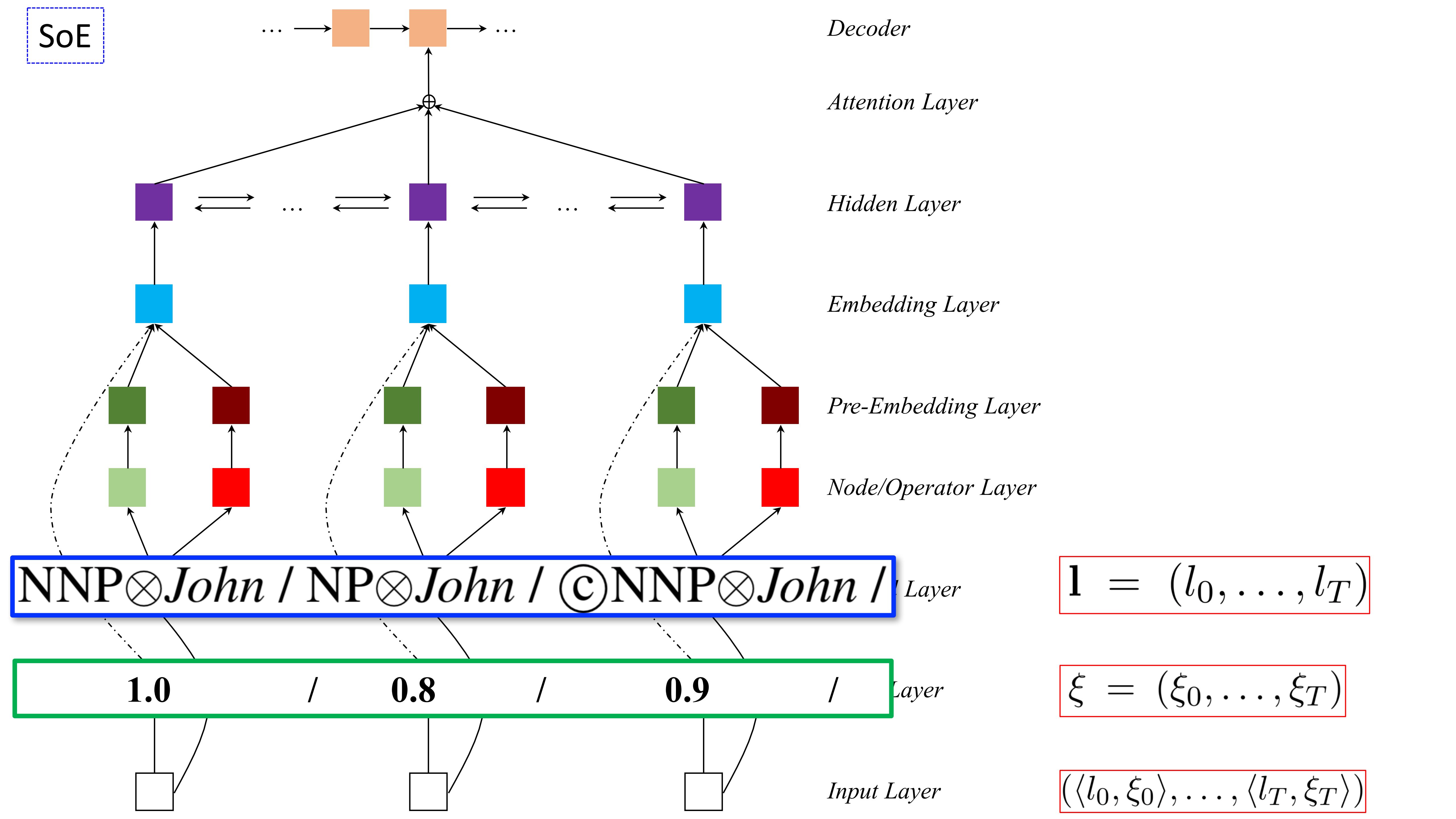

```
procedure ExpandSeq(v, r, ⟨V, E⟩, w)
    for e ∈ E do
        if head(e) = v then
            if tails(e) ≠ ∅ then
                for t ∈ Sort(tails(e)) do
                    ExpandSeq(t, r, ⟨V, E⟩, w)
                l ← LinearizeEdge(head(e), w)
                r.append(⟨l, σ(0.0)⟩)
                l ← ©LinearizeEdges(tails(e), w)
                r.append(⟨l, σ(score(e))⟩)
            else
                l ← LinearizeEdge(head(e), w)
                r.append(⟨l, σ(0.0)⟩)
```

```
function LinearizeEdges(v, w)
    return ⊕_{v∈v} LinearizeEdge(v, w)
```

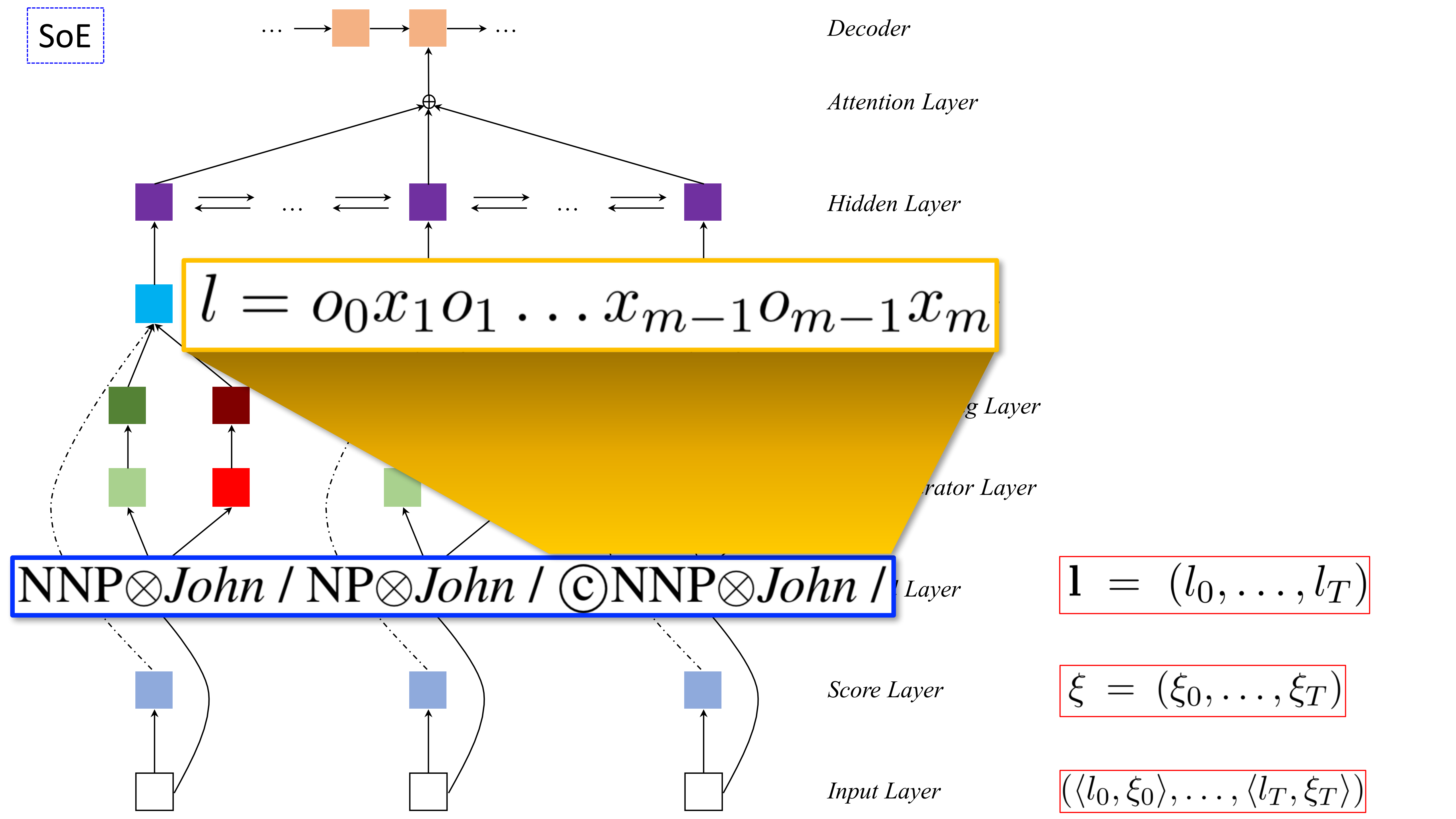
NNP⊗*John* / NP⊗*John* / ©NNP⊗*John* / VBZ⊗*has* / DT⊗*a* / NN⊗*dog* / NP⊗*a*⊙*dog* / ©DT⊗*a*⊕NN⊗*dog* / NP⊗*a*⊙*dog* / ©DT⊗*a*⊕NN⊗*dog* / S⊗*a*⊙*dog* / ©NP⊗*a*⊙*dog* / VP⊗*has*⊙*a*⊙*dog* / ©VBZ⊗*has*⊕NP⊗*a*⊙*dog* / ©VBZ⊗*has*⊕S⊗*a*⊙*dog* / .⊗. / S⊗*John*⊙*has*⊙*a*⊙*dog*⊙. / ©NP⊗*John*⊕VP⊗*has*⊙*a*⊙*dog*⊕.⊗.


SoE
Decoder
Attention Layer
Hidden Layer
Embedding Layer
Pre-Embedding Layer
Node/Operator Layer
Symbol Layer
Score Layer
Input Layer

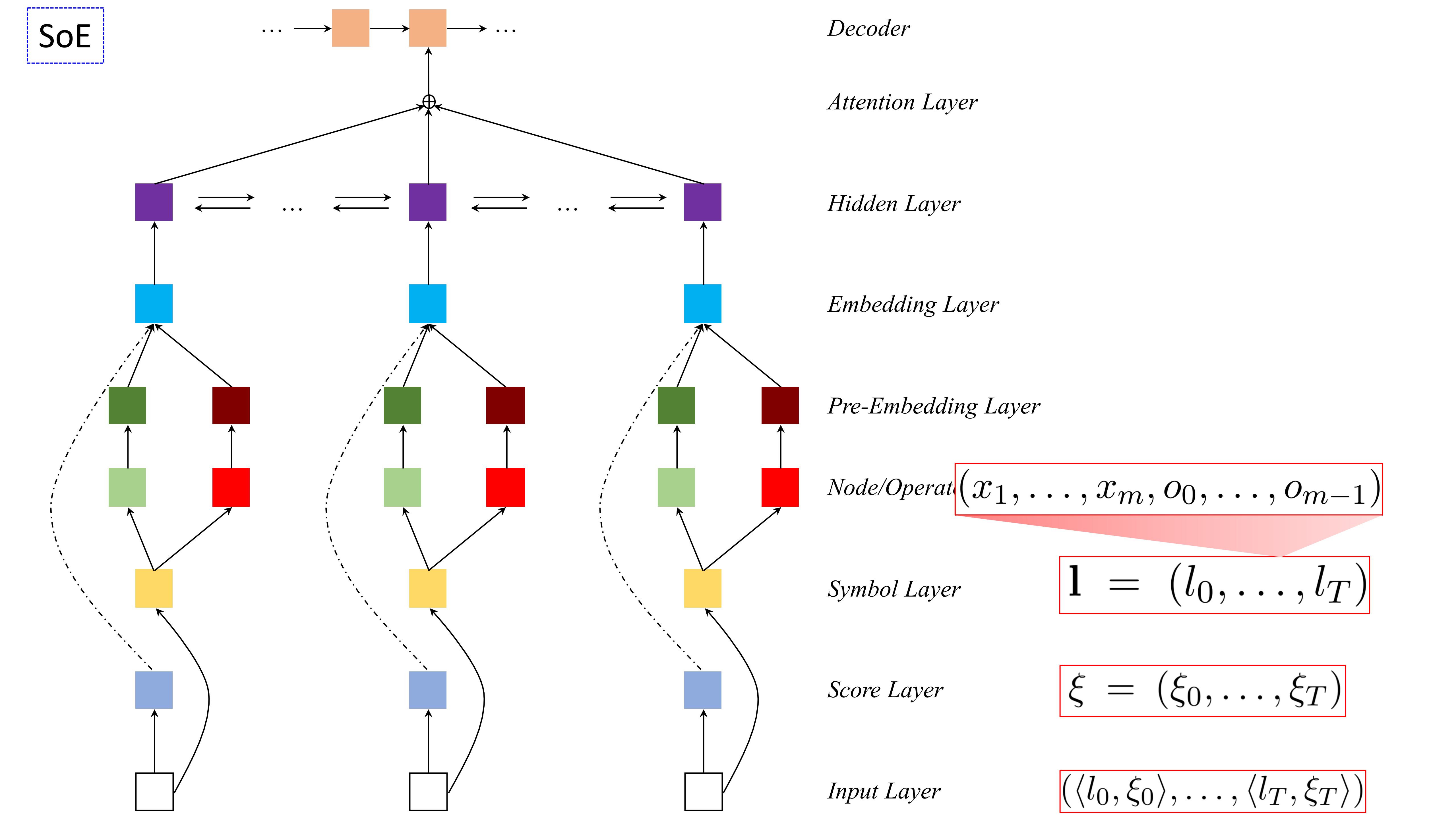

SoE
Decoder
Attention Layer
Hidden Layer
Embedding Layer
Pre-Embedding Layer
Node/Operator Layer
Symbol Layer
NNP⊗John / NP⊗John / ©NNP⊗John /
1.0 / 0.8 / 0.9 /
Input Layer
$(\langle l_0, \xi_0\rangle, \ldots, \langle l_T, \xi_T\rangle)$

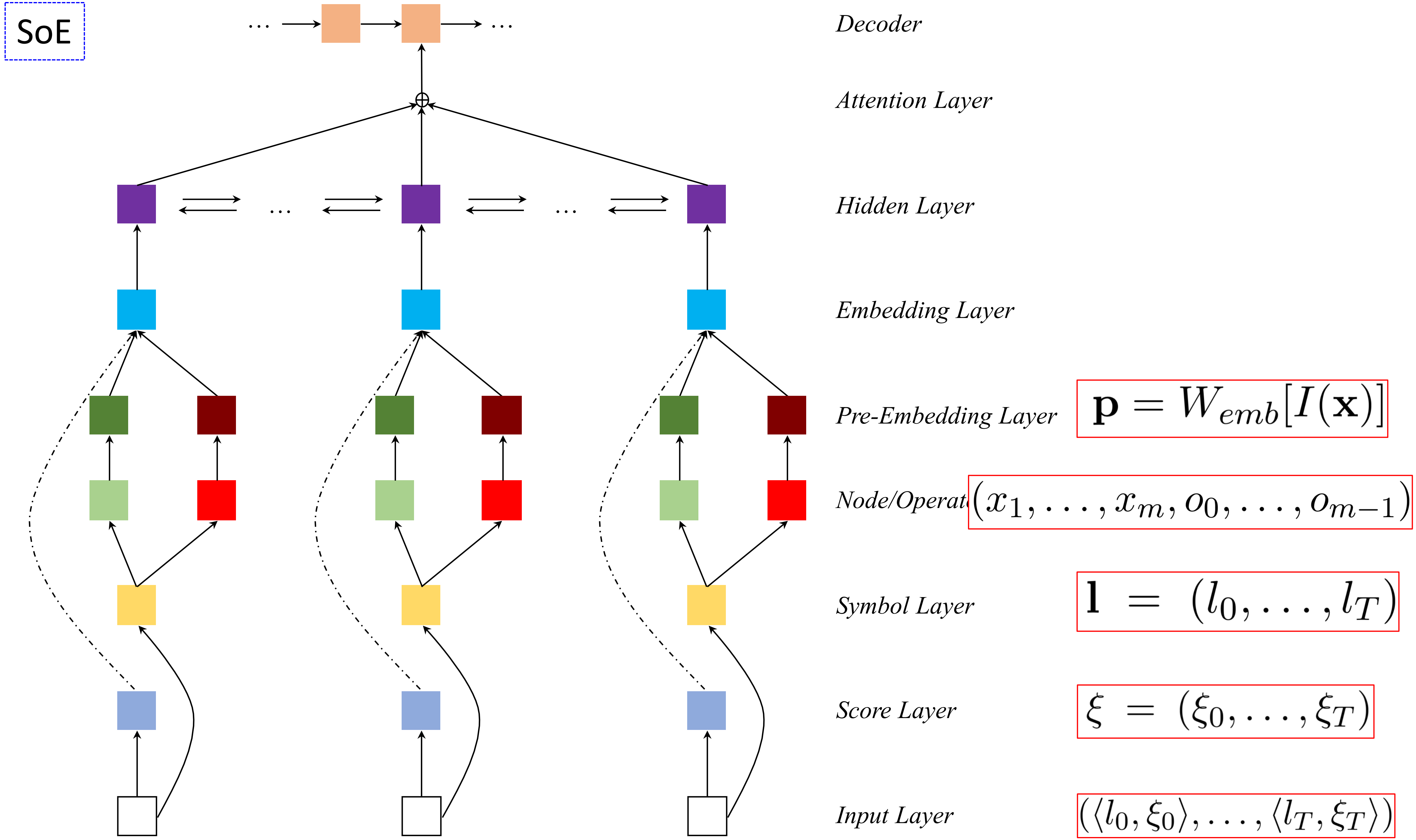


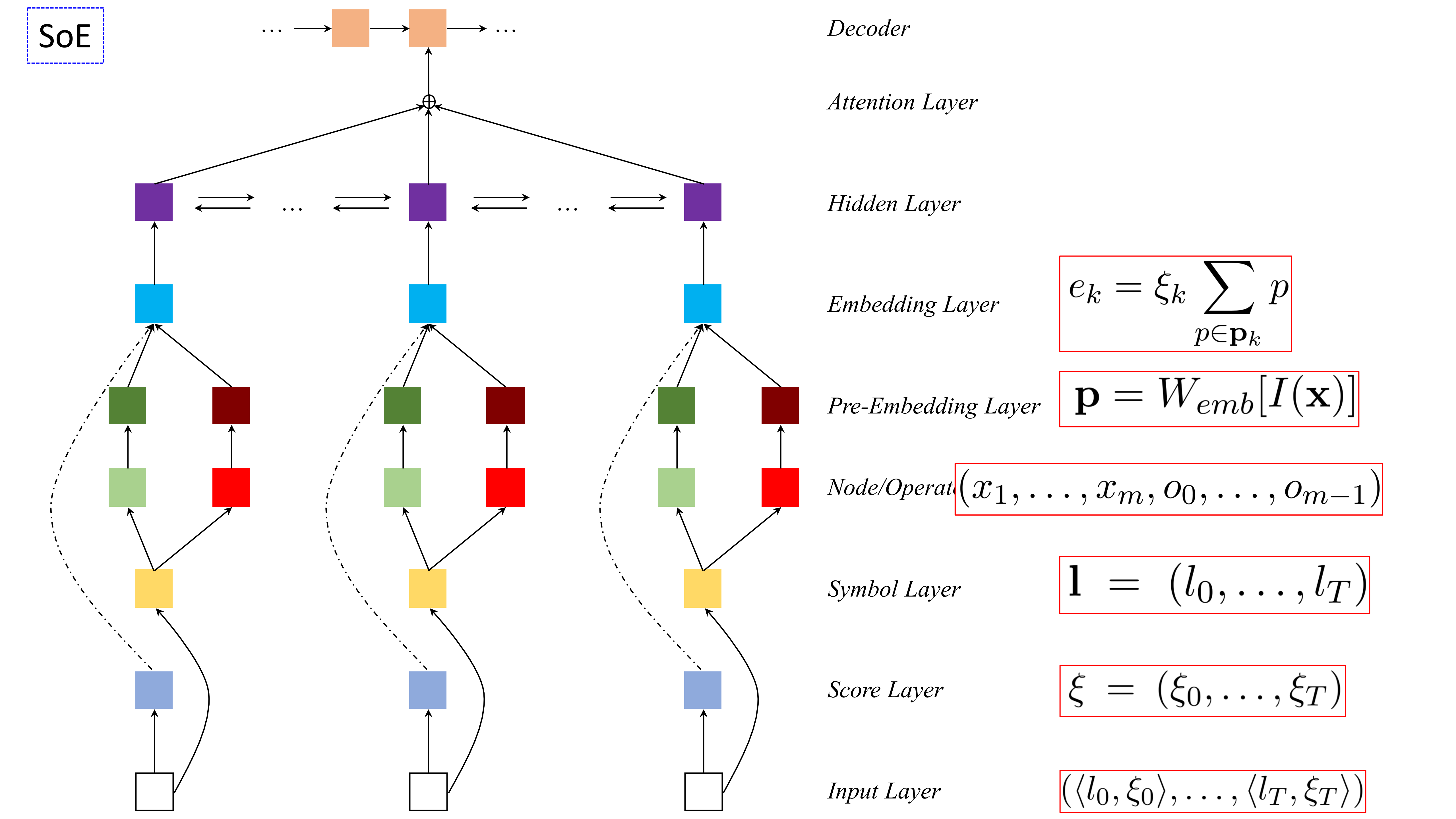
SoE
Decoder
Attention Layer
Hidden Layer
Embedding Layer
Pre-Embedding Layer
Node/Operator Layer
NNP⊗John / NP⊗John / ©NNP⊗John /
Layer
1.0 / 0.8 / 0.9 /
Layer
$\mathbf{l} = (l_0, \ldots, l_T)$
$\xi = (\xi_0, \ldots, \xi_T)$
Input Layer
$(\langle l_0, \xi_0 \rangle, \ldots, \langle l_T, \xi_T \rangle)$

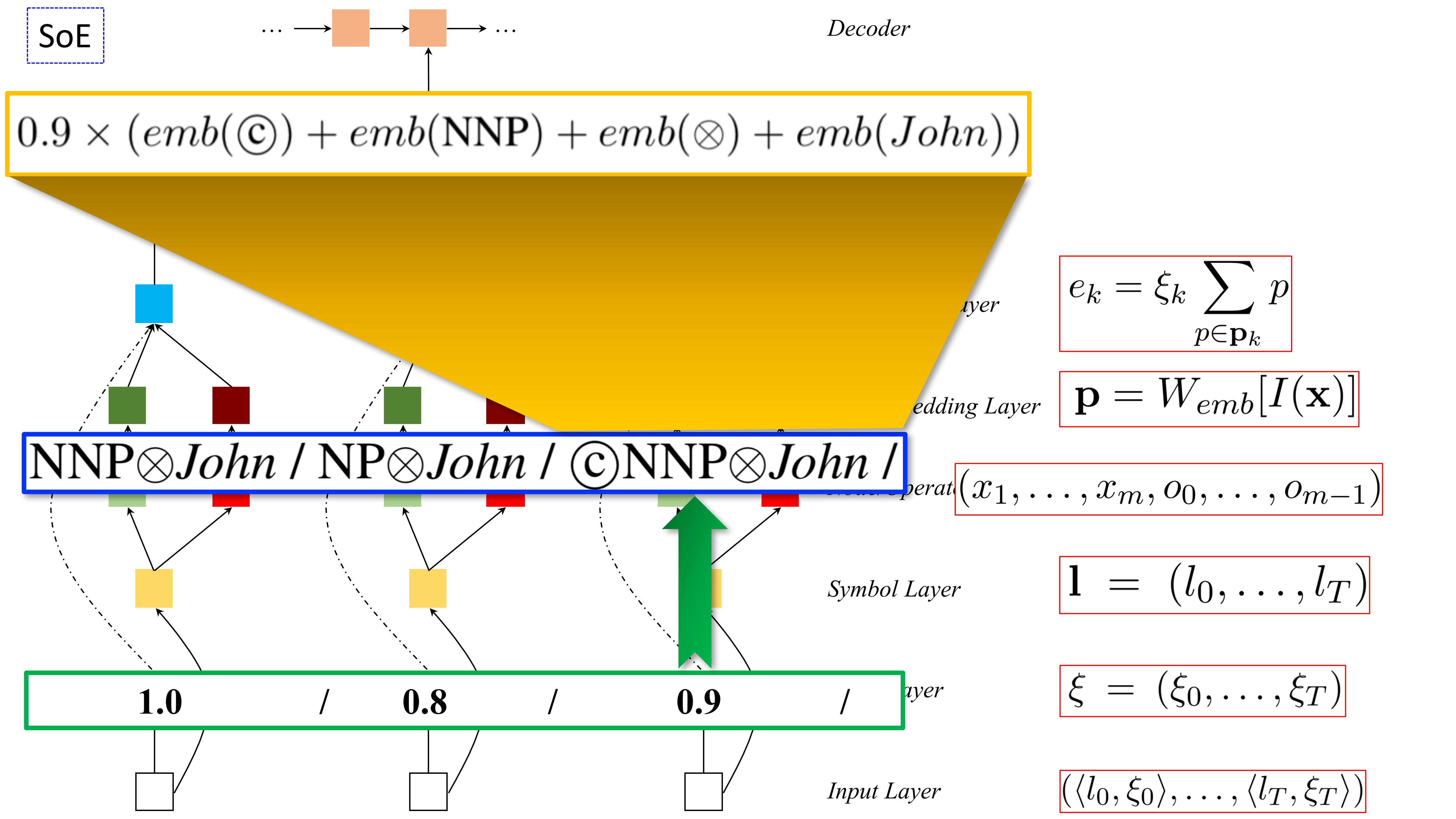


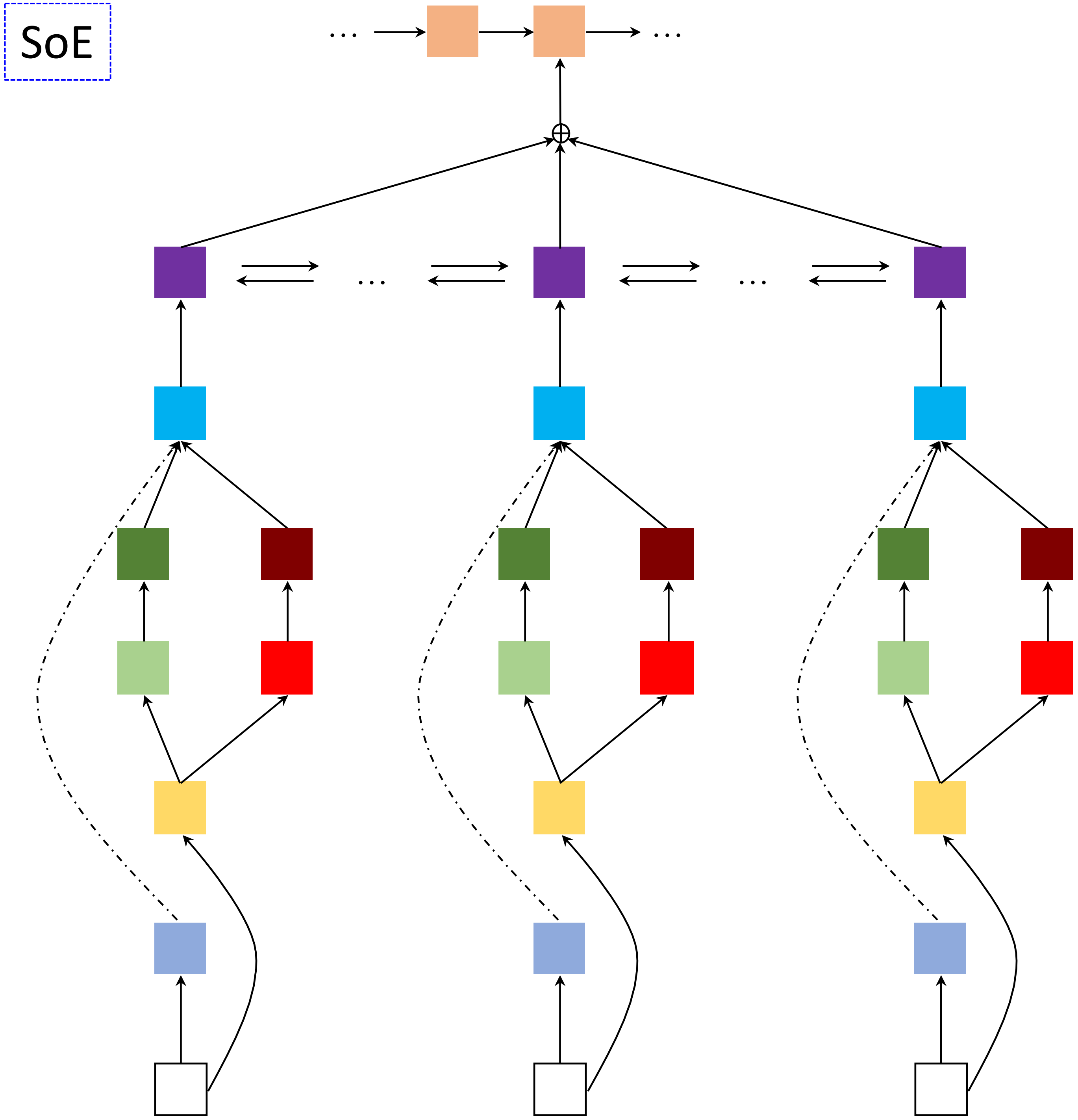
SoE
Decoder
Attention Layer
Hidden Layer
$l = o_0 x_1 o_1 \ldots x_{m-1} o_{m-1} x_m$
g Layer
rator Layer
NNP⊗John / NP⊗John / ⓒNNP⊗John /
Layer
Score Layer
Input Layer
$\mathbf{l} = (l_0, \ldots, l_T)$
$\xi = (\xi_0, \ldots, \xi_T)$
$(\langle l_0, \xi_0 \rangle, \ldots, \langle l_T, \xi_T \rangle)$


SoE
Decoder
Attention Layer
Hidden Layer
Embedding Layer
Pre-Embedding Layer
Node/Operat
$(x_1, \ldots, x_m, o_0, \ldots, o_{m-1})$
Symbol Layer
$\mathbf{l} = (l_0, \ldots, l_T)$
Score Layer
$\xi = (\xi_0, \ldots, \xi_T)$
Input Layer
$(\langle l_0, \xi_0 \rangle, \ldots, \langle l_T, \xi_T \rangle)$


SoE
Decoder
Attention Layer
Hidden Layer
Embedding Layer
Pre-Embedding Layer
$\mathbf{p} = W_{emb}[I(\mathbf{x})]$
Node/Operat
$(x_1, \ldots, x_m, o_0, \ldots, o_{m-1})$
Symbol Layer
$\mathbf{l} = (l_0, \ldots, l_T)$
Score Layer
$\xi = (\xi_0, \ldots, \xi_T)$
Input Layer
$(\langle l_0, \xi_0 \rangle, \ldots, \langle l_T, \xi_T \rangle)$


SoE
…
…
Decoder
Attention Layer
Hidden Layer
…
…
Embedding Layer
$e_k = \xi_k \sum_{p \in \mathbf{p}_k} p$
Pre-Embedding Layer
$\mathbf{p} = W_{emb}[I(\mathbf{x})]$
Node/Operat
$(x_1, \ldots, x_m, o_0, \ldots, o_{m-1})$
Symbol Layer
$\mathbf{l} \ = \ (l_0, \ldots, l_T)$
Score Layer
$\xi \ = \ (\xi_0, \ldots, \xi_T)$
Input Layer
$(\langle l_0, \xi_0 \rangle, \ldots, \langle l_T, \xi_T \rangle)$


SoE
Decoder
$0.9 \times (emb(©) + emb(\mathrm{NNP}) + emb(\otimes) + emb(John))$
$e_k = \xi_k \sum_{p \in \mathbf{p}_k} p$
edding Layer
$\mathbf{p} = W_{emb}[I(\mathbf{x})]$
NNP⊗*John* / NP⊗*John* / ©NNP⊗*John* /
$(x_1, \ldots, x_m, o_0, \ldots, o_{m-1})$
Symbol Layer
$\mathbf{l} = (l_0, \ldots, l_T)$
1.0 / 0.8 / 0.9 /
$\xi = (\xi_0, \ldots, \xi_T)$
Input Layer
$(\langle l_0, \xi_0 \rangle, \ldots, \langle l_T, \xi_T \rangle)$



Decoder

Attention Layer $\alpha_{ij} = \frac{\exp(a(s_{i-1}, h_j))}{\sum_{k=0}^{T} \exp(a(s_{i-1}, h_k))}$

Hidden Layer

Embedding Layer $e_k = \xi_k \sum_{p \in \mathbf{p}_k} p$

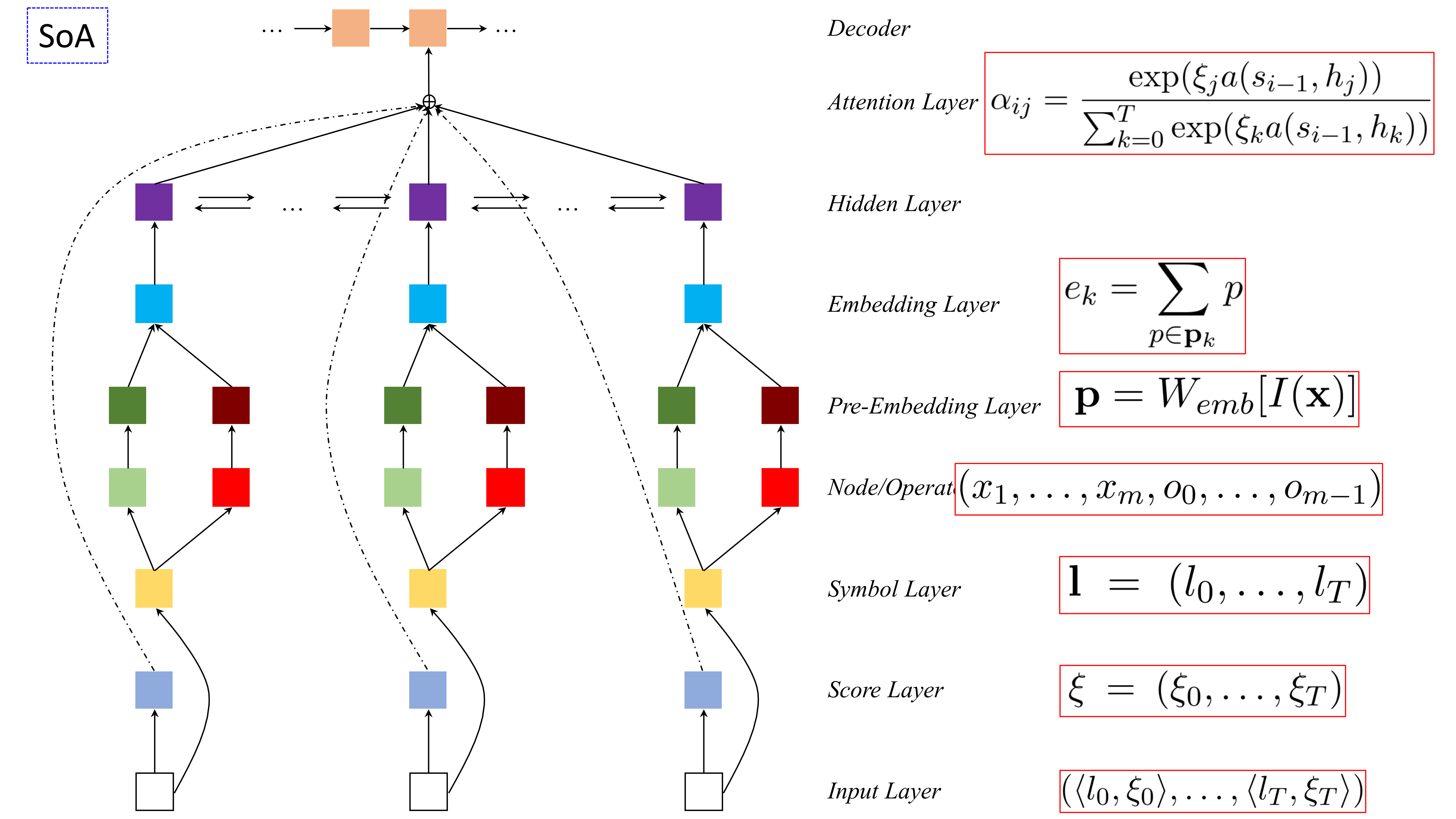
Pre-Embedding Layer $\mathbf{p} = W_{emb}[I(\mathbf{x})]$

Node/Operat $(x_1, \ldots, x_m, o_0, \ldots, o_{m-1})$

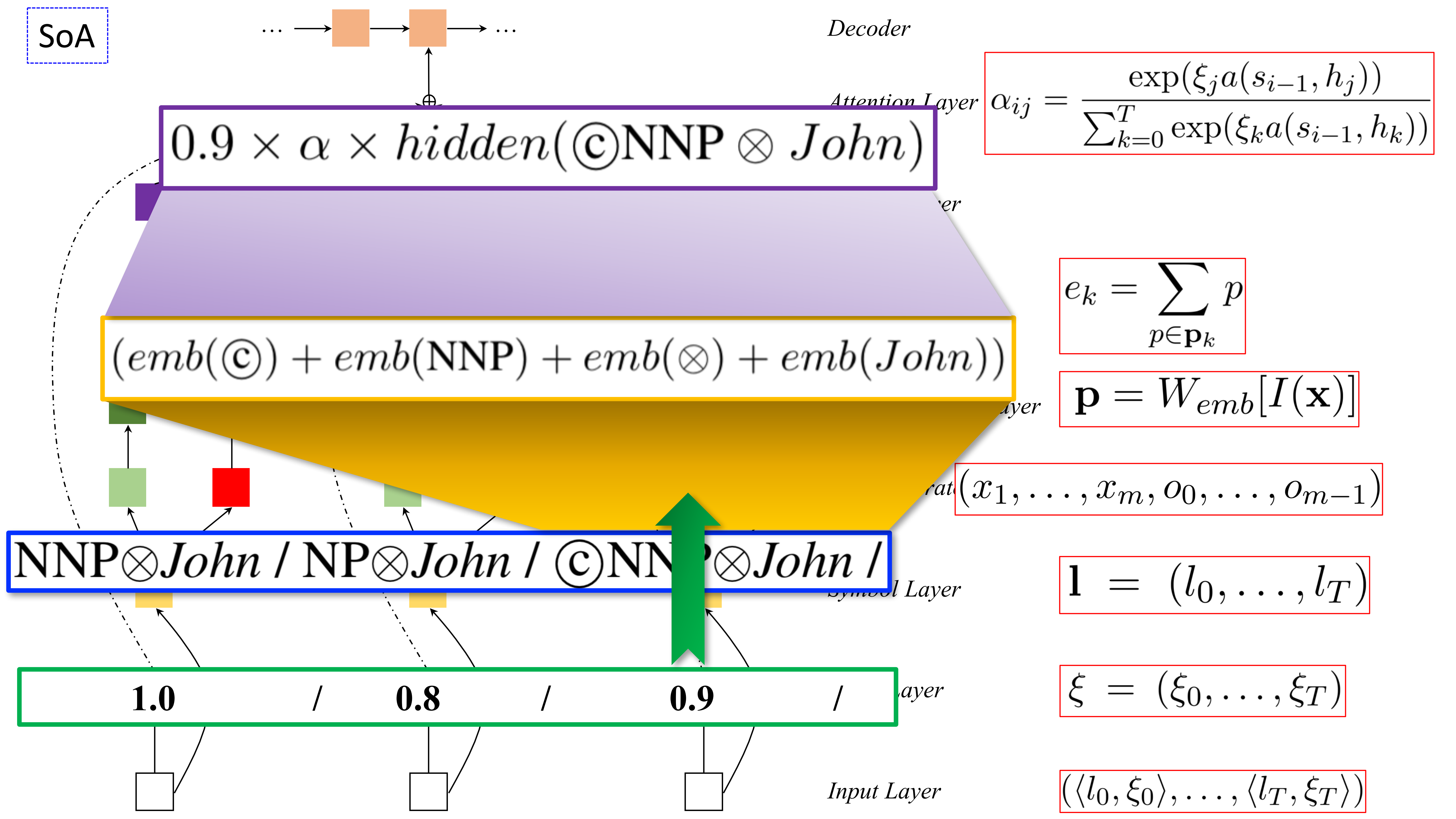
Symbol Layer $\mathbf{l} = (l_0, \ldots, l_T)$

Score Layer $\xi = (\xi_0, \ldots, \xi_T)$

Input Layer $(\langle l_0, \xi_0 \rangle, \ldots, \langle l_T, \xi_T \rangle)$



SoA
... ...
Decoder
Attention Layer
$\alpha_{ij} = \frac{\exp(\xi_j a(s_{i-1}, h_j))}{\sum_{k=0}^{T} \exp(\xi_k a(s_{i-1}, h_k))}$
Hidden Layer
... ...
Embedding Layer
$e_k = \sum_{p \in \mathbf{p}_k} p$
Pre-Embedding Layer
$\mathbf{p} = W_{emb}[I(\mathbf{x})]$
Node/Operat
$(x_1, \ldots, x_m, o_0, \ldots, o_{m-1})$
Symbol Layer
$\mathbf{l} = (l_0, \ldots, l_T)$
Score Layer
$\xi = (\xi_0, \ldots, \xi_T)$
Input Layer
$(\langle l_0, \xi_0 \rangle, \ldots, \langle l_T, \xi_T \rangle)$



SoA
Decoder
Attention Layer
$\alpha_{ij} = \frac{\exp(\xi_j a(s_{i-1}, h_j))}{\sum_{k=0}^{T} \exp(\xi_k a(s_{i-1}, h_k))}$
$0.9 \times \alpha \times hidden(©NNP \otimes John)$
$(emb(©) + emb(NNP) + emb(\otimes) + emb(John))$
$e_k = \sum_{p \in \mathbf{p}_k} p$
$\mathbf{p} = W_{emb}[I(\mathbf{x})]$
$(x_1, \ldots, x_m, o_0, \ldots, o_{m-1})$
NNP⊗John / NP⊗John / ©NNP⊗John /
Symbol Layer
$\mathbf{l} = (l_0, \ldots, l_T)$
1.0 / 0.8 / 0.9 /
$\xi = (\xi_0, \ldots, \xi_T)$
Input Layer
$(\langle l_0, \xi_0 \rangle, \ldots, \langle l_T, \xi_T \rangle)$

Experiments

Data

Language	Corpus	Usage	#Sent.
English-Japanese	ASPEC	train	100,000
		dev.	1790
		test	1812
English-Chinese	LDC	train	1,423,695
	FBIS		233,510
	NIST MT 02	dev.	876
	NIST MT 03	test	919
	NIST MT 04		1,788
	NIST MT 05		1,082

System Types		Systems & Configurations	MT 03 FBIS	MT 03 LDC	MT 04 FBIS	MT 04 LDC	MT 05 FBIS	MT 05 LDC
Previous	FS	Mi et al. (2008)	27.10	28.21	28.67	30.09	26.57	28.36
	TN	Eriguchi et al. (2016)	29.00	29.71	30.24	31.56	28.38	30.33
		Chen et al. (2017)	28.34	29.64	30.00	31.25	28.14	29.59
		Li et al. (2017)	28.40	29.60	29.66	31.96	27.74	29.84
Ours	SN	s2s	27.44	29.18	29.73	30.53	27.32	28.80
	TN	1-best (No score)	28.61	29.38	30.07	31.58	28.59	30.01
		1-best (SoE)	28.78	30.65	30.36	32.22	29.31	30.16
		1-best (SoA)	29.39	30.80	30.25	32.39	29.30	30.61
	FN	Forest (No score)	28.06	29.63	29.51	31.41	28.48	29.75
		Forest (SoE)	29.58	31.07	**30.67**	32.69	29.26	30.41
		Forest (SoA)	**29.63**	**31.35**	30.31	**33.14**	**29.87**	**31.23**

- s2s is the worst
  - Syntactic information is useful

English-Chinese

System Types		Systems & Configurations	MT 03 FBIS	MT 03 LDC	MT 04 FBIS	MT 04 LDC	MT 05 FBIS	MT 05 LDC
Previous	FS	Mi et al. (2008)	27.10	28.21	28.67	30.09	26.57	28.36
	TN	Eriguchi et al. (2016)	29.00	29.71	30.24	31.56	28.38	30.33
		Chen et al. (2017)	28.34	29.64	30.00	31.25	28.14	29.59
		Li et al. (2017)	28.40	29.60	29.66	31.96	27.74	29.84
Ours	SN	s2s	27.44	29.18	29.73	30.53	27.32	28.80
	TN	1-best (No score)	28.61	29.38	30.07	31.58	28.59	30.01
		1-best (SoE)	28.78	30.65	30.36	32.22	29.31	30.16
		1-best (SoA)	29.39	30.80	30.25	32.39	29.30	30.61
	FN	Forest (No score)	28.06	29.63	29.51	31.41	28.48	29.75
		Forest (SoE)	29.58	31.07	**30.67**	32.69	29.26	30.41
		Forest (SoA)	**29.63**	**31.35**	30.31	**33.14**	**29.87**	**31.23**

- s2s is the worst
  - Syntactic information is useful
- No score is the worst
  - Score is useful

English-Chinese

System Types		Systems & Configurations	MT 03		MT 04		MT 05	
			FBIS	LDC	FBIS	LDC	FBIS	LDC
Previous	FS	Mi et al. (2008)	27.10	28.21	28.67	30.09	26.57	28.36
	TN	Eriguchi et al. (2016)	29.00	29.71	30.24	31.56	28.38	30.33
		Chen et al. (2017)	28.34	29.64	30.00	31.25	28.14	29.59
		Li et al. (2017)	28.40	29.60	29.66	31.96	27.74	29.84
Ours	SN	s2s	27.44	29.18	29.73	30.53	27.32	28.80
	TN	1-best (No score)	28.61	29.38	30.07	31.58	28.59	30.01
		1-best (SoE)	28.78	30.65	30.36	32.22	29.31	30.16
		1-best (SoA)	29.39	30.80	30.25	32.39	29.30	30.61
	FN	Forest (No score)	28.06	29.63	29.51	31.41	28.48	29.75
		Forest (SoE)	29.58	31.07	**30.67**	32.69	29.26	30.41
		Forest (SoA)	**29.63**	**31.35**	30.31	**33.14**	**29.87**	**31.23**

- SoA is better than SoE
  - Adjusting attention is better than adjusting word embedding

English-Chinese

System Types		Systems & Configurations	MT 03		MT 04		MT 05	
			FBIS	LDC	FBIS	LDC	FBIS	LDC
Previous	FS	Mi et al. (2008)	27.10	28.21	28.67	30.09	26.57	28.36
	TN	Eriguchi et al. (2016)	29.00	29.71	30.24	31.56	28.38	30.33
		Chen et al. (2017)	28.34	29.64	30.00	31.25	28.14	29.59
		Li et al. (2017)	28.40	29.60	29.66	31.96	27.74	29.84
Ours	SN	s2s	27.44	29.18	29.73	30.53	27.32	28.80
	TN	1-best (No score)	28.61	29.38	30.07	31.58	28.59	30.01
		1-best (SoE)	28.78	30.65	30.36	32.22	29.31	30.16
		1-best (SoA)	29.39	30.80	30.25	32.39	29.30	30.61
	FN	Forest (No score)	28.06	29.63	29.51	31.41	28.48	29.75
		Forest (SoE)	29.58	31.07	**30.67**	32.69	29.26	30.41
		Forest (SoA)	**29.63**	**31.35**	30.31	**33.14**	**29.87**	**31.23**

- SoA is better than SoE
  - Adjusting attention is better than adjusting word embedding
- Forest is better than 1-best
  - More syntactic information is useful

English-Chinese

System Types		Systems & Configurations	MT 03		MT 04		MT 05	
			FBIS	LDC	FBIS	LDC	FBIS	LDC
Previous	FS	Mi et al. (2008)	27.10	28.21	28.67	30.09	26.57	28.36
	TN	Eriguchi et al. (2016)	29.00	29.71	30.24	31.56	28.38	30.33
		Chen et al. (2017)	28.34	29.64	30.00	31.25	28.14	29.59
		Li et al. (2017)	28.40	29.60	29.66	31.96	27.74	29.84
Ours	SN	s2s	27.44	29.18	29.73	30.53	27.32	28.80
	TN	1-best (No score)	28.61	29.38	30.07	31.58	28.59	30.01
		1-best (SoE)	28.78	30.65	30.36	32.22	29.31	30.16
		1-best (SoA)	29.39	30.80	30.25	32.39	29.30	30.61
	FN	Forest (No score)	28.06	29.63	29.51	31.41	28.48	29.75
		Forest (SoE)	29.58	31.07	**30.67**	32.69	29.26	30.41
		Forest (SoA)	**29.63**	**31.35**	30.31	**33.14**	**29.87**	**31.23**

- Forest (No score) is worse than 1-best (SoE/SoA)
  - Influence of noise

English-Chinese

System Types		Systems & Configurations	MT 03		MT 04		MT 05	
			FBIS	LDC	FBIS	LDC	FBIS	LDC
Previous	FS	Mi et al. (2008)	27.10	28.21	28.67	30.09	26.57	28.36
	TN	Eriguchi et al. (2016)	29.00	29.71	30.24	31.56	28.38	30.33
		Chen et al. (2017)	28.34	29.64	30.00	31.25	28.14	29.59
		Li et al. (2017)	28.40	29.60	29.66	31.96	27.74	29.84
Ours	SN	s2s	27.44	29.18	29.73	30.53	27.32	28.80
	TN	1-best (No score)	28.61	29.38	30.07	31.58	28.59	30.01
		1-best (SoE)	28.78	30.65	30.36	32.22	29.31	30.16
		1-best (SoA)	29.39	30.80	30.25	32.39	29.30	30.61
	FN	Forest (No score)	28.06	29.63	29.51	31.41	28.48	29.75
		Forest (SoE)	29.58	31.07	**30.67**	32.69	29.26	30.41
		Forest (SoA)	**29.63**	**31.35**	30.31	**33.14**	**29.87**	**31.23**

- Forest (No score) is worse than 1-best (SoE/SoA)
  - Influence of noise
- FS/TN is worse than 1-best (SoE/SoA)
  - Better to use score in linearization

English-Chinese

English-Japanese

System Types		Systems & Configurations	BLEU (test)
Previous	FS	Mi et al. (2008)	34.13
	TN	Eriguchi et al. (2016)	37.52
		Chen et al. (2017)	36.94
		Li et al. (2017)	36.21
Ours	SN	s2s	37.10
	TN	1-best (No score)	38.01
		1-best (SoE)	38.53
		1-best (SoA)	39.42
	FN	Forest (No score)	37.92
		Forest (SoE)	41.35
		Forest (SoA)	**42.17**

- s2s is the worst
- No score is the worst
- SoA is better than SoE
- Forest is better than 1-best
- Forest (No score) is worse than 1-best (SoE/SoA)
- FS/TN is worse than 1-best (SoE/SoA)

Merits & Demerits

- Use syntactic information explicitly
- Simpler model, more information
- Robust to parsing errors

- Lots of sentences are filtered out due to lengths
- Memory consumption
- Training/decoding efficiency
- Implementation tricks

Conclusion

First attempt to use forest in neural machine translation

Thanks

Q & A