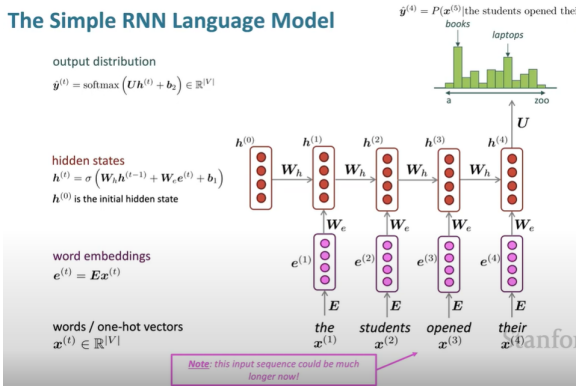
Lecture 6 - Simple and LSTM RNNs

1. The Simple RNN Language Model



옆에 나와 있는 RNN 구조에서 중요한 것은 word embedding 과정에서의 가중치 벡터 W_e와 Hidden States 과정에서의 가중치 벡터 W_h모두 매 time step 별로 동일한 값을 이용한다는 것이다.

Training an RNN Language Model

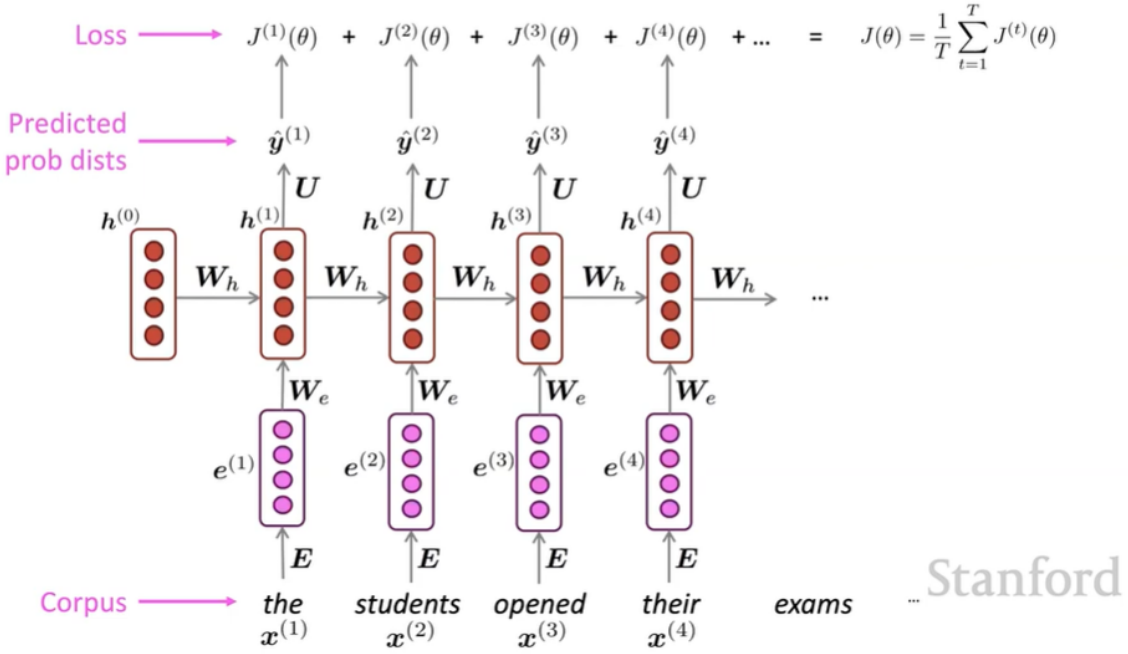
1. 무조건 큰 양의 text를 얻는 것, 단어의 시퀀스로 $x^{(1)}, \ldots, x^{(T)}$ 까지의 일련의 단어로 볼 수 있다.
2. RNN-LM에 input data로 넣는다. 그러면 매 time step 별로 출력 분포 $\hat{y}^{(t)}$를 계산한다. -. 우리의 모델을 훈련시키기 위함
3. Cross-Entropy를 사용해서 손실함수를 매 step별로 계산한다. → 이를 통해 손실을 줄이면서 어떤 단어가 나올지 예측

$$J^{(t)}(\theta) = CE(\boldsymbol{y}^{(t)}, \hat{\boldsymbol{y}}^{(t)}) = -\sum_{w \in V} \boldsymbol{y}_w^{(t)} \log \hat{\boldsymbol{y}}_w^{(t)} = -\log \hat{\boldsymbol{y}}_{\boldsymbol{x}_{t+1}}^{(t)}$$

출력값 → 확률 분포로 나타내는데, 1에 가까울수록 확실하게 예측할 수 있다는 것을 의미한다.

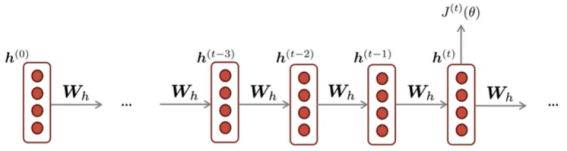
4. 전체 training set에 대해서 모든 loss를 더해 평균을 낸다.

$$J(\theta) = \frac{1}{T}\sum_{t=1}^{T} J^{(t)}(\theta) = \frac{1}{T}\sum_{t=1}^{T} -\log \hat{\boldsymbol{y}}_{\boldsymbol{x}_{t+1}}^{(t)}$$



전반적인 과정의 형태 하지만 이런 **신경모델을 훈련시키는 관점에서 너무 복잡, 불필요**

Training the parameters of RNNs : Backpropagation for RNNs

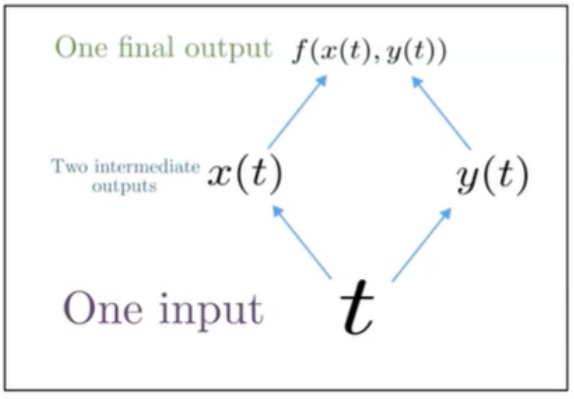


손실 함수 $J^{(t)}(\theta)$의 도함수 → 가중치 벡터 W_h에 대해서 편미분한 값의 합

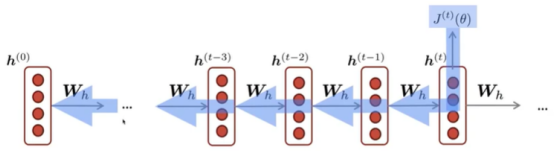
Multivariable Chain Rule

$$\underbrace{\frac{d}{dt} f(x(t), y(t))}_{\text{Derivative of composition function}} = \frac{\partial f}{\partial x}\frac{dx}{dt} + \frac{\partial f}{\partial y}\frac{dy}{dt}$$

$f(x, y)$ 이변수 함수가 주어져있다. input t에 대해서 $x(t), y(t)$ 두 개의 단일 변수로 나누고, 다시 output으로 대입하는 것



Backpropagation for RNNs



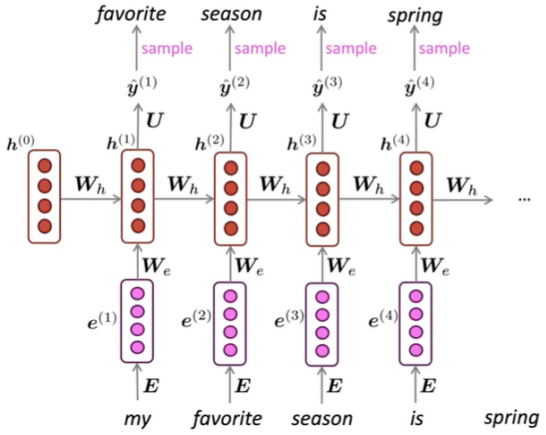
W_h 가중치 행렬을 업데이트하는 과정으로 역전파를 이용한다. W_h에 대해서 매 time step별로 손실함수를 편미분한다. 이 값들을 다 더해서 gradients 값으로 넣고 학습률을 곱해서 업데이트를 진행한다.

$$② \frac{\partial J^{(t)}(\theta)}{\partial W_h} = \sum_{i=1}^{t} \frac{\partial J^{(t)}(\theta)}{\partial \hat{y}^{(t)}} \cdot \frac{\partial \hat{y}^{(t)}}{\partial h^{(t)}} \cdot \frac{\partial h^{(t)}}{\partial h^{(i)}} \cdot \frac{\partial h^{(i)}}{\partial W_h}$$

$$\Rightarrow \frac{\partial J(\theta)}{\partial W_h} = \sum_{t=1}^{T} \frac{\partial J^{(t)}(\theta)}{\partial W_h}$$

$$W_h{}^{new} = W_h{}^{old} - \alpha \frac{\partial J(\theta)}{\partial W_h}$$

Generating text with a RNN Language Model



- N-gram Language Model 처럼, 반복된 sampling으로 text를 생성할 수 있다.
- my가 input으로 들어가면 word embedding, hidden state 들을 거쳐서 output 확률 분포로 favorite sample이 나올 것이다.
- output으로 나온 favorite은 다음 step의 input으로 들어가서 위 과정 동일하게 진행

> 💡 **RNN 언어 모델은 어떤 종류의 텍스트라도 훈련시킬 수 있다. (해리포터 책, 요리 레시피 책 등등…)**

Evaluating Language Models

>> perplexity (PPL) - 쉽게 생각해서 헷갈리는 정도로 의미 파악

$$\text{perplexity} = \prod_{t=1}^{T} \left(\frac{1}{P_{\text{LM}}(\boldsymbol{x}^{(t+1)} | \boldsymbol{x}^{(t)}, \ldots, \boldsymbol{x}^{(1)})} \right)^{1/T}$$

Normalized by number of words

Inverse probability of corpus, according to Language Model

- T개의 테스트 텍스트 샘플의 길이
- 식에서 현재 t에 대해서 t+1를 예측할 때 이전 sequence들에 대해서 $x^{(x+1)}$

에 대한 역수 값으로 나와 있어서 정답에 가까울 수록 0에 가까워지게 된다.

- PPL 수치가 낮을수록 좋은 성능의 언어 모델임을 나타낸다.
- 손실함수 Cross-Entropy에 지수 형태로 나타낸 것과 값이 같다.

- This is equal to the exponential of the cross-entropy loss $J(\theta)$:

$$= \prod_{t=1}^{T} \left(\frac{1}{\hat{\boldsymbol{y}}_{\boldsymbol{x}_{t+1}}^{(t)}} \right)^{1/T} = \exp\left(\frac{1}{T} \sum_{t=1}^{T} -\log \hat{\boldsymbol{y}}_{\boldsymbol{x}_{t+1}}^{(t)} \right) = \exp(J(\theta))$$

💡 **Lower perplexity is better!!**

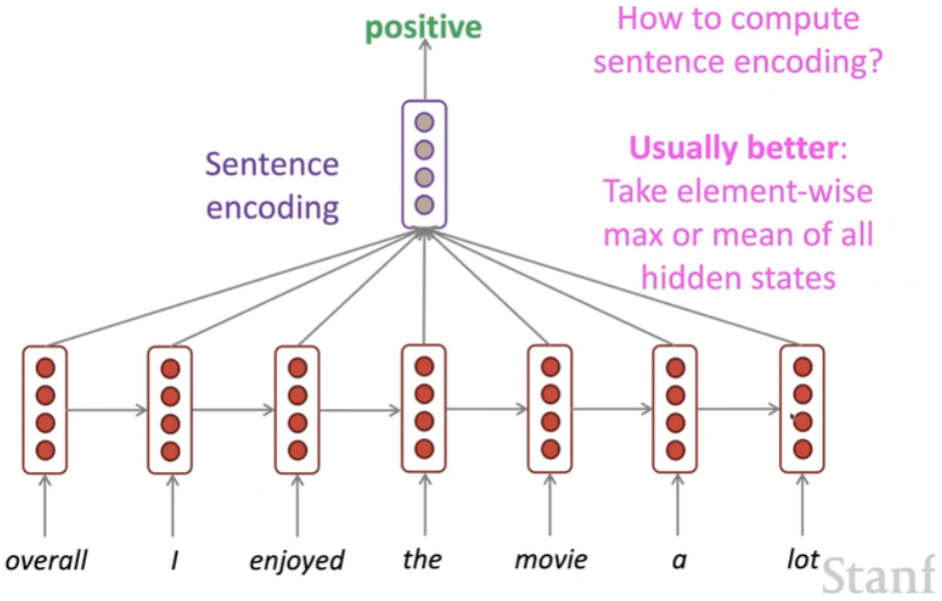
Recap

> **Language Model : 다음 단어를 예측하는 시스템이다.**
>
> **Recurrent Neural Network : 어떤 T개의 길이의 순차적인 input으로 취하고, hidden state를 생성하기 위해 동일한 weight를 각 time step 별로 적용시킨다.**
>
> **각 step에서 output을 최적으로 생산하는 것이다.**
>
> $RNN\ !=\ Language\ Model$

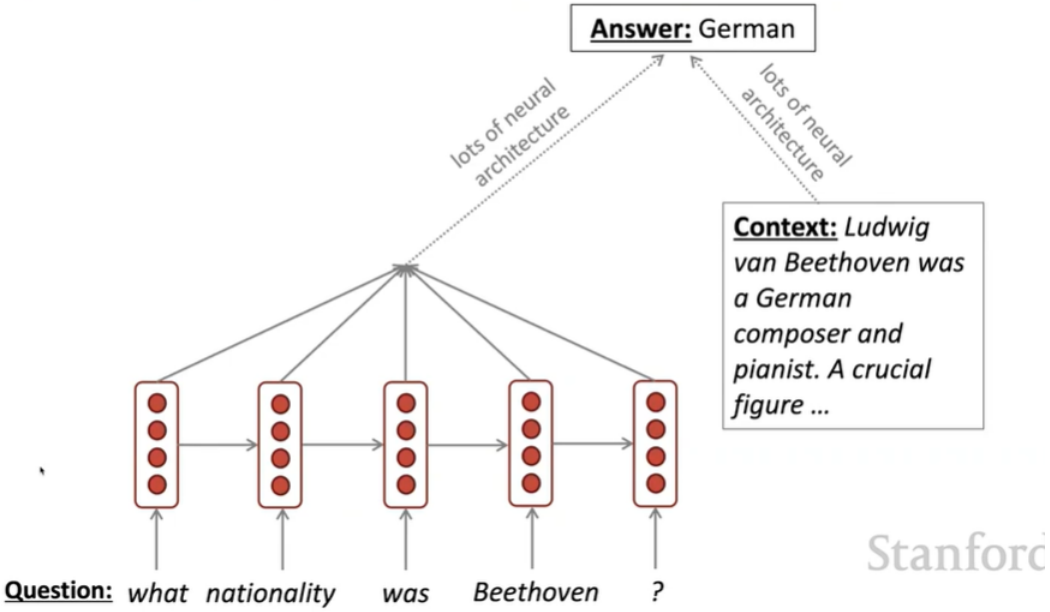
---

2. RNN의 다른 용도

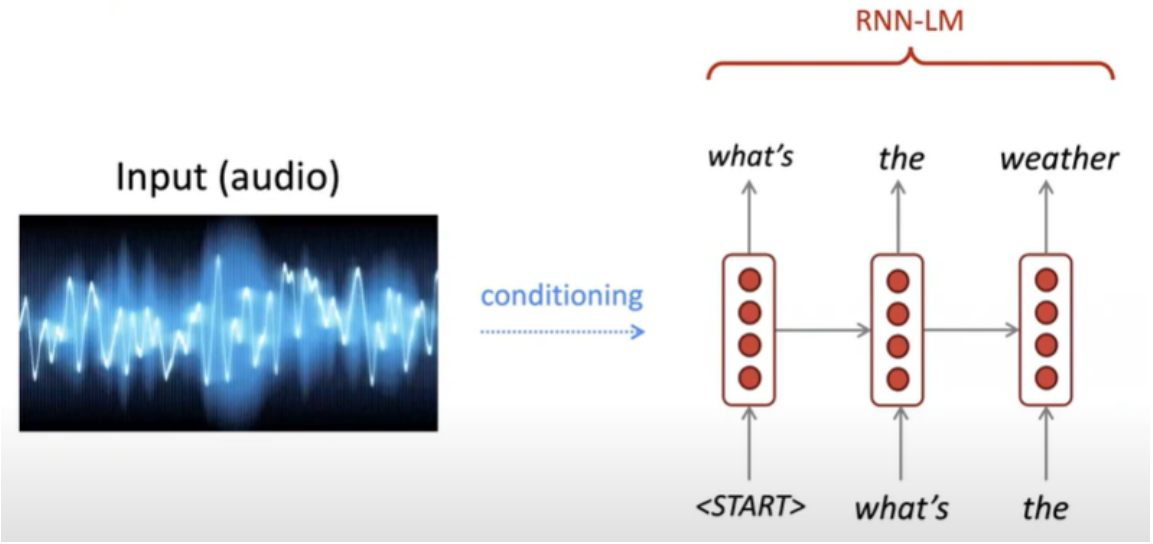
▼ Sequence Tagging : 텍스트 분류 작업 (sentiment classification 감성 분류)



▼ Language encoder module : question answering(QA), 기계 번역 등등

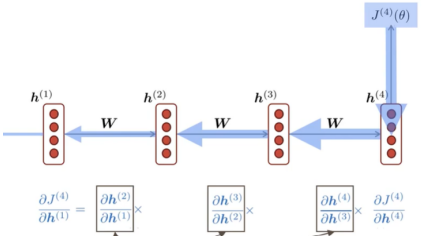


▼ Generate text : speech recognition(음성 인식), 기계 번역

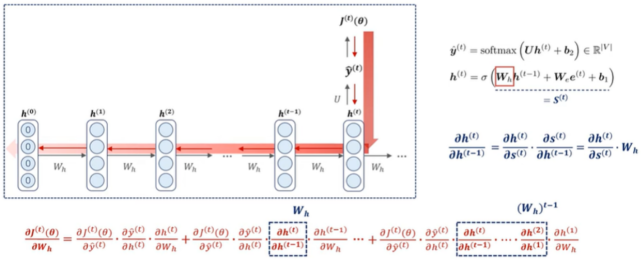


3. Problems with Vanishing and Exploding Gradients

기울기 소실 문제



- 옆에 그림에서 역전파과정에서 $J^{(4)}$을 $h^{(1)}$로 도 함수를 계산하고자 한다.
- 그러면 연쇄 규칙으로 편미분들의 곱으로 나타낼 수 있게 된다.
- 이 때 편미분 값들이 작이지게 되면, gradient는 점점 작아지게 되고 사리지기 시작한다.

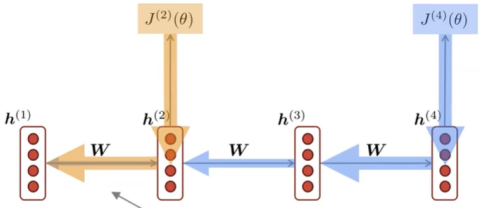


- 빨간색 식을 보면, W_h의 gradient를 구하기 위한 식이 나와 있다(각 time step 별로 연쇄 과정 추가). 그 중 hidden state 값끼리 편미분을 도출하는 과정을 자세히 살펴보고자 한다.
- 옆에 그림을 보면,$h^{(t)}$를 구할 때, 시그모이드 함수를 씌우기 전의 식을 $S^{(t)}$라고 하고자 한다.

- 손실함수의 gradient를 계산하기 전에, hidden state를 먼저 고려해야 하는데, 남색 식을 살펴보면, 편미분에 대해서 W_h의 곱으로 간략하게 나타낼 수 있다.
- 각 time step 별로 hidden state 끼리 편미분을 할 때 W_h가 횟수만큼 곱해지게 된다.

> **동일한 가중치(W_h)를 공유**
>
> **>> 가중치가 작을수록 (<1) 반복적으로 곱해지는 값이 0에 가까워져서 gradient vanishing 문제**
>
> **>> 가중치가 클수록 (>1) 반복적으로 곱해지는 값이 기하급수적으로 커져서 gradient exploding 문제**

왜 기울기 소실이 문제가 될까 (장기 의존성)



- $h^{(1)}$까지 도달하는 데에 있어서 가까운 영향을 모델링하는 데는 매우 좋지만, 멀리서 오는 그래디언트 신호가 너무 많이 손실된다.
- 즉 모델의 가중치는 가까운 곳에 있는 영향으로만 update가 이뤄지고, 멀리 있는 영향은 고려되지 않는다.

왜 기울기 폭주는 문제가 될까

$$\theta^{new} = \theta^{old} - \overbrace{\alpha}^{\text{learning rate}} \underbrace{\nabla_\theta J(\theta)}_{\text{gradient}}$$

- **gradient가 너무 크게 되면, update가 제대로 되지 않을 것이다.**
- **이상한 지점으로 도달하게 되거나, 아예 파라미터 업데이트 자체가 이뤄지지 않을 수 있다.**

Gradient clipping : 기울기 폭주 문제 해결 (깎는 것)

Algorithm 1 Pseudo-code for norm clipping

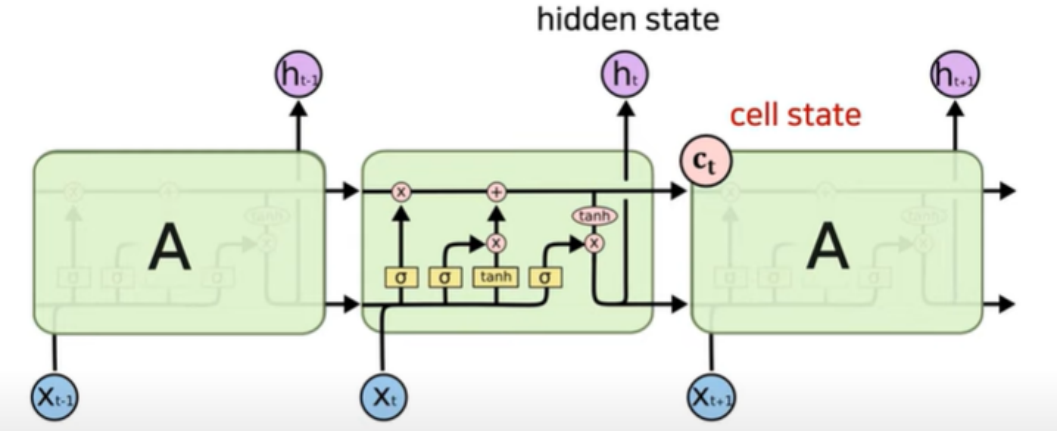
$\hat{\mathbf{g}} \leftarrow \frac{\partial \mathcal{E}}{\partial \theta}$
if $\|\hat{\mathbf{g}}\| \geq threshold$ then
  $\hat{\mathbf{g}} \leftarrow \frac{threshold}{\|\hat{\mathbf{g}}\|}\hat{\mathbf{g}}$
end if

- gradient가 일정 threshold를 넘어가게 되면, clipping 진행
- clipping → gradient의 norm(보통은 L2 norm)으로 나눔
- threshold → gradient가 가질 수 있는 최대 L2norm을 뜻
  - 하이퍼파라미터로 사용자가 설정

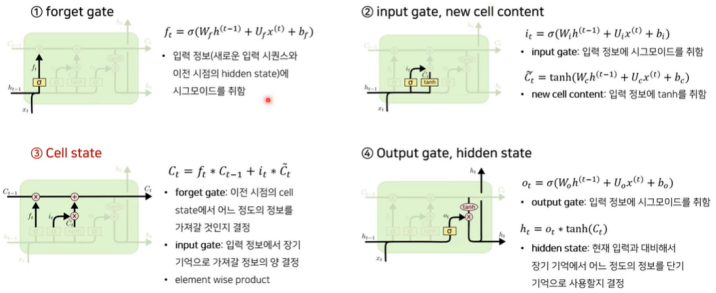
---

4. Long Short - Term Memory RNNs (LSTMs)

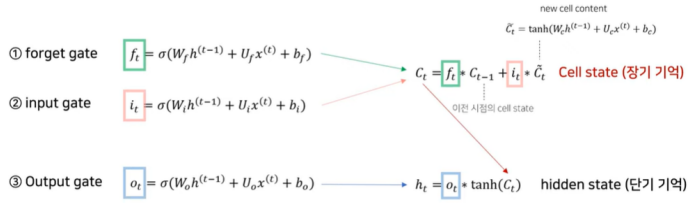
>> 기울기 소실로 인한 장기 의존성 문제점 해결



💡 **time step t에 대해서 hidden state(short-term memory 조절) $h^{(t)}$ 와 cell state(long-term memory 보존) $c^{(t)}$ 존재한다.**



Cell state를 설명하기에 앞서 forget gate와 input gate, new cell content가 정의되어야 한다.



처음에 구한 Cell state에 대해서 h_t는 tanh 활성화함수를 통해 현재 입력과 대비해서 장기 기억에서 어느 정도의 정보를 단기 기억으로 사용할 것인지 사용 이때 o_t(Output gate)와 함께 고려

1. forget gate : 입력 정보(새로운 입력 시퀀스(x^t)와 이전 시점의 hidden state($h^{(t-1)}$)에 시그모이드를 취한다.
2. input gate : 입력 정보에 시그모이드를 취한다.
3. new cell content : 입력 정보에 tanh를 취한다.
4. ouput gate : 입력 정보에 시그모이를 취한다.

Cell state

Hidden state

- 하다마드 곱으로 나타낸다. (element wise product)

→ 이 모든 것들은 같은 길이의 벡터 형태로 이뤄져 있다.

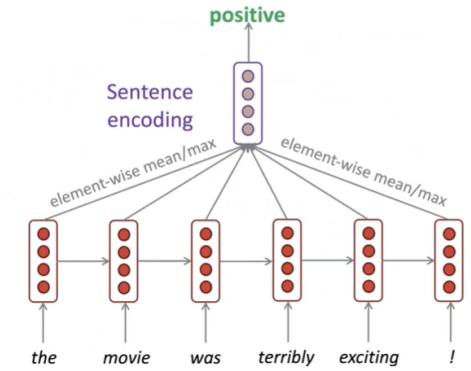
LSTM을 통한 문제점 해결

>> 많은 time step 마다의 정보량들을 보존하는 면에서 기존의 RNN 모델 보다 쉽게 할 수 있다.

그러나 LSTM은 기울기 소실과 폭주를 없앤다고는 말할 수 없다. 하지만 LSTM은 장기 의존성 문제를 해결할 수는 있다.

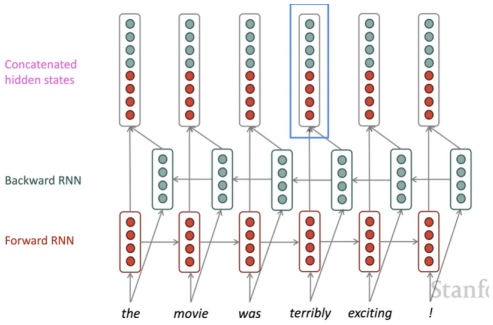
---

5. Bidirectional and Multi-layer RNNs: motivation



- contextual representation : 같은 단어라도 문맥에 따라 다른 의미로 쓰이는 것
- terribly라는 끔찍하게 라는 단어가 있을 때 일반적으로 negative한 의미로 쓰인다.

Bidirectional RNNs



- 2개의 Forward와 Backward RNN을 통해 최종 output은 두 RNN의 은닉상태를 통해 concat 계산.
- 첫번째 토큰부터 시작하는 forward mode의 RNN
- 맨 마지막에서 시작해서 처음으로 가는 backward mode의 RNN

Forward RNN $\overrightarrow{h}^{(t)} = \mathrm{RNN}_{\mathrm{FW}}(\overrightarrow{h}^{(t-1)}, x^{(t)})$

Backward RNN $\overleftarrow{h}^{(t)} = \mathrm{RNN}_{\mathrm{BW}}(\overleftarrow{h}^{(t+1)}, x^{(t)})$

Concatenated hidden states $h^{(t)} = [\overrightarrow{h}^{(t)}; \overleftarrow{h}^{(t)}]$

>> 감정 분류 및 QA와 같은 문제를 풀 때에는 양방향 RNN을 사용하는 것이 아주 좋은 방법이다. (단 전체의 input sequence를 접근할 수 있을 때)

++) 보충

teacher forcing 기법 추가하기