

Natural Language Processing with DeepLearning

week 6

The course

- RNN Language models
- Other uses of RNNs
- Exploding and Vanishing Gradients
- LSTMs
- bidirectional and Multi layer RNNs

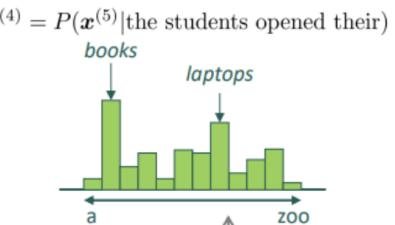
1. RNN Language models

simple RNN을 사용하여 입력값의 크기에 의존하지 않으며 어순에 대한 고려 가능

A Simple RNN Language Model

output distribution

$$\hat{y}^{(t)} = \text{softmax}(\mathbf{U}\mathbf{h}^{(t)} + \mathbf{b}_2) \in \mathbb{R}^{|V|}$$



hidden states

$$\mathbf{h}^{(t)} = \sigma(\mathbf{W}_h \mathbf{h}^{(t-1)} + \mathbf{W}_e \mathbf{e}^{(t)} + \mathbf{b}_1)$$

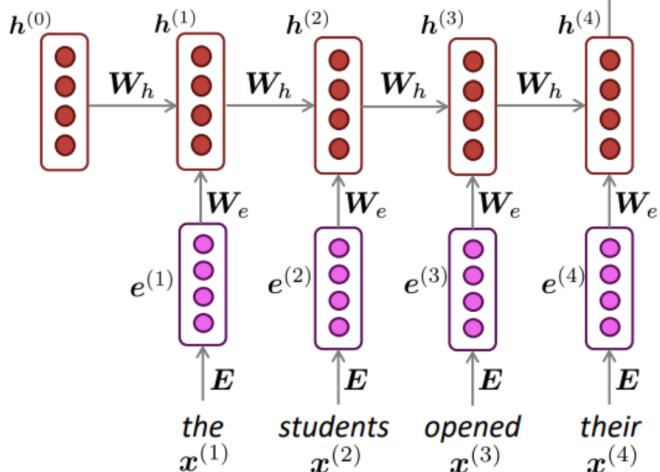
$\mathbf{h}^{(0)}$ is the initial hidden state

word embeddings

$$\mathbf{e}^{(t)} = \mathbf{E}\mathbf{x}^{(t)}$$

words / one-hot vectors

$$\mathbf{x}^{(t)} \in \mathbb{R}^{|V|}$$



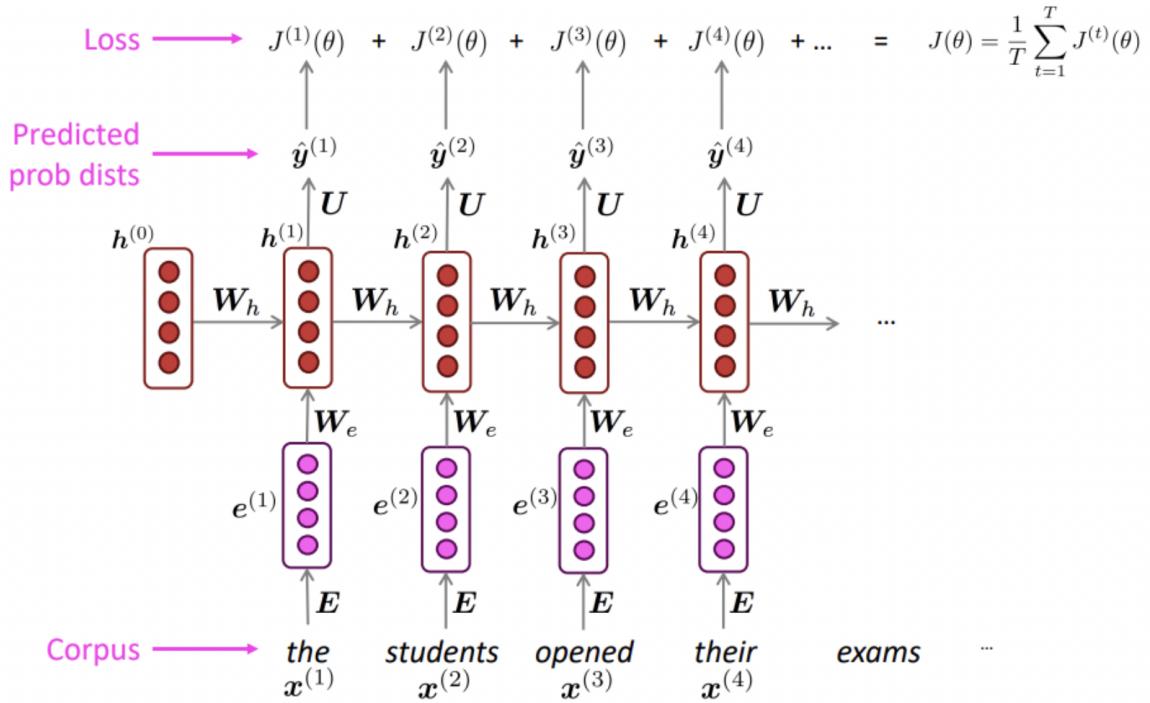
Note: this input sequence could be much longer now!

<RNN 언어 모델의 학습>

- 단어들의 나열인 $x^{(1)}, x^{(2)}, \dots, x^{(T)}$ 큰 말뭉치를 준비
- RNN 언어 모델에 넣어 결과 확률 분포 $\hat{y}^{(t)}$ 을 매 timestep t마다 구한다
- timestep t에서의 손실 함수(Loss function)은 예측 확률 분포 $\hat{y}^{(t)}$ 와 실제 단어 $y^{(t)}$ 의 cross-entropy

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

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



- teacher forcing: 연속적인 예측을 수행할 경우 한 timestep에서 잘못 예측하였을 때 다음 timestep에 계속해서 잘못된 추론이 입력으로 들어갈 수 있음.

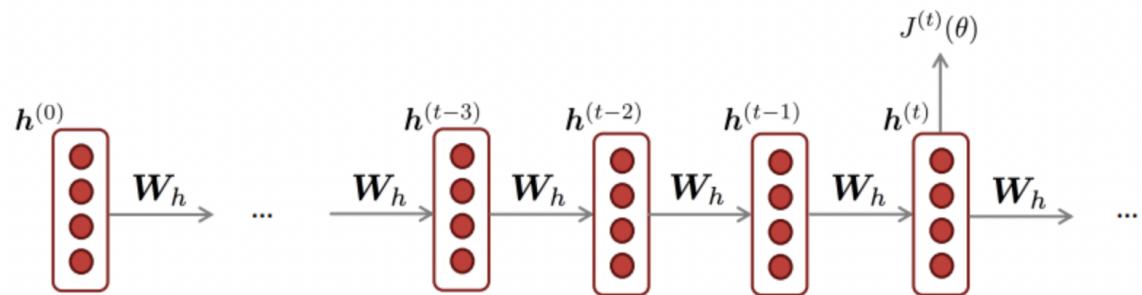
=> 각각의 입력은 ground truth 값으로 넣어주어 매 timestep마다 조건이 제대로 주어졌을 때를 기준으로 학습할 수 있도록 함

+) 전체 밀봉치를 학습: 너무 연산량이 많음

=> 일반적으로 문서 또는 문장 단위로 나누어서 학습을 진행

SGD(stochastic gradient descent)에서 데이터의 일부만 이용해서 기울기를 update 했었기 때문에 실제로 문장 혹은 여러 문장 단위로 기울기를 계산하고 update 함

<RNN 매개변수의 학습>

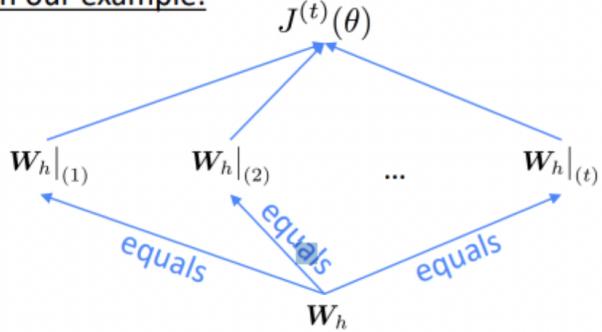


- 가중치에 대한 손실함수 미분 ?

$$\text{A: } \frac{\partial J^{(t)}}{\partial W_h} = \sum_{i=1}^t \frac{\partial J^{(t)}}{\partial W_h} |_{(i)}$$

- 모든 timestep에 대한 W_h 는 아래와 같이 정리됨, 이 때 각 time step에서 W_h 는 같은 값을 사용해서 1이 된다

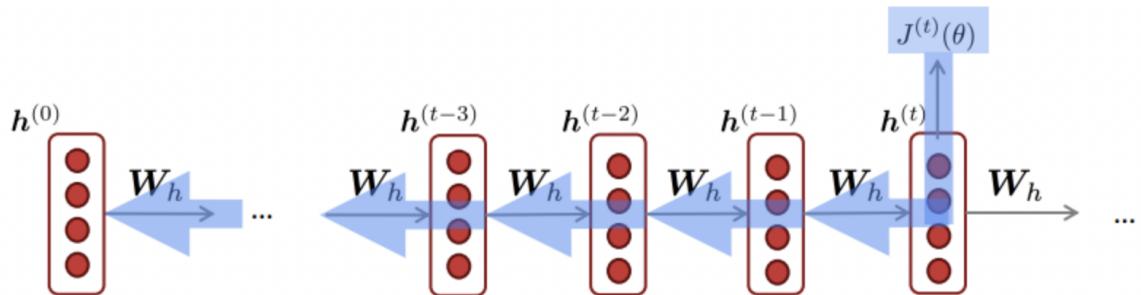
In our example:



Apply the multivariable chain rule:

$$= 1$$

$$\begin{aligned} \frac{\partial J^{(t)}}{\partial W_h} &= \sum_{i=1}^t \frac{\partial J^{(t)}}{\partial W_h} \Big|_{(i)} \frac{\partial W_h|_{(i)}}{\partial W_h} \\ &= \sum_{i=1}^t \frac{\partial J^{(t)}}{\partial W_h} \Big|_{(i)} \end{aligned}$$

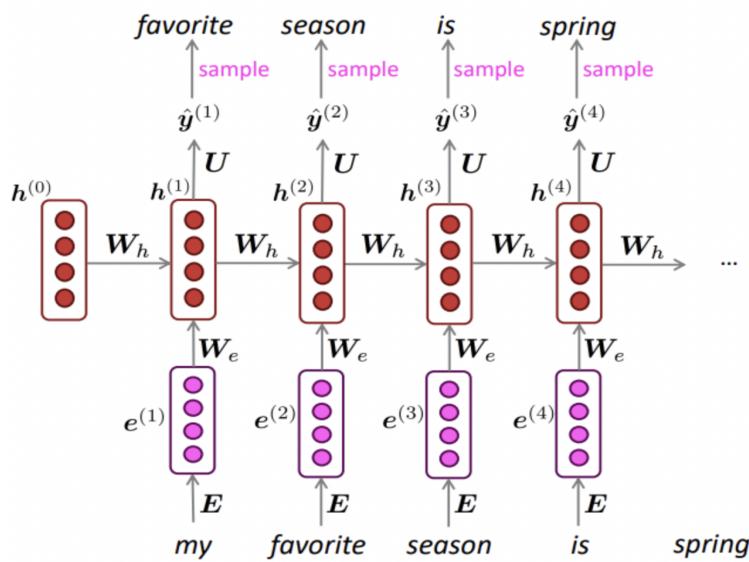


RNN의 역전파 개요

=> 위와 같이 각 timestep에 대한 기울기를 최종 손실 함수로부터 구하여 역전파해 나감
이 때 timestep에 따라 역전파 해 시퀀스의 처음까지 간다고 해서 BPTT(backpropagation through time)라 부름

가끔 시퀀스가 너무 길면 연산속도가 너무 느려지기 때문에 update하는 최대 timestep 길이에 제한을 두는 방법을 truncated BPTT라고 함.

<RNN 언어모델의 문서 생성>



RNN으로 문서를 생성하는 과정

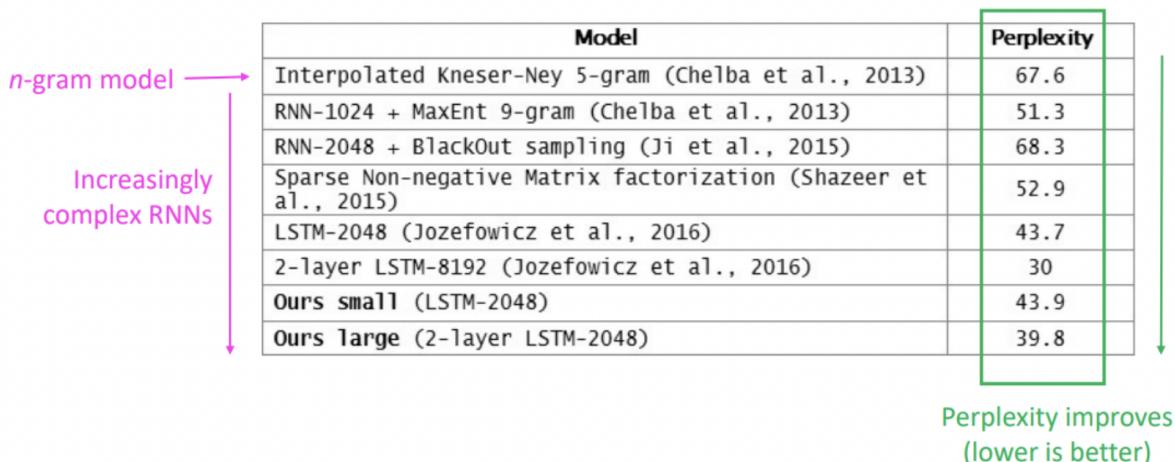
: 위의 예시처럼 단어를 주고 시작할 수도 있고 아무것도 없이 시작하려면 **special token**을 입력으로 준 다음 확률 분포로부터 단어를 샘플링함.

이후 샘플링된 단어: 다음 **timestep**의 입력으로 들어가게되고 최종적으로 문장의 끝을 나타내는 **special token**이 나타나면 생성은 종료됨

<언어 모델 평가>

perplexity는 작을수록 좋음

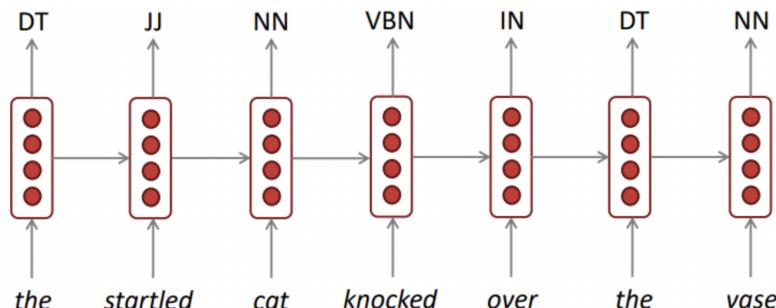
$$\text{perplexity} = \prod_{t=1}^T \left(\frac{1}{P_{LM}(x^{(t+1)} | x^{(t)}, \dots, x^{(1)})} \right)^{1/T}$$



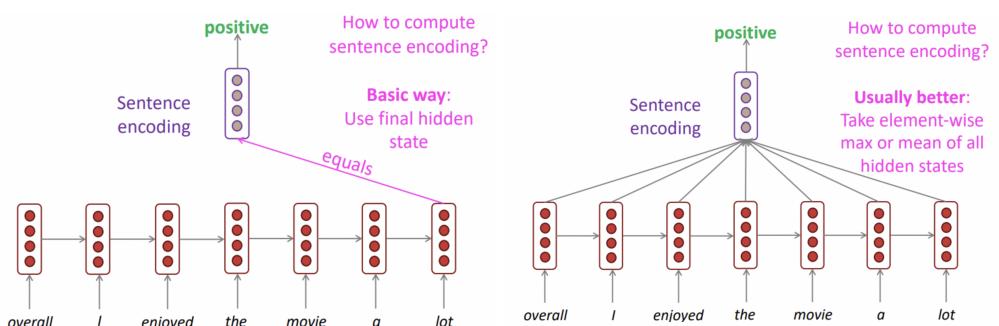
모델 별 perplexity 비교

2. Other uses of RNNs

- Sequence tagging
ex) 품사태깅, NER



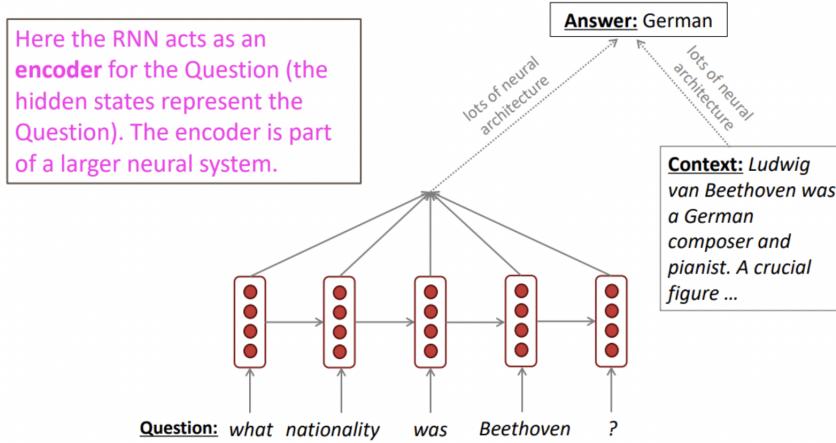
- Sentiment Classification



최종 분류기를 RNN위에 추가적으로 올려서 sentence encoding을 수행하게 되는데, 이를 수행하는 가장 기본적인 방법은 문장의 마지막 단어의 hidden state를 입력으로 넣음.

3) Language Encoder module

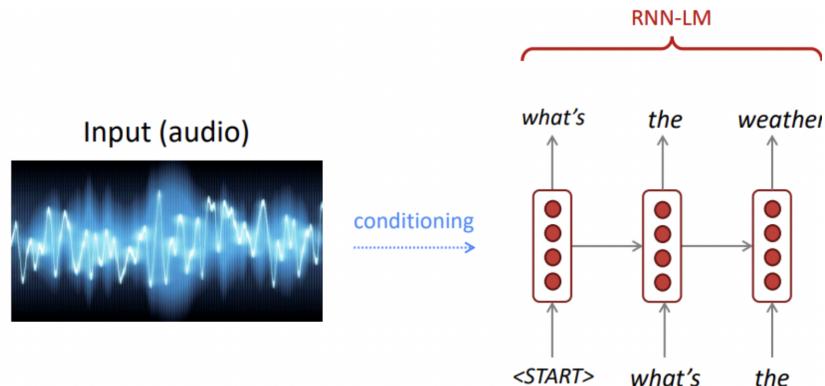
ex) 질의 응답, 기계번역



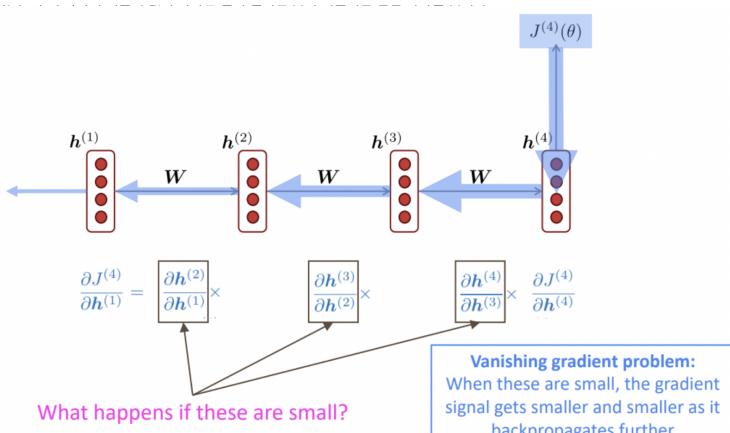
: 질의응답의 경우 RNN이 질문을 인코딩한 후 다른 신경망 아키텍처를 통해 정답을 추론하고, 정답을 추론하기 위해 문맥을 학습하는 또 다른 신경망 아키텍처도 구축

4) Text generation

ex) 음성 인식, 기계번역, 요약



3. Exploding and Vanishing Gradients



: 화살표의 두께 = 기울기의 크기

<Vanishing Gradient Proof 스케치>

- Recall: $\mathbf{h}^{(t)} = \sigma(\mathbf{W}_h \mathbf{h}^{(t-1)} + \mathbf{W}_x \mathbf{x}^{(t)} + \mathbf{b}_1)$
- What if σ were the identity function, $\sigma(x) = x$?

$$\frac{\partial \mathbf{h}^{(t)}}{\partial \mathbf{h}^{(t-1)}} = \text{diag}(\sigma'(\mathbf{W}_h \mathbf{h}^{(t-1)} + \mathbf{W}_x \mathbf{x}^{(t)} + \mathbf{b}_1)) \mathbf{W}_h \quad (\text{chain rule})$$

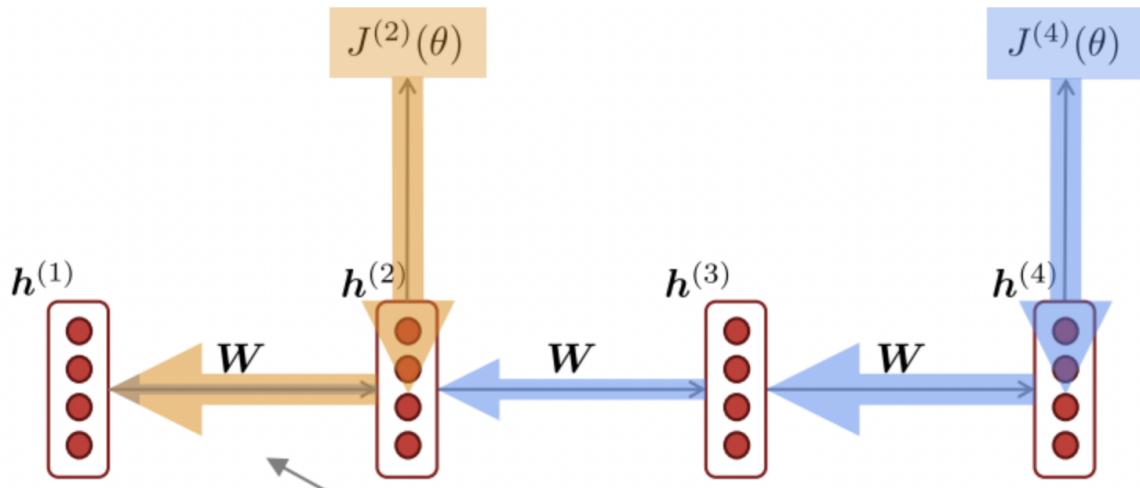
 $= \mathbf{I} \mathbf{W}_h = \mathbf{W}_h$
- Consider the gradient of the loss $J^{(i)}(\theta)$ on step i , with respect to the hidden state $\mathbf{h}^{(j)}$ on some previous step j . Let $\ell = i - j$

$$\begin{aligned} \frac{\partial J^{(i)}(\theta)}{\partial \mathbf{h}^{(j)}} &= \frac{\partial J^{(i)}(\theta)}{\partial \mathbf{h}^{(i)}} \prod_{j < t \leq i} \frac{\partial \mathbf{h}^{(t)}}{\partial \mathbf{h}^{(t-1)}} \quad (\text{chain rule}) \\ &= \frac{\partial J^{(i)}(\theta)}{\partial \mathbf{h}^{(i)}} \prod_{j < t \leq i} \mathbf{W}_h = \frac{\partial J^{(i)}(\theta)}{\partial \mathbf{h}^{(i)}} \boxed{\mathbf{W}_h^\ell} \quad (\text{value of } \frac{\partial \mathbf{h}^{(t)}}{\partial \mathbf{h}^{(t-1)}}) \end{aligned}$$

If \mathbf{W}_h is "small", then this term gets exponentially problematic as ℓ becomes large.

<기울기 소실이 문제인 이유>

: timestep이 멀리 떨어진 경우 기울기가 거의 소실되어 인접한 기울기만 매개 변수 업데이트에 반영됨



h^1 업데이트 시 인접한 (2)의 정보(주황색)가 멀리 떨어진 (4)의 정보(파란색)보다 많이 반영된다.

ex)

LM task: When she tried to print her [tickets](#), she found that the printer was out of toner.
She went to the stationery store to buy more toner. It was very overpriced. After
installing the toner into the printer, she finally printed her _____.

=> 사람: "tickets"인 것을 파악 가능, RNN 언어 모델: 기울기가 작을 경우, 이 관계성을 학습불가, 일반적으로 RNN의 경우 7번 timestep 이상의 정보는 잊어버림

<Gradient Exploding and Clipping>

$$\theta^{new} = \theta^{old} - \alpha \bigtriangledown_{\theta} J(\theta)$$

기울기의 값이 너무 커지게 될 경우 매개변수가 너무 가파르게 업데이트 되어 이상하거나 나쁜 분포를 갖게 됨. 최악의 경우 그 결과값이 Inf, NaN 값을 갖게 됨

Clipping :

Algorithm 1 Pseudo-code for norm clipping

```

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

```

: 기울기에 threshold를 정해주어 SGD 업데이트 전에 그 크기를 조정

4. LSTMs

: 1997년 Hochreiter and Schmidhuber가 제안한 RNN기반의 LSTM이 기울기 소실 문제 해결을 위해 제안됨. 구성: step t에 대해서 hidden state, cell state

- 벡터의 길이는 n으로 같다

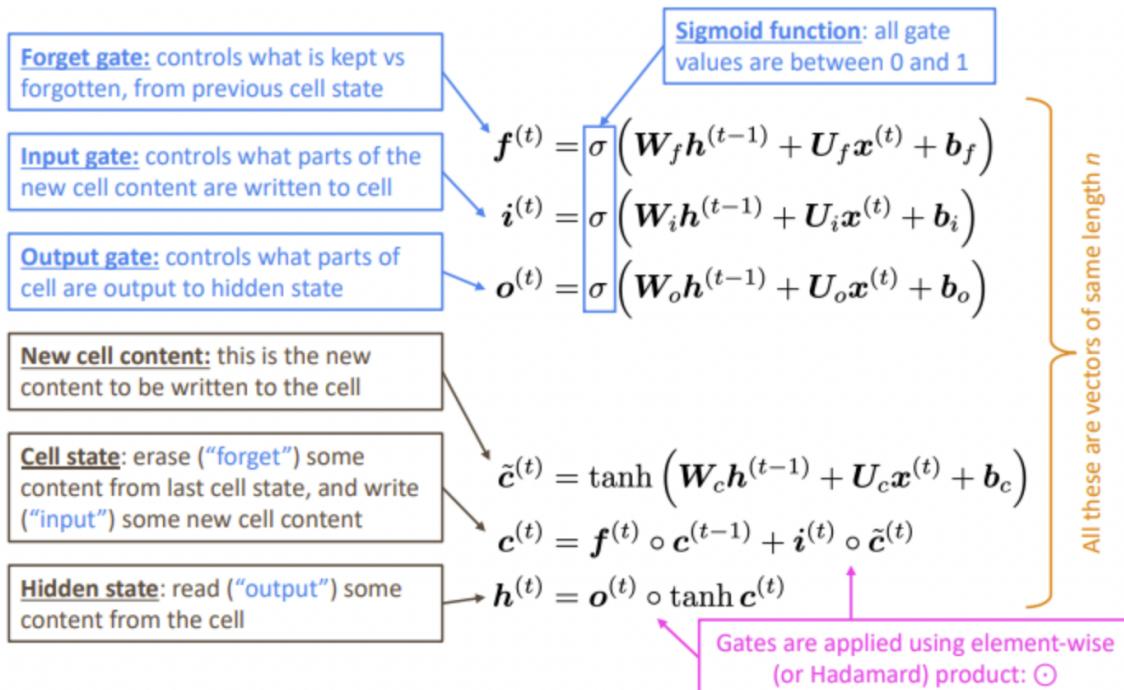
- cell 이 장기(long-term) 정보를 보관한다

- LSTM은 cell에 정보를 읽고, 지우고, 쓰기가 가능하다

: 정보의 읽기, 지우기, 쓰기는 gates로 조작된다

- gates 역시 길이는 n
- open(1), closed(0)로 각 timestep에 대해 정보를 조작
- dynamic gates로 현재 시점의 문맥에 따라 조정된다.

$x(t)$ 의 입력에 따라 hidden state와 cell state를 time step에 따라 각각 계산



- LSTM은 2개의 hidden vector를 사용한다

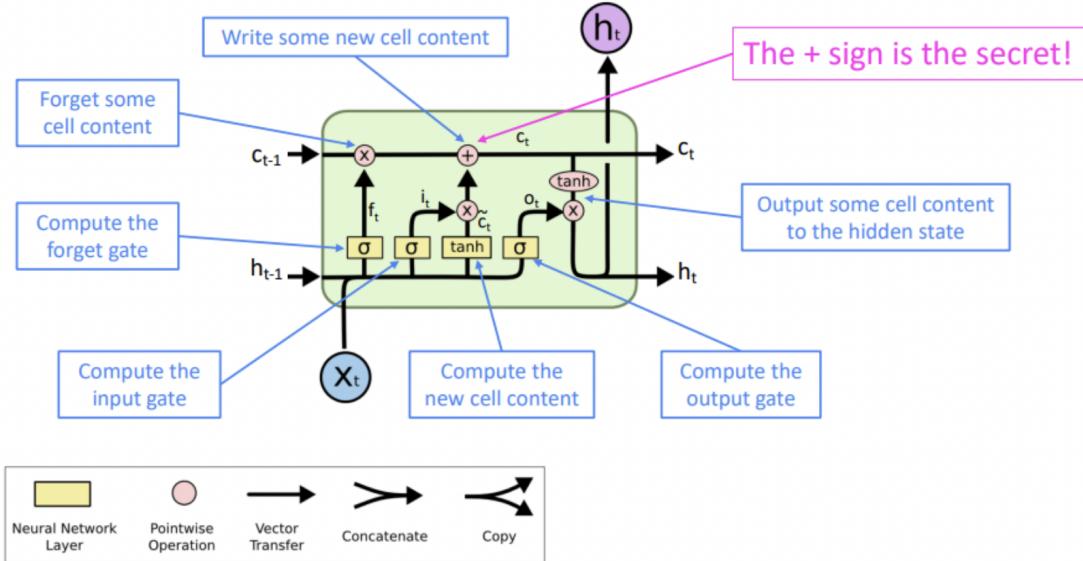
- hidden state: output gate를 통해 현재 cell 정보로부터 일부를 추출.
- cell state: long-term 정보 저장. forget gate로부터 이전 cell의 정보를 일부 지우고, input gate로 부터 현재 정보 일부를 추가하여 생성.

- 3개의 gate를 사용하여 매 time step의 cell state와 hidden state, input에서 취할 정보의 양을 결정한다.

forget gate: 이전 timestep의 cell state ($t-1$)에서 어느 정도의 정보를 가져갈 것인지 결정

input gate: 현재 timestep의 입력 $x(t)$ 과 이전 timestep의 hidden state $h(t-1)$ 을 통해 생성된 현재 cell 정보에서 cell state로 가져갈 정보의 양 결정

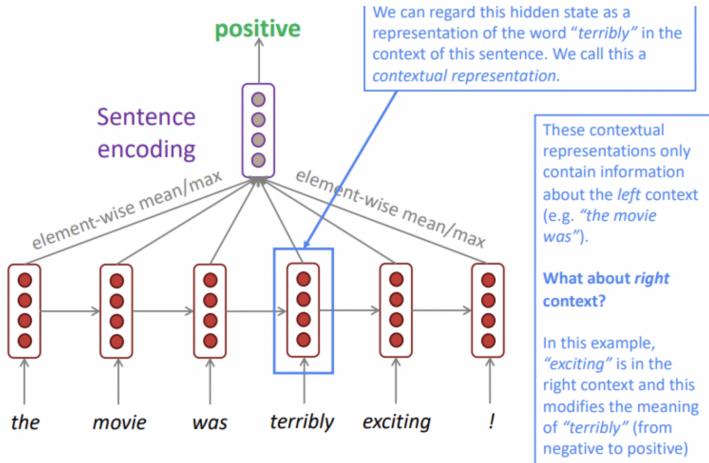
output gate: 생성된 현재 cell state $c(t)$ 에서 hidden state로 사용할 비율 결정



5. Bidirectional and Multi layer RNNs

단어는 문맥따라 다른 의미를 갖는데 왼쪽정보만 가져온다면 문제가 됨
아래 sentiment classification 예시:

terribly만 보면 부정적인 단어지만 exciting이 이후에 나오게 되면 긍정적인 단어가 됨!
즉 exciting을 보기 전까지 해당 감성 분류는 계속 부정적인 문제가 발생



문장을 거꾸로 입력해준 값을 concatenate하여 hidden state를 생성하게되면 terribly는 양방향의 문맥을 모두 반영한 representation을 갖게 됨

Bidirectional RNNs

