AIエンジニアリング　(9)

Recursive Neural Network

時系列処理

- RNN(Recurrent, Recursive) NNは主に時系列処理に用いられる手法
- 時系列：音声、テキスト、株価などの時間的な変化を表すデータ系列
- 例えば音声認識や機械翻訳などが該当する
- 特に自然言語処理での研究は非常に活発

Recurrent & Recursive NN

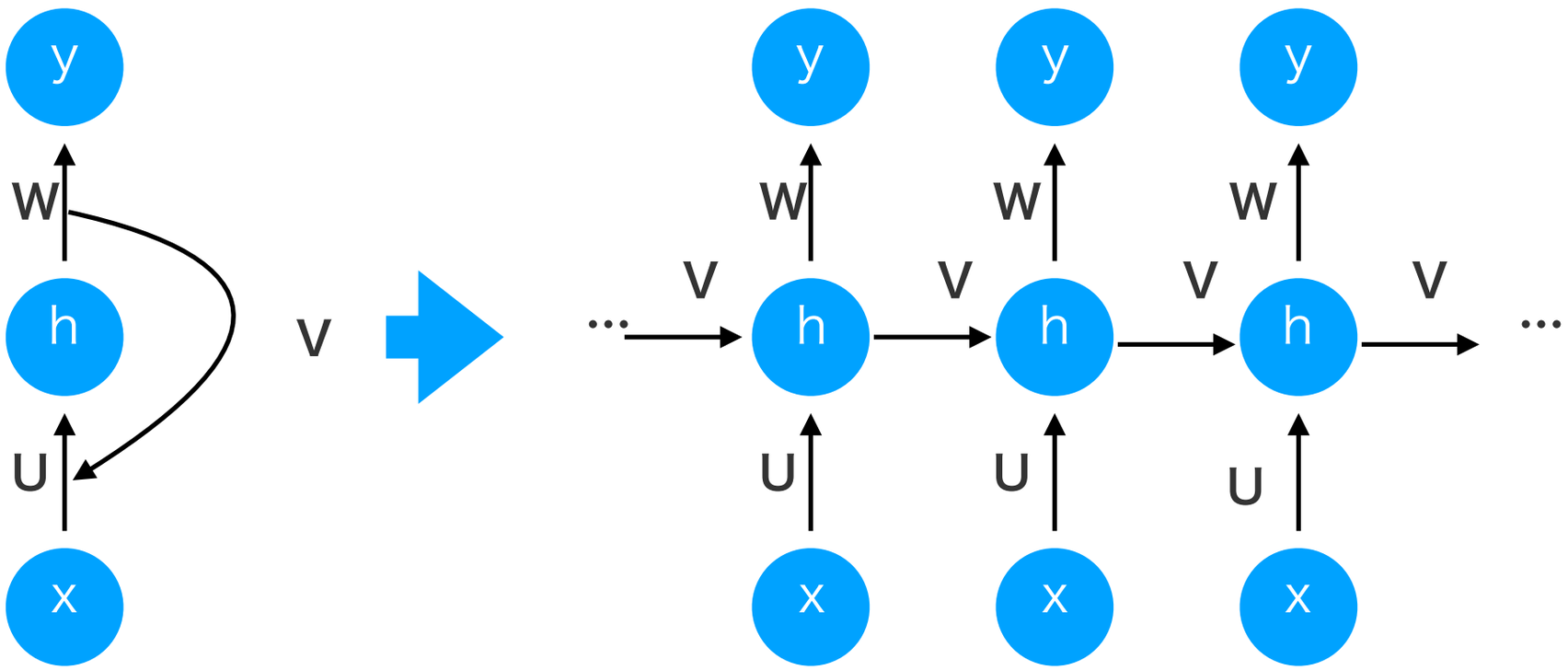
RNN

- 時系列データを扱うためには今までの情報を保持しなければならない
- データ保持のためにはネットワークの何処かにループが必要になる
- しかしループがあるとBackpropできないように思える

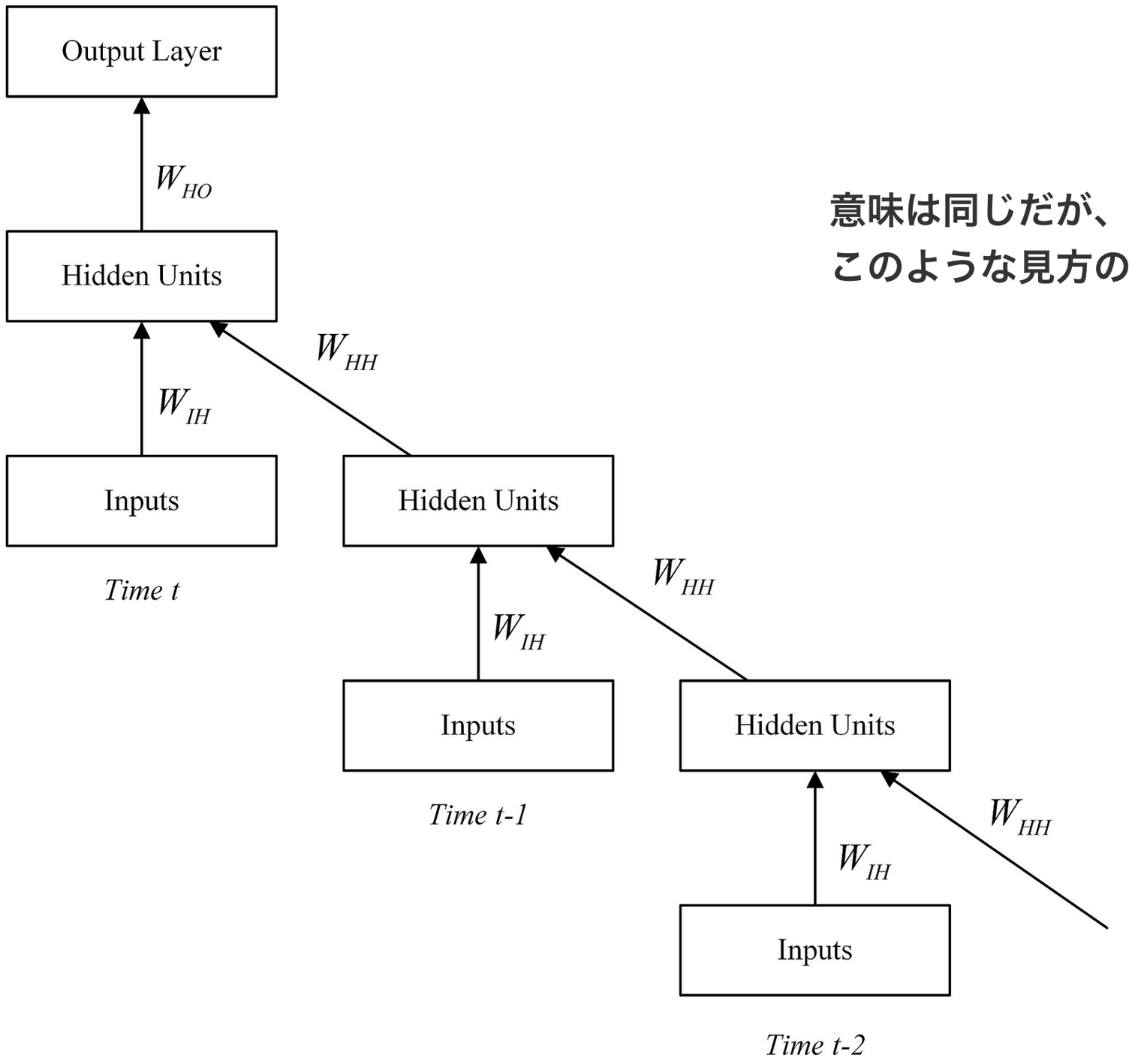
Back Propagation Through Time (BPTT)

- 時系列データを扱うためには今までの情報を保持しなければならない
- データ保持のためにはネットワークの何処かにループが必要になる
- しかしループがあるとBackpropできないように思える
- BPTTという方法を使うと普通にBackpropできる

BPTT

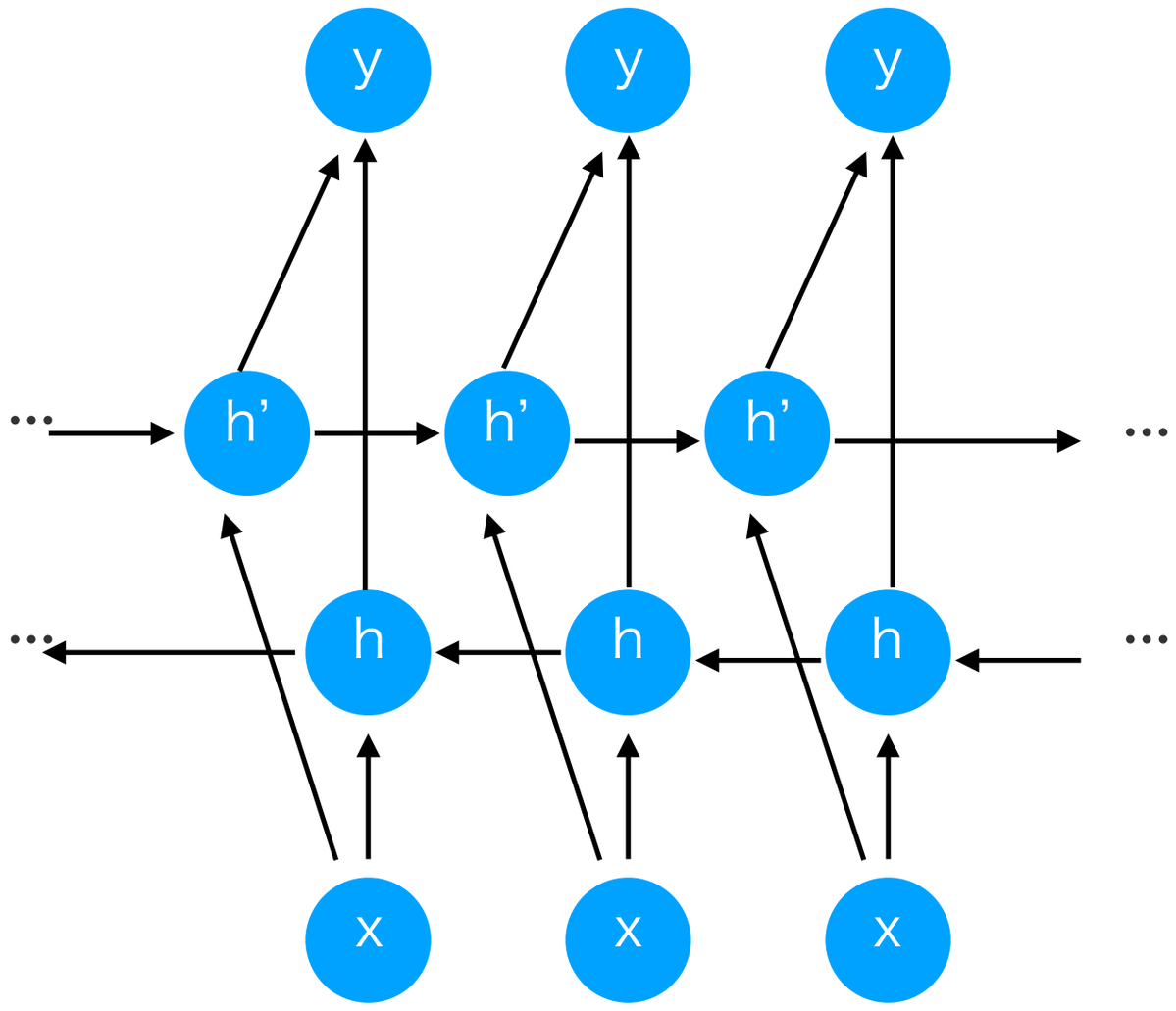


右のように展開して巨大なNNだと考えれば
ループがなくなるので普通にBackpropできる



意味は同じだが、
このような見方のほうがわかりやすいかも

Bi-directional RNN



難しく見えるが、
過去だけではなく
未来からの情報も通す
NN

過去→未来のRNNと、
未来→過去のRNNの2つ
を
用意して、両方から出力
を予測するネットワーク
ともみなせる

長期依存性の問題

- 長い系列を学習しようとするとネットワークが大きくなりすぎる
- 最初の方の情報が次第に上書きされていって消えてしまう
- 勾配が消える or 発散する
- ネットワークが大きくなりすぎる問題についてはデータをある程度の長さで打ち切ればいいが、情報が消えてしまうのはどうしようもない

長期依存性の問題

- 単純化して考えてみよう
- 保持したい情報があったとして、1より小さい値を何度も掛けていくと0になってしまう(消える)
- 逆に1より大きい値を掛けていくと発散してしまう
- 無理やり1付近に留めようとすると、今度はネットワークの表現能力が低下してしまう

長期依存性の問題
スペクトル半径

- 重み行列Wのスペクトル半径ρ(W)を考える
  - 行列の固有値の絶対値の中で、一番大きいもの
- 実はこれがRNNの重みがうまくいくかと関係がある
  - ρ(W) < 1 だと出力が 0 になる (指数的に減少する)
  - ρ(W) > 1 だと出力が ∞ になる (指数的に増大する)
- 1に近いほど安定する(ちょうど1なら変わらない)

長期依存性の問題
スペクトル半径

$W=$

1	-1
2	-2

$\rho(W) = 1$

$W^{10}=$

-1	1
-2	2

←10回掛けても（ノルムは）
大きくも小さくもならない

長期依存性の問題
スペクトル半径

$$W=\begin{pmatrix} 1 & 0.5 \\ -0.5 & -1 \end{pmatrix}$$

$\rho(W) = 0.866$

(まあまあ1に近い)

$$W^{10}=\begin{pmatrix} 0.24 & 0 \\ 0 & 0.24 \end{pmatrix}$$

←10回掛けても1/4くらい
=ゆっくりと減衰

長期依存性の問題
スペクトル半径

$$W=\begin{pmatrix} -2 & -0.1 \\ -0.3 & 1 \end{pmatrix}$$

$$\rho(W) = 2.01$$

$$W^{10}=\begin{pmatrix} 1072 & 36 \\ 107 & 5 \end{pmatrix}$$

←10回掛けただけで500倍くらい
=すぐ∞に発散

長期依存性の問題スペクトル半径

- というわけで、一応スペクトル半径をうまく抑えることができれば爆発も消失もしなさそうである
- とはいえスペクトル半径を常に1(付近)に押さえておくのはネットワーク全体の表現能力が落ちる
- 実際、勾配の発散・消失こそ起きないものの、RNNが安定して長期間記憶を保持してくれないことがわかっている

  →Learning Long-Term Dependencies with Gradient Descent is Difficult (Bengio, 1993)

長期依存性の問題

- つまり、普通のRNNは長期依存を考慮できない
- 簡単に言えば、大昔の情報は忘れてしまう
- 覚えておかないといけない情報と、そうでもない情報を取捨選択できるような仕組みが必要
- ちなみに「スペクトル半径を1付近にすることで挙動を安定させる」方針をとっている手法もある(ESN)

これまでの手法の問題

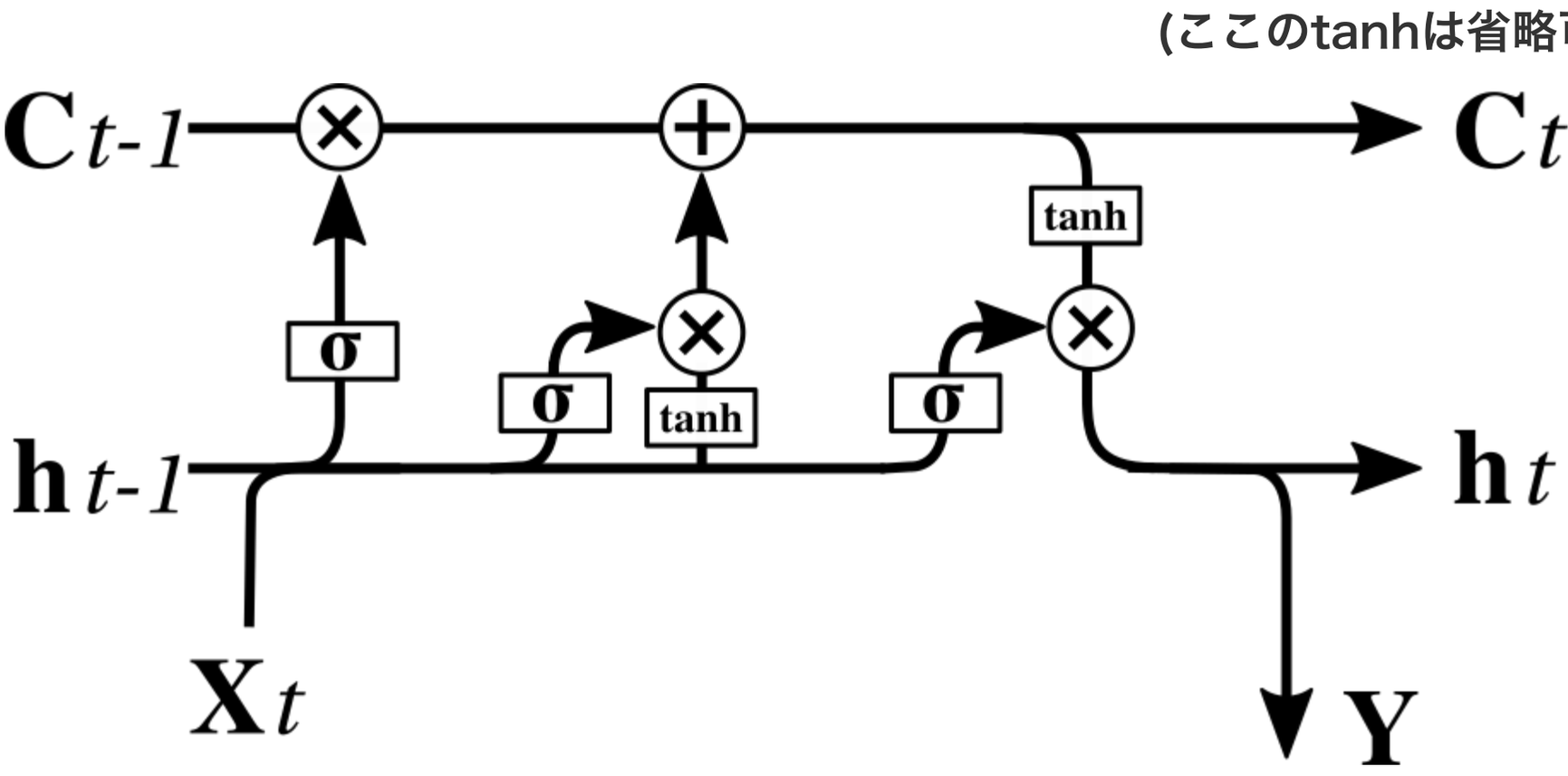
- もう使わなくなった情報は忘れたい、ということもある
- 覚えるべき情報はずっと記憶したい
- 使わなくなった情報は積極的に消去したい
- 記憶するか忘却するかの切り替え機構があるネットワークが必要

Long-Short Term Memory (LSTM)

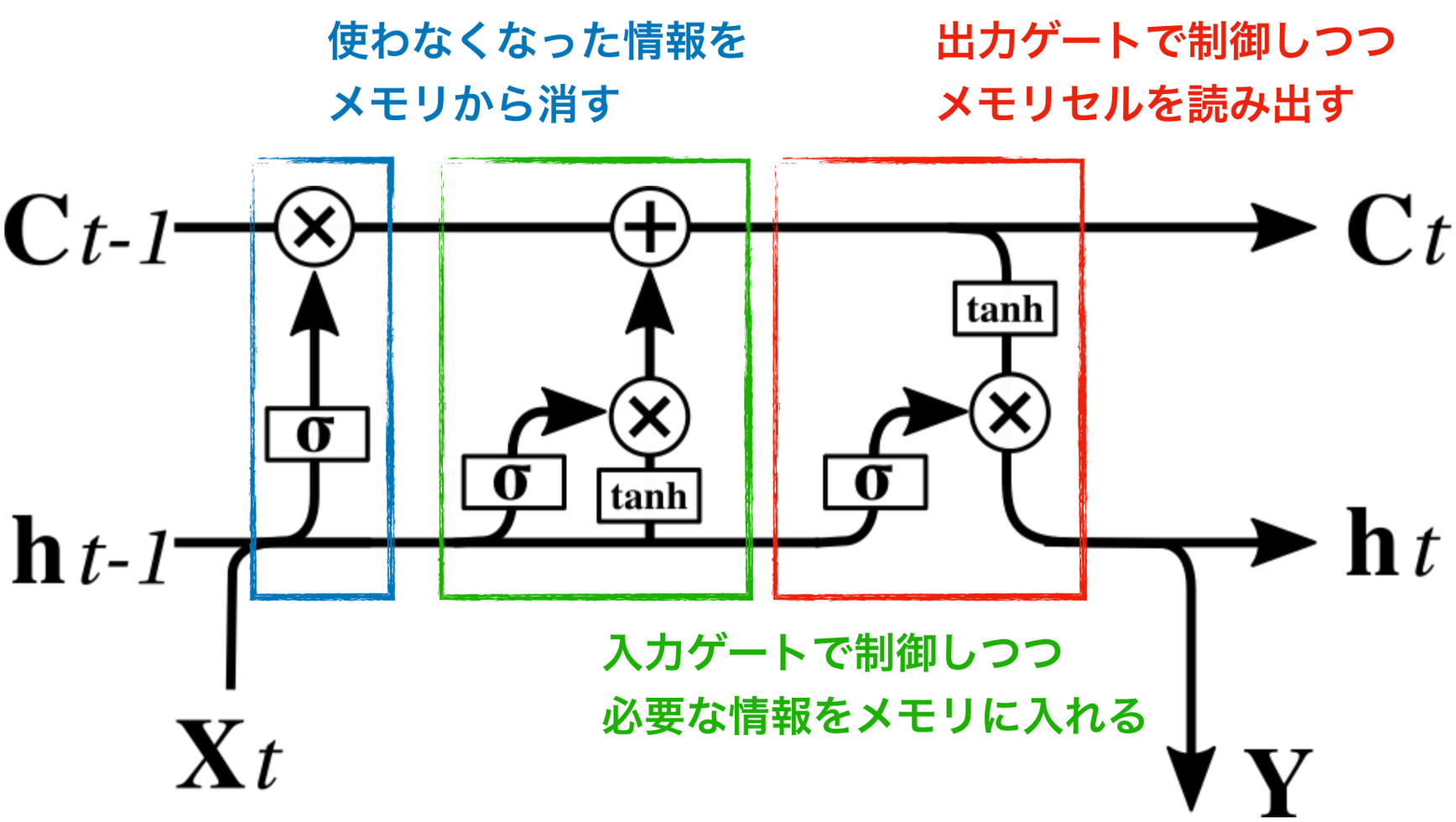
LSTMの前に

- LSTMを含めこれから紹介する方法は複雑なものも多いが、基本的には「もっとシンプルなネットワークでも**理屈上は**学習可能」である (→万能近似定理参照)
- ただ普通のMLPやRNNを学習しようとしても良い解に収束させるのは難しい
- そこで「こうやればもっと学習が楽になるはず」と手を加えてあげる事を考えて、それがResNetだったりLSTMやその他いろいろな方法である
- LSTMも複雑だが「こうしないと学習不可能」というわけではないことに注意

LSTM



LSTM



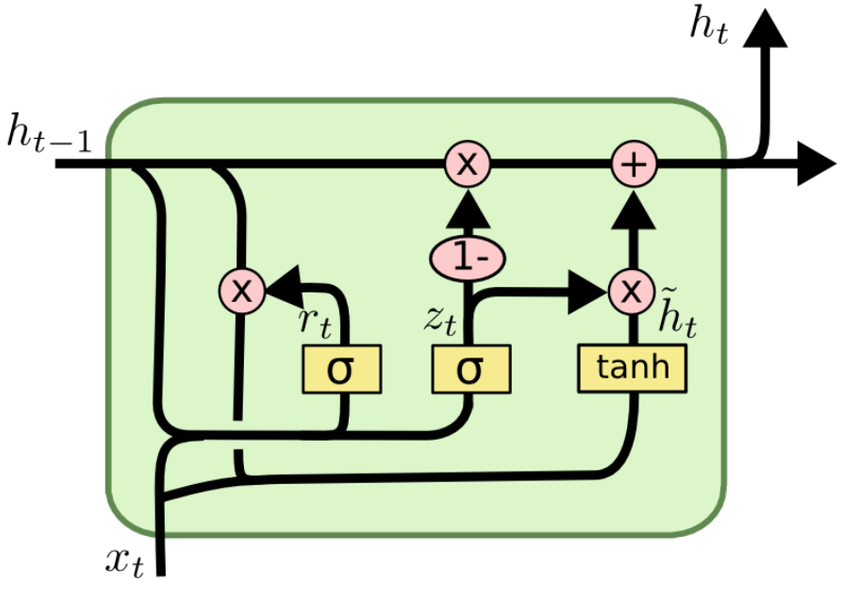
忘却ゲート、入力ゲート、出力ゲートの3つのSigmoidがある
2つのtanhはどれくらいメモリに書き込むかと、どれくらいメモリから読み込むか

GRU(Gated Recurrent Unit)

- LSTMは歴史的経緯からいろいろなパーツを追加する形で進歩してきた
- 思い切って簡略化したモデルも存在し、代表例がGRU
- メモリセルをなくして、忘却ゲートと入力ゲートを合体(更新ゲート)したモデル

GRU(Gated Recurrent Unit)

- LSTMは歴史的経緯からいろいろなパーツを追加する形で進歩してきた
- 思い切って簡略化したモデルも存在し、代表例がGRU
- メモリセルをなくして、忘却ゲートと入力ゲートを合体(更新ゲート)したモデル



$$z_t = \sigma\left(W_z \cdot [h_{t-1}, x_t]\right)$$

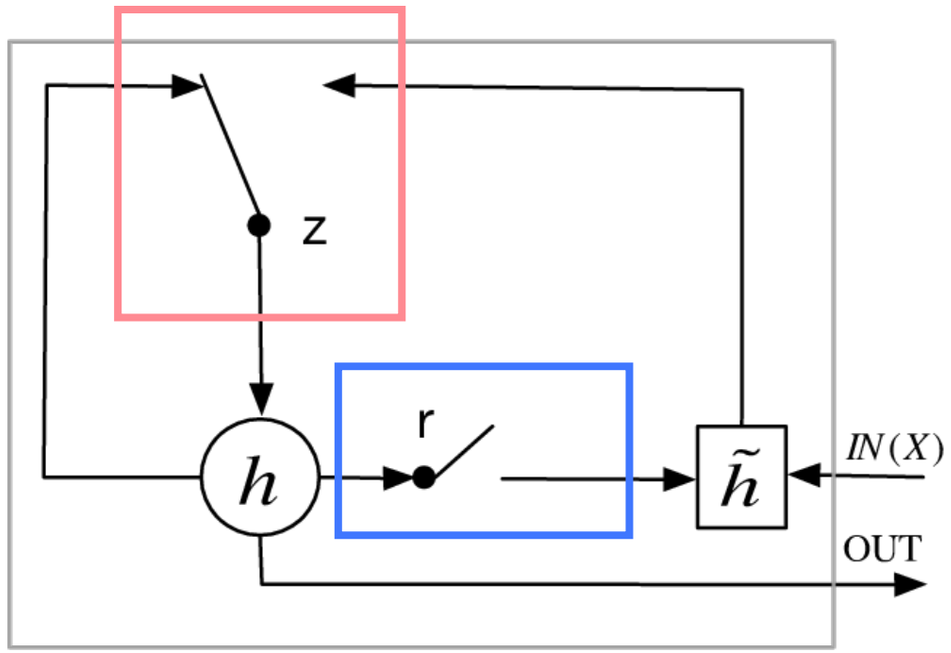
$$r_t = \sigma\left(W_r \cdot [h_{t-1}, x_t]\right)$$

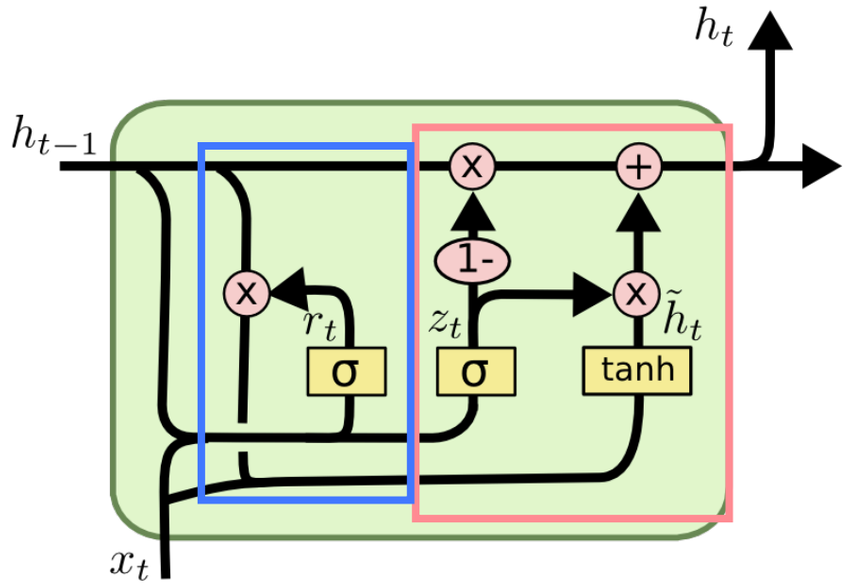
$$\tilde{h}_t = \tanh\left(W \cdot [r_t * h_{t-1}, x_t]\right)$$

$$h_t = (1 - z_t) * h_{t-1} + z_t * \tilde{h}_t$$

GRU

更新ゲート





リセットゲート

Encoder-Decoder Sequence-to-Sequence

Encoder-Decoder

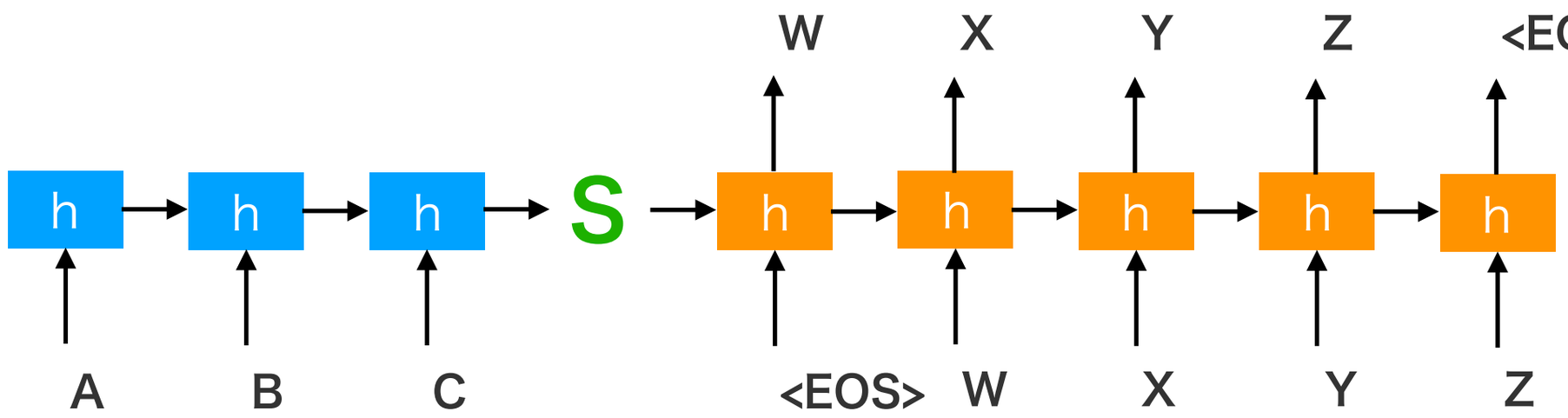
- Encoder部分とDecoder部分に分かれているNNの総称
- Encoder部分は入力を中間表現に変換する役割
- Decoderは中間表現から出力に変換する役割
- 日英翻訳でいえば、日本語の文章を中間表現に変換するのがEncoderで、その中間表現を英語の文章に変換するのがDecoder

Sequence-to-Sequence (seq2seq)

- 2014年に英仏翻訳用のモデルとして登場
- Encoder-Decoder を系列データに利用したもの
- 入力系列と出力系列の長さが一致せず、またどちらの系列も長さがまちまちな場合(翻訳など)に特に有用
- (似ているがEncoder-Decoderのほうが範囲が広い概念)

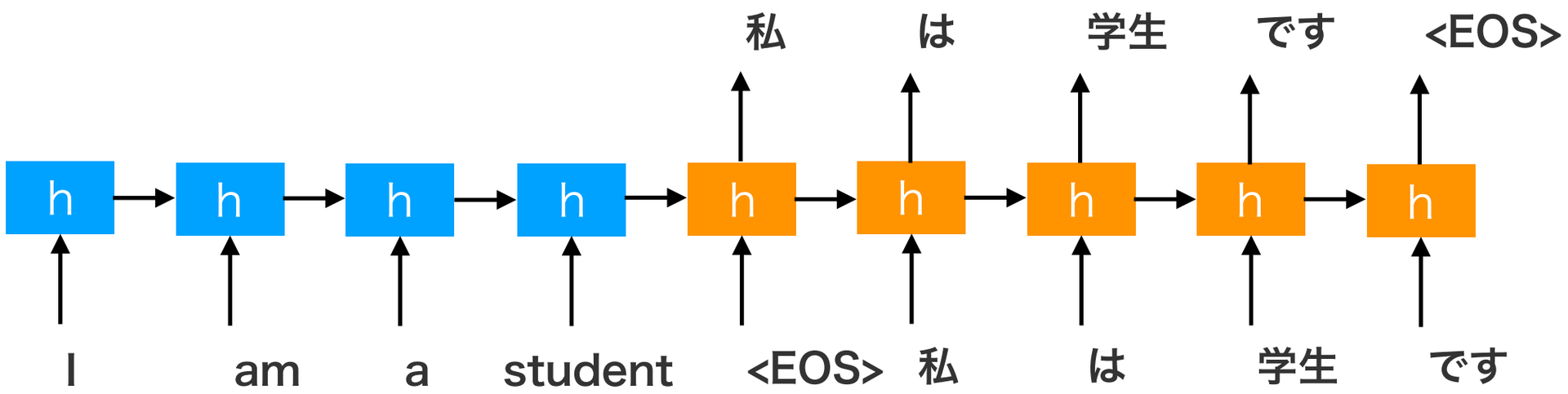
seq2seq

ABCを入力するとWXYZが出力される例
(EOS: End Of Sentence)



Encoder (ブルー) と Decoder (オレンジ) の2つのRNNを使う
2つのRNN同士をつなげるのはEncoderが出力したベクトルSのみ

seq2seq



英和翻訳に使う例（Latent Vector S は省略してある）

Attention

なぜAttention？

- 一応「注意」という訳語が与えられているが、あまり実態に即していないと言われているので単にattentionと呼ばれることが多い

- LSTMやseq2seqで長い系列が処理できるようになったが、入力が長すぎるとやはりDecoderに情報を渡すのが難しくなる

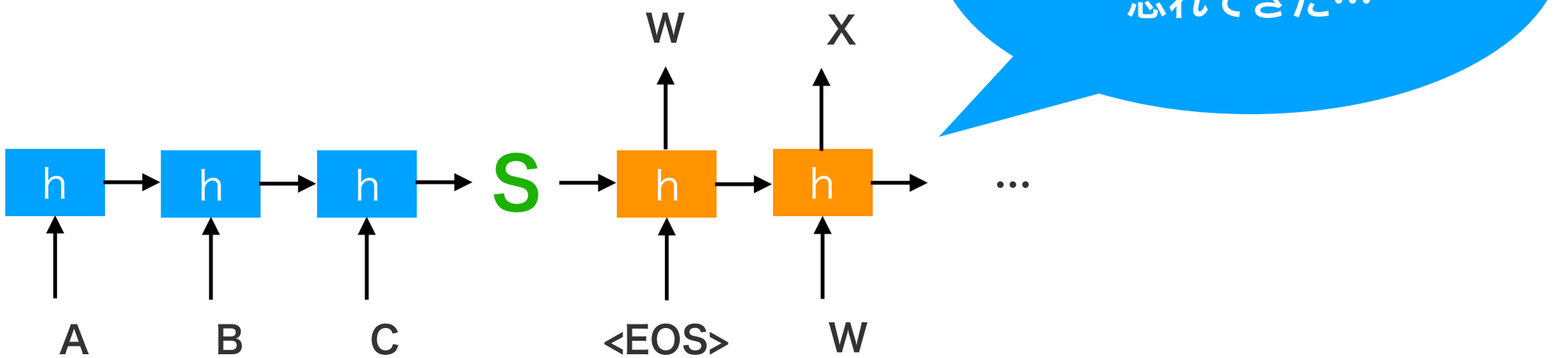
- そこでDecoderに入力を参照できるような能力を与える

なぜAttention？

- たとえばLSTMを使えば長い系列を学習することは可能だが、Encoder-Decoderモデルではの出力は1つの固定次元ベクトルに収めなければならず、長文では精度低下が避けられなかった
- attentionでは、「重点的に見なければいけない単語」には大きい重み、「そうでもない(重要でない)単語」には小さい重みを付けたベクトルもDecoderに渡す

- 時間方向にも重みを用意することで、情報の取捨選択をやりやすくする

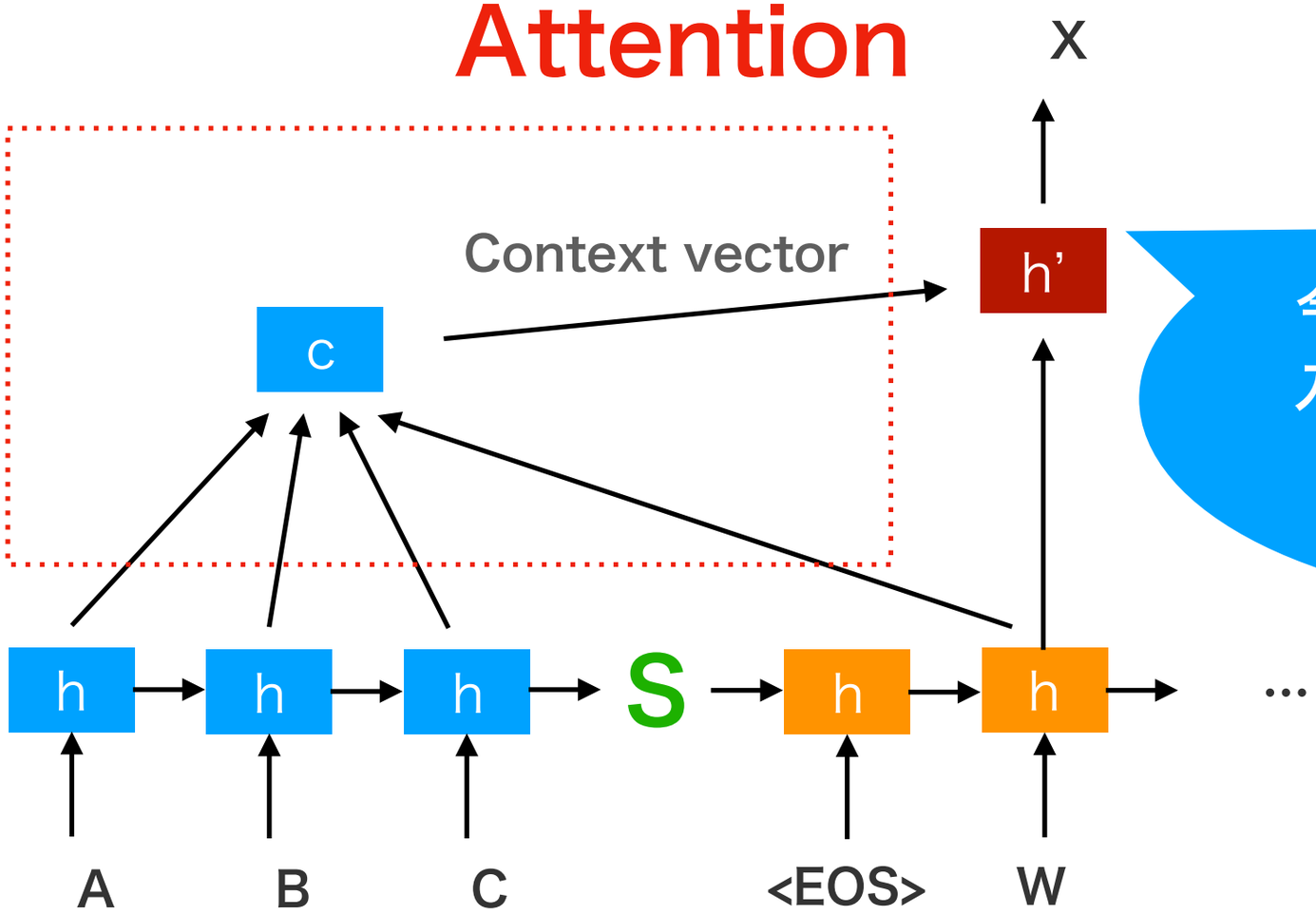
seq2seq



Decoderは再帰的結合で単語を出力していくので古い情報は次第に消えていく
これはEncoderが1つの固定ベクトルSしかdecoderに渡さないせい。
decoderに1つのベクトルを渡すのは同じだが、
そのベクトルが文脈に応じて動的に変わってくれたら嬉しい。

Attention

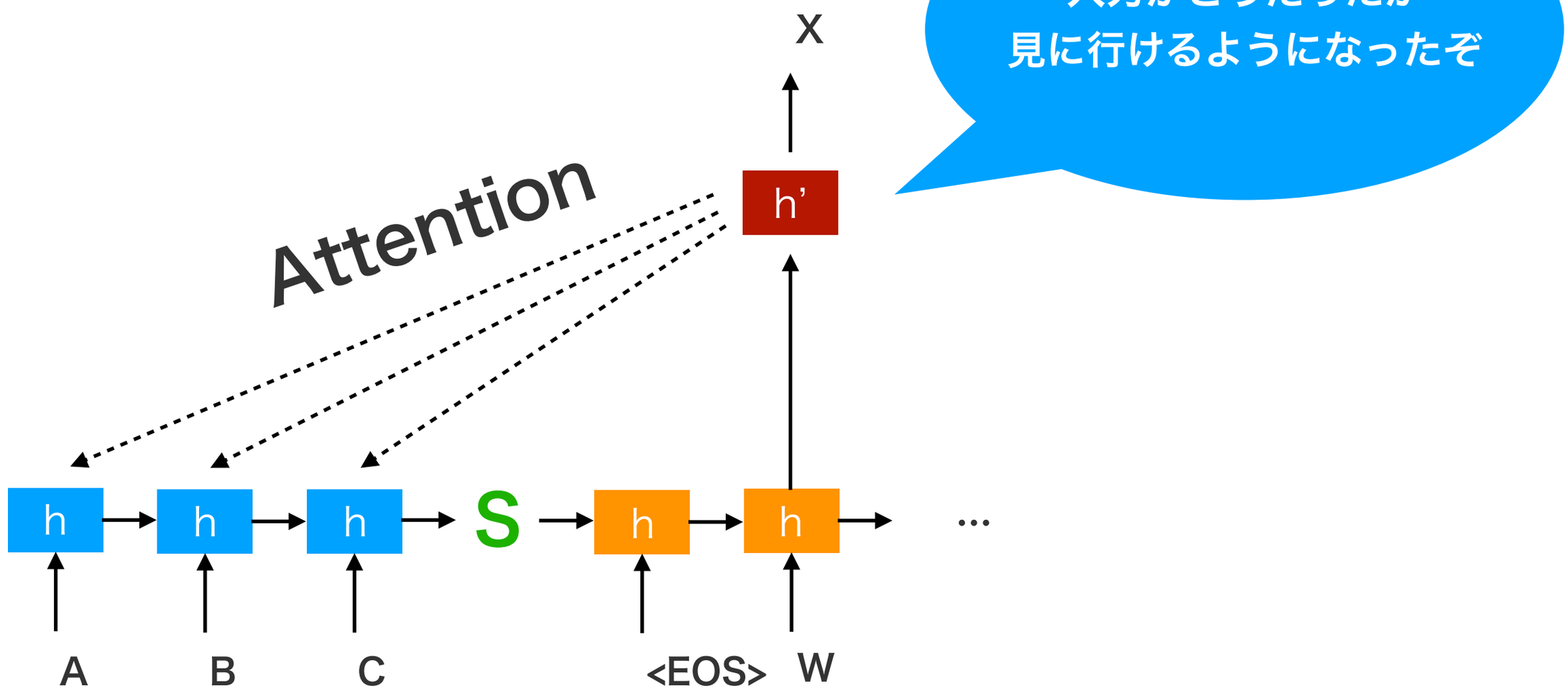
正確な実装とはやや異なる
(Luong et al. 2015)



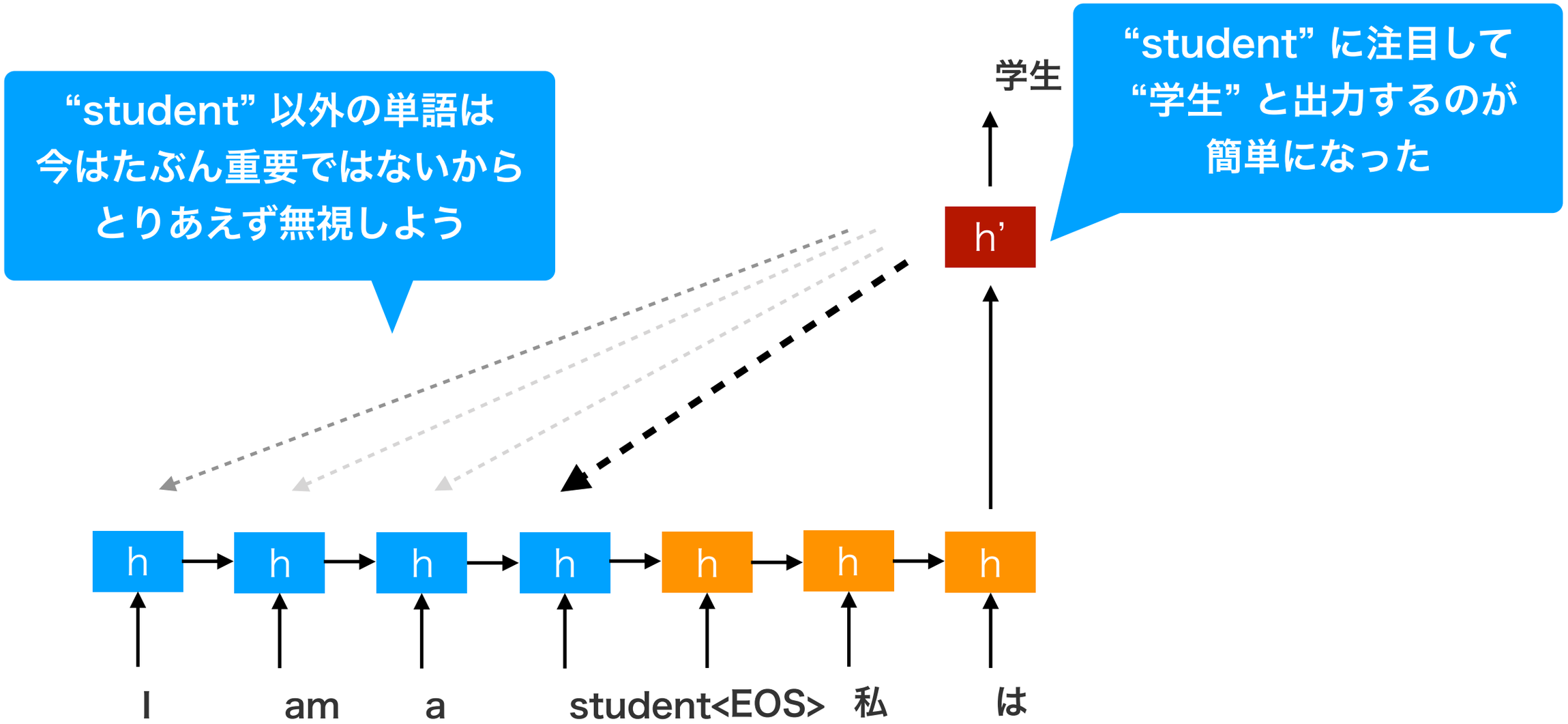
今までの情報に加えて
入力がどうだったかも
参照できるぞ

Decoderは再帰的結合で単語を出力していくので古い情報は次第に消えていく
これはEncoderが1つの固定ベクトルSしかdecoderに渡さないせい。
decoderに1つのベクトルを渡すのは同じだが、
そのベクトルが文脈に応じて動的に変わってくれたら嬉しい。

（あくまでも感覚としては）
こんな感じに解釈できる



Attentionがあればその時々に応じて適応的に重みを変化させて入力を参照できる



Attentionの応用

- Attentionはもともとは時系列のモデルに対して提案された手法だが、CNNに対しても適用する事例があり、汎用的な考え方であるといえる
- たとえば物体認識タスクでは背景領域は重要ではないが前景(物体)部分は重要であり、「どこを重点的に見るべきか」をAttentionでは陽に扱う
- チャンネルごとに Attention する(つまり、各チャンネルを均等に扱わず、適応的に重みをかけるようにする)、 Squeeze and Excitation Networks (SENet) なども有名

画像から文章を生成するときのAttention

A woman is throwing a frisbee in a park.

A dog is standing on a hardwood floor.

A stop sign is on a road with a mountain in the background.

A little girl sitting on a bed with a teddy bear.

A group of people sitting on a boat in the water.

A giraffe standing in a forest with trees in the background.

Xu, Kelvin, et al.
“Show, attend and tell: Neural image caption generation with visual attention.”