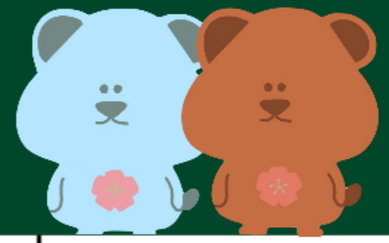
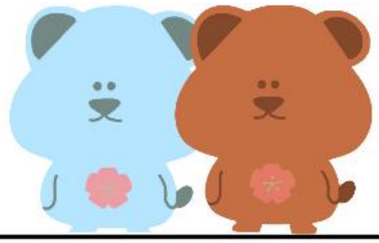


Week 06. Lecture 6 – Language Models and RNNs

발표자: 황채원, 조서영

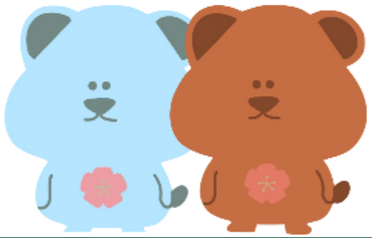
EWHA EURON

목차



EWHA EURON

Language Model

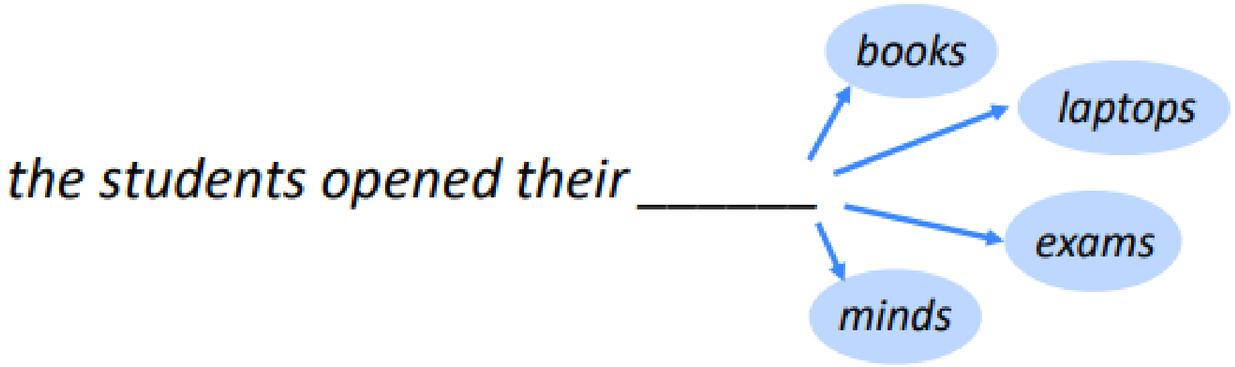


EWHA EURON

Language Model

#01 Language Model 이란?

- 주어진 단어의 시퀀스에 대해, 다음에 올 단어를 예측하는 것



Language Model

#01 Language Model 이란?

- $x^{(t+1)}$의 확률분포

$$P(\boldsymbol{x}^{(t+1)}|\ \boldsymbol{x}^{(t)},\ldots,\boldsymbol{x}^{(1)})$$

- 문장에 확률을 할당하여 다음에 나타날 단어를 예측하는 작업

$$P(\boldsymbol{x}^{(1)},\ldots,\boldsymbol{x}^{(T)}) = P(\boldsymbol{x}^{(1)}) \times P(\boldsymbol{x}^{(2)}|\ \boldsymbol{x}^{(1)}) \times \cdots \times P(\boldsymbol{x}^{(T)}|\ \boldsymbol{x}^{(T-1)},\ldots,\boldsymbol{x}^{(1)})$$

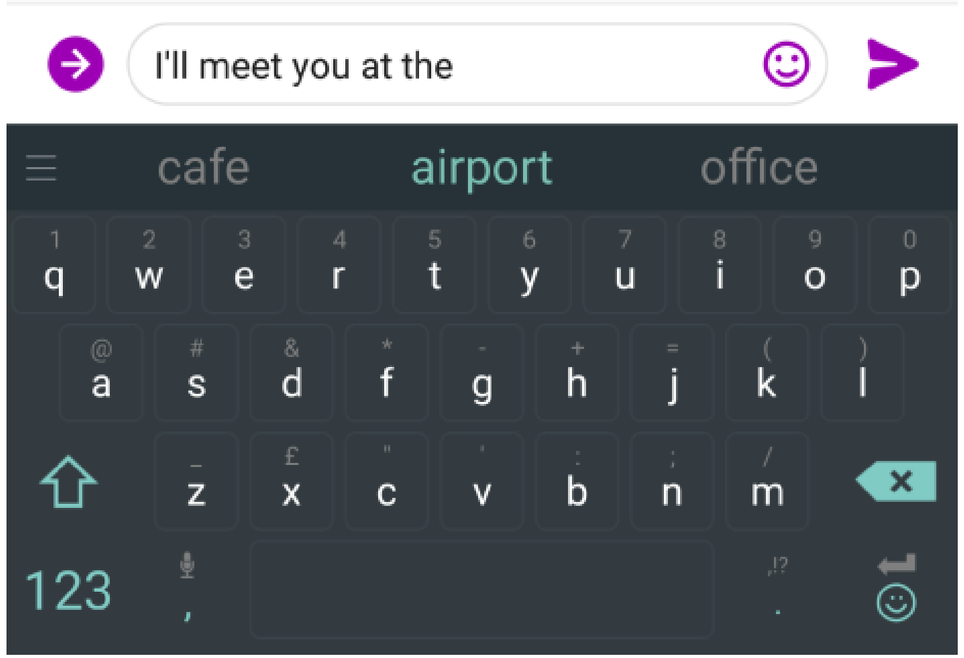
$$= \prod_{t=1}^{T} P(\boldsymbol{x}^{(t)}|\ \boldsymbol{x}^{(t-1)},\ldots,\boldsymbol{x}^{(1)})$$

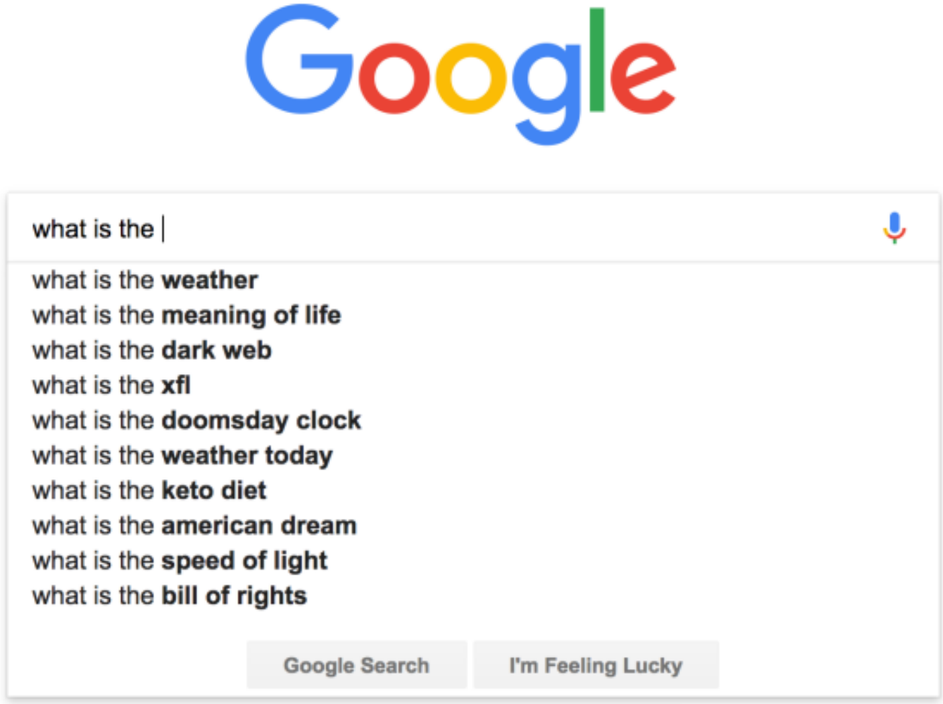
This is what our LM provides

Language Model

#01 Language Model 이란?

- You use Language Models every day!
- 기계번역, 음성인식, 자동완성 등에 이용된다





Language Model

#02 n-gram Language Models

- Neural Network 이전에 사용되었던 Language Model
- 이전에 등장한 n-1개의 단어들을 이용하여 다음 단어를 예측

N-gram은 n개의 연속된 단어들의 chunk

Definition: A *n*-gram is a chunk of *n* consecutive words.

- unigrams: “the”, “students”, “opened”, ”their”
- bigrams: “the students”, “students opened”, “opened their”
- trigrams: “the students opened”, “students opened their”
- 4-grams: “the students opened their”

Language Model

#02 n-gram Language Models

- Idea : 각 n-gram의 빈도를 수집하여 단어의 예측에 사용한다

$$P(\boldsymbol{x}^{(t+1)}|\boldsymbol{x}^{(t)},\ldots,\boldsymbol{x}^{(1)}) = P(\boldsymbol{x}^{(t+1)}|\overbrace{\boldsymbol{x}^{(t)},\ldots,\boldsymbol{x}^{(t-n+2)}}^{n-1 \text{ words}})$$ (assumption)

prob of a n-gram

prob of a (n-1)-gram

$$= \frac{P(\boldsymbol{x}^{(t+1)},\boldsymbol{x}^{(t)},\ldots,\boldsymbol{x}^{(t-n+2)})}{P(\boldsymbol{x}^{(t)},\ldots,\boldsymbol{x}^{(t-n+2)})}$$ (definition of conditional prob)

By counting them in some large corpus of text!

$$\approx \frac{\text{count}(\boldsymbol{x}^{(t+1)},\boldsymbol{x}^{(t)},\ldots,\boldsymbol{x}^{(t-n+2)})}{\text{count}(\boldsymbol{x}^{(t)},\ldots,\boldsymbol{x}^{(t-n+2)})}$$ (statistical approximation)

Language Model

#02 n-gram Language Models

Suppose we are learning a 4-gram Language Model.

~~as the proctor started the clock, the~~ *students opened their* _____

discard

condition on this

$$P(\boldsymbol{w}|\text{students opened their}) = \frac{\text{count}(\text{students opened their } \boldsymbol{w})}{\text{count}(\text{students opened their})}$$

For example, suppose that in the corpus:

- "students opened their" occurred 1000 times
- "students opened their books" occurred 400 times
  - → P(books | students opened their) = 0.4
- "students opened their exams" occurred 100 times
  - → P(exams | students opened their) = 0.1

4-gram이므로
앞의 (4-1)단어 이외의 단어는 무시하고
확률을 계산한다

Language Model

#03 Problems with n-gram Language Models

- Sparsity Problems
- Storage Problems
- Incoherence Problems

"students opened their ω"가
데이터에서 한 번도 등장하지 않는 경우
ω의 확률은 0이 되는 문제 발생 (분자가 0)

→ Smoothing
모든 $\omega \in V$ 의 count에 대해
매우 작은 값의 δ를 더해준다

$$P(\omega|students\ opened\ their) = \frac{Count(students\ opened\ their\ \omega)}{Count(students\ opened\ their)}$$

"students opened their"가
데이터에서 한 번도 등장하지 않는 경우
ω의 확률을 계산할 수 없는 문제 발생 (분모가 0)

→ Backoff
"opened their"를 사용하여
N-gram의 단계를 낮춘다

N이 커질수록 sparsity problem은 심해진다
따라서 일반적으로 n을 5 이하로 설정한다

EWHA EURON

Language Model

#03 Problems with n-gram Language Models

- Sparsity Problems
- Storage Problems
- Incoherence Problems

Corpus 내 모든 n-gram에 대한 Count를 저장해야 한다 → N이 커지거나 corpus가 커질수록 모델의 크기도 증가하는 문제 발생

$$P(\omega|students\ opened\ their) = \frac{Count(students\ opened\ their\ \omega)}{Count(students\ opened\ their)}$$

Language Model

#03 Problems with n-gram Language Models

- Sparsity Problems
- Storage Problems
- Incoherence Problems

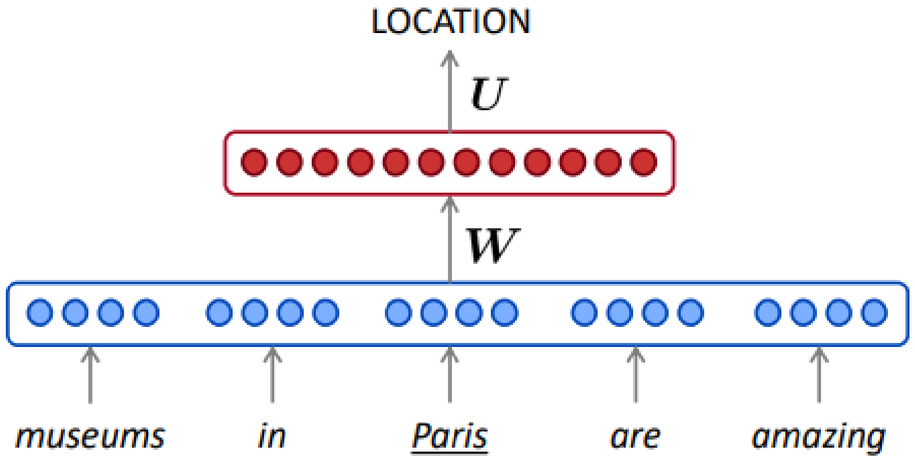
N-gram 모델이 문맥을 충분히 반영하지 못하는 문제

직전의 (n-1) 단어만 확률에 영향을 미칠 수 있기 때문이다

문맥을 충분히 반영하기 위해 n의 크기를 키우면 sparsity 문제가 심해진다

Language Model

#04 Neural Language Model



NER에 적용되었던 window-based neural network는
중심단어를 기준으로 앞뒤 window size가 결정된다

~~as the proctor started the clock~~ *the students opened their* _____

discard

fixed window

예측할 단어 이전의 window size를 고정한다

Language Model

#04 Neural Language Model

output distribution

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

hidden layer

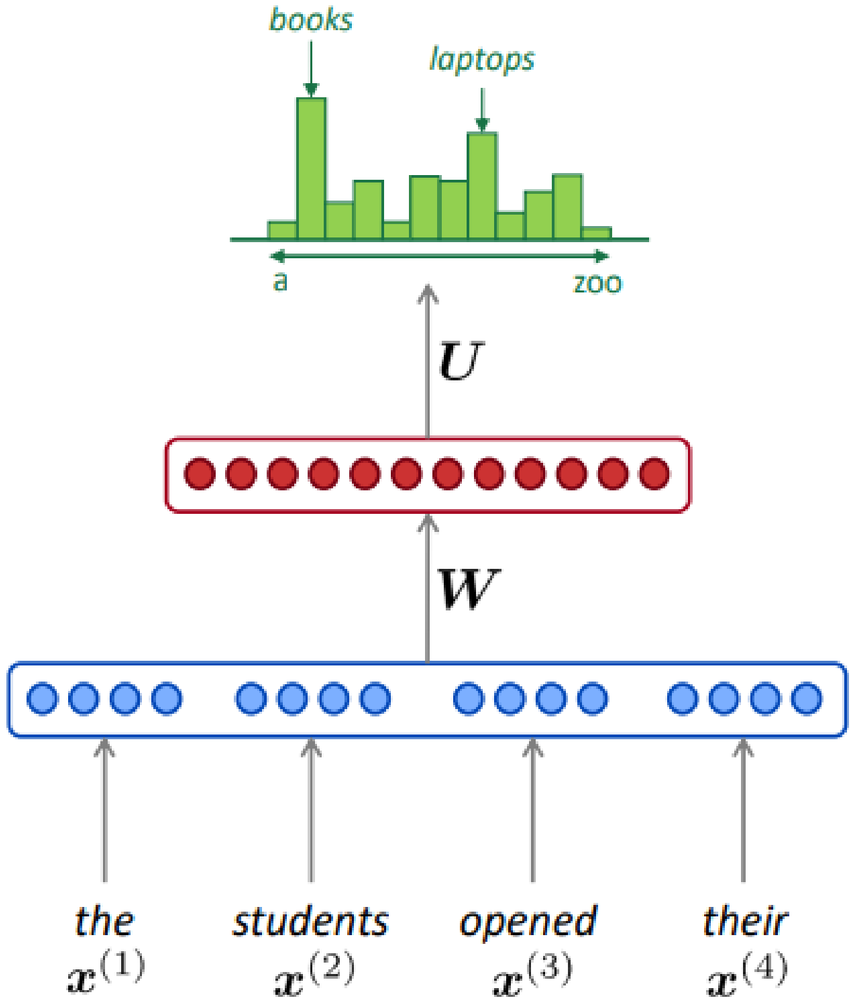
$$\boldsymbol{h} = f(\boldsymbol{W}\boldsymbol{e} + \boldsymbol{b}_1)$$

concatenated word embeddings

$$\boldsymbol{e} = [\boldsymbol{e}^{(1)}; \boldsymbol{e}^{(2)}; \boldsymbol{e}^{(3)}; \boldsymbol{e}^{(4)}]$$

words / one-hot vectors

$$\boldsymbol{x}^{(1)}, \boldsymbol{x}^{(2)}, \boldsymbol{x}^{(3)}, \boldsymbol{x}^{(4)}$$

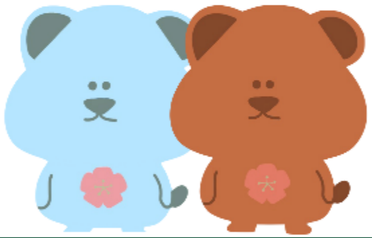


Language Model

#04 Neural Language Model

- 장점
  - Embedding을 통한 sparsity problem이 해결
  - 관측된 n-gram을 저장할 필요 없음
- 단점
  - 작은 window size가 문맥을 반영하지 못함
  - Window size 늘리면 W도 증가하므로 크기를 키우는데 한계가 존재한다
  - 단어의 위치에 따라 곱해지는 가중치가 달라서 모델이 비슷한 내용을 여러 번 학습하는 비효율성을 가진다

Recurrent Neural Network(RNN)

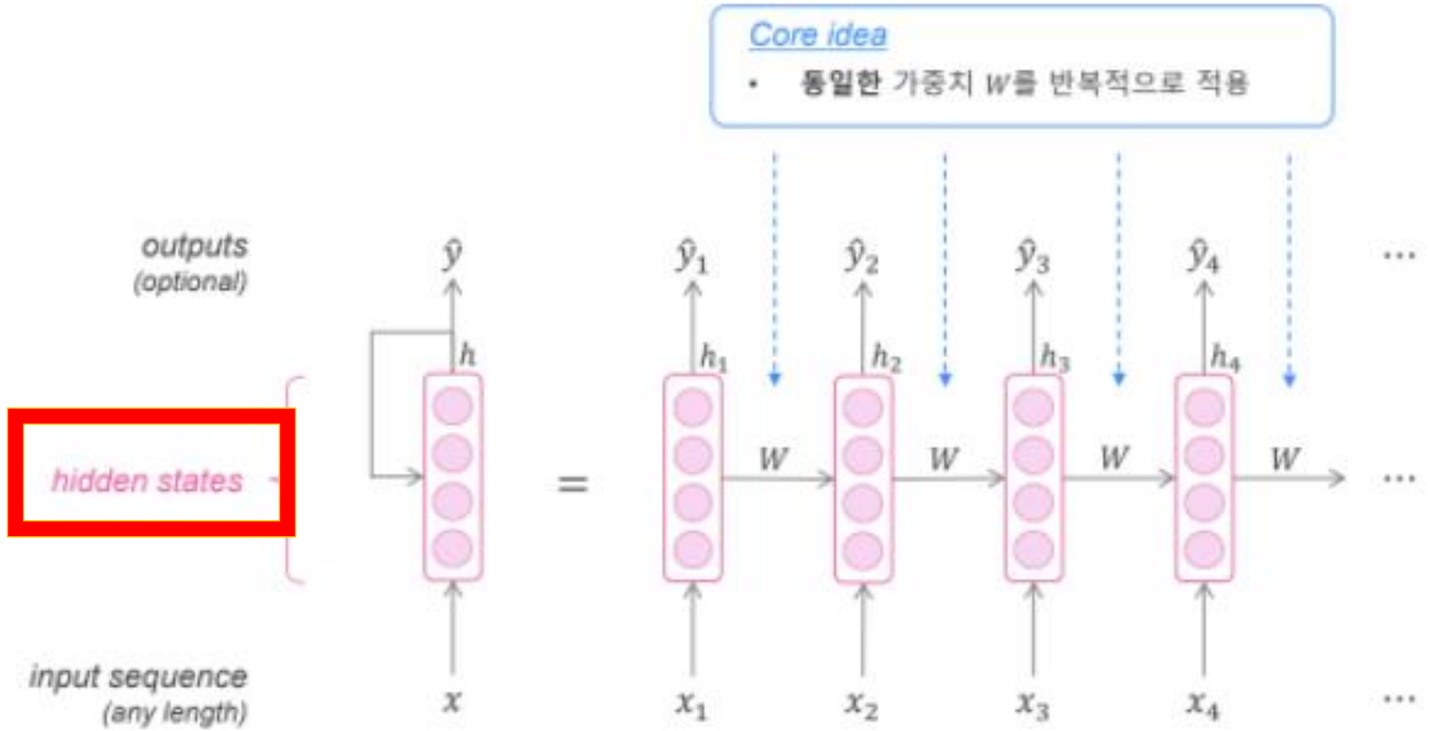


EWHA EURON

Recurrent Neural Network(RNN)

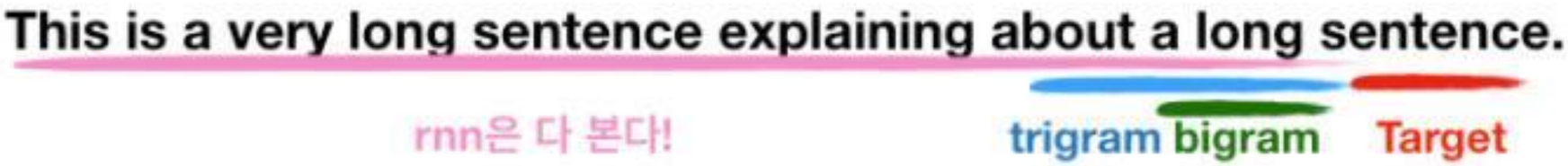
Core Idea: 동일한 가중치 W를 반복적으로 적용하자!

✓ **Recurrent Neural Network(RNN)**



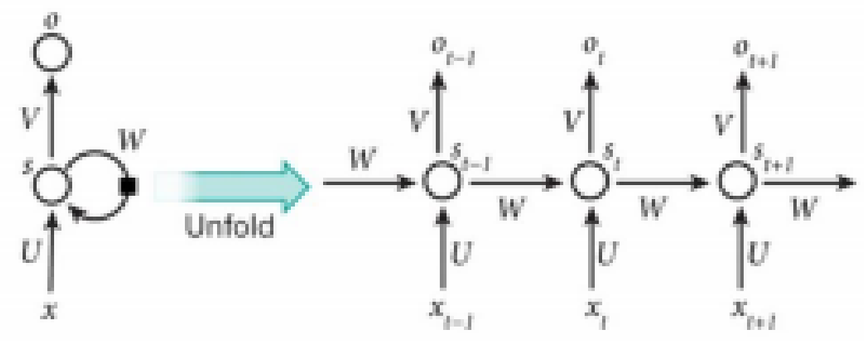
Recurrent Neural Network(RNN)

순차 정보를 처리하는 데 적합



- 기존의 신경망 구조 -> 모든 입력이 각각 독립적이라고 가정
- RNN -> 이전의 계산 결과에 영향을 받음 (hidden state 이용)

Recurrent Neural Network(RNN)



x_t: 시간 스텝(time step) t에서의 입력값

s_t: 시간 스텝 t 에서의 hidden state

$S_t=f(Ux_t+Ws_{t-1})$

비선형 함수 f는 보통 tanh이나 ReLU가 사용됨

첫 hidden state를 계산하기 위한 hidden state는 보통 0으로 초기화 시킨다고 함

o_t: 시간 스텝 t에서의 출력값

- Input 개수 = hidden state 개수
- 각 hidden state는 이전 단계의 hidden state와 해당 단계의 input을 이용하여 계산된다.
  - -> 즉 각 단계의 결과는 이전의 계산 결과에 영향을 받고, 시간을 고려한 순차 예측 가능
- output은 각 단계에 대해 모두 계산할 수도 있고 선택해서 계산할 수도 있다.

Recurrent Neural Network(RNN)

RNN Language Model

✓ **RNN** Language Model

output distribution

$\hat{y}_t = \text{softmax}(Uh_t + b_2) \in \mathbb{R}^{|V|}$

hidden states

$h_t = \sigma(W_h h_{(t-1)} + W_e e_t + b_1)$

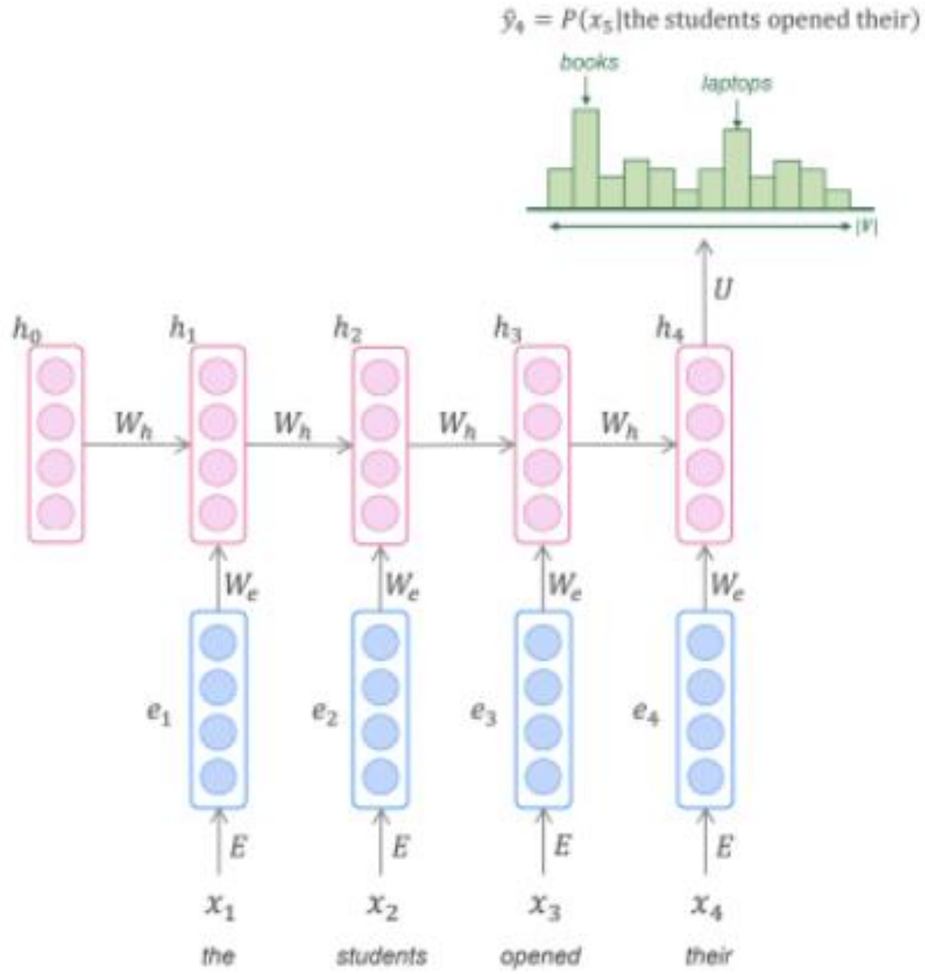
h_0 is the initial hidden state

word embeddings

$e_t = Ex_t$

words → one-hot vectors

x_1, x_2, x_3, x_4



예측 단어 확률

Recurrent Neural Network(RNN)

장단점

장점

- 입력의 길이에 제한이 없음
- 길이가 긴 timestep t 에 대해 처리 가능함 (이론적으로는 그러함, 실제로는…)
- 입력에 따라서 모델 크기가 증가하지 않음
- 매 timestep t 에 동일한 가중치를 적용하므로 symmetric

단점

- 다음 단계로 진행하기 위해서는 이전 단계의 계산이 완료되어야 하므로 계산이 병렬적으로 진행되지 않음
  -> 느림
- 실제로는 vanishing gradient problem 등의 문제가 있어 context가 반영되지 않는 경우가 있음

Recurrent Neural Network(RNN)

Training a RNN model

① $x_1, \ldots, x_T$ 의 단어들로 이루어진 시퀀스의 Corpus를 준비한다.

② $x_1, \ldots, x_T$를 차례대로 RNN-LM 에 주입하고, 매 step t 에 대한 $\hat{y}_t$ 를 계산한다.

- 주어진 단어에서부터 시작하여 그 다음 모든 단어들에 대한 확률을 예측

③ Step t 에 대한 손실함수 Cross-Entropy를 계산한다. (y_t is one-hot for x_{t+1})

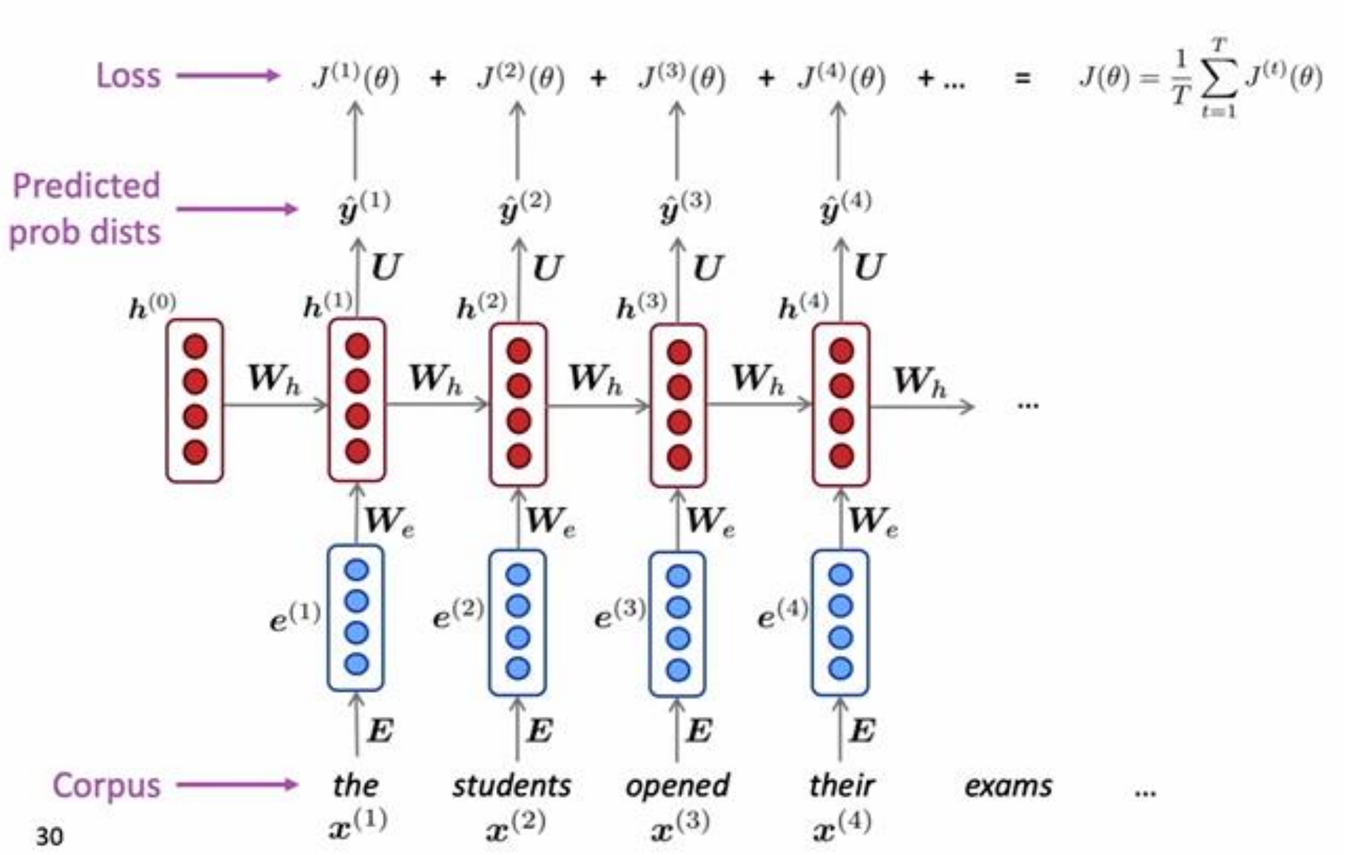
$$L_t = CE(y_t, \hat{y}_t) = -\sum_{w \in |V|} y_{t,w} \times \log(\hat{y}_{t,w}) = -\log(\hat{y}_{t,x_{t+1}})$$

④ 전제 step T 에 대해 계산한 손실함수 L_t 의 평균을 계산한다.

$$L = \frac{1}{T}\sum_{t=1}^{T} L_t = -\frac{1}{T}\sum_{t=1}^{T}\sum_{w=1}^{|V|} y_{t,w} \times \log(\hat{y}_{t,w}) = -\frac{1}{T}\sum_{t=1}^{T} -\log(\hat{y}_{t,x_{t+1}})$$

Recurrent Neural Network(RNN)

Training a RNN model

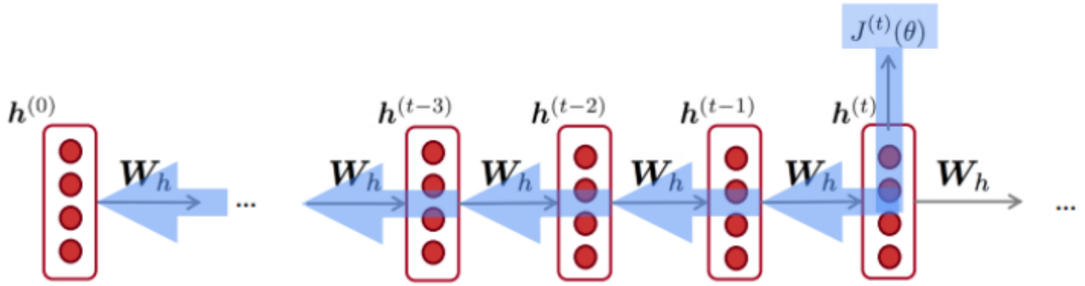


30

- 모든 corpus에 대해 loss와 gradient를 계산하는 것은 너무 많은 계산량
- stochastic gradient descent 이용해서 작은 데이터 단위에 대해 loss와 gradient를 계산하고 update한다.
- 즉 문장들의 집합인 batch에 대해 loss를 계산하고, gradient를 계산한 후 weight를 업데이트 한다.

Recurrent Neural Network(RNN)

Training a RNN model - Backpropagation



$$\frac{\partial J^{(t)}}{\partial \boldsymbol{W_h}} = \sum_{i=1}^{t} \frac{\partial J^{(t)}}{\partial \boldsymbol{W_h}}\Big|_{(i)}$$

Question: How do we calculate this?

Answer: Backpropagate over timesteps *i*=*t*,...,0, summing gradients as you go.
This algorithm is called **"backpropagation through time"**

- BPTT (Backpropagation Through Time)이라는, 약간 변형된 알고리즘을 사용
- Timestep을 거치면서 backpropagation을 하는데, 이때 gradients들을 더한다.

Evaluating Language Model

EWHA EURON

Evaluating Language Model

- Language model은 주어진 과거 단어들로부터 다음에 출현할 단어의 확률분포를 출력하는 모델
- Language model을 평가하는 대표적인 척도는 Perplexity
- **Perplexity = 출현할 단어의 확률에 대한 역수 = exponential of the cross-entropy loss**
- **Perplexity 값이 작을수록 좋은 Language model**

as the proctor started the clock, the students opened their _____

Evaluating Language Model

- Language model은 주어진 과거 단어들로부터 다음에 출현할 단어의 확률분포를 출력하는 모델
- Language model을 평가하는 대표적인 척도는 Perplexity
- **Perplexity = 출현할 단어의 확률에 대한 역수 = exponential of the cross-entropy loss**

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

Normalized by number of words

Inverse probability of corpus, according to Language Model

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

LM을 통해 예측한 corpus의 invers를 corpus 길이로 normalize 해준 값

n-gram model

Increasingly complex RNNs

Model	Perplexity
Interpolated Kneser-Ney 5-gram (Chelba et al., 2013)	67.6
RNN-1024 + MaxEnt 9-gram (Chelba et al., 2013)	51.3
RNN-2048 + BlackOut sampling (Ji et al., 2015)	68.3
Sparse Non-negative Matrix factorization (Shazeer et al., 2015)	52.9
LSTM-2048 (Jozefowicz et al., 2016)	43.7
2-layer LSTM-8192 (Jozefowicz et al., 2016)	30
Ours small (LSTM-2048)	43.9
Ours large (2-layer LSTM-2048)	39.8

Perplexity improves (lower is better)

Perplexity가 낮을 수록 좋은 Language Model

Evaluating Language Model

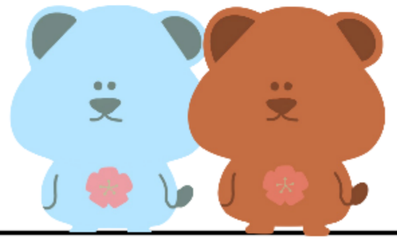
- Perplexity – 코드 구현 예

```
def compute_perplexity(model, sess, name):

 total_loss = 0
 total_predict_count = 0
 start_time = time.time()

 while True:
  try:
    loss, predict_count, batch_size = model.eval(sess)
    total_loss += loss * batch_size
    total_predict_count += predict_count
  except tf.errors.OutOfRangeError:
    break

 # exponential of the cross-entropy loss
 perplexity = utils.safe_exp(total_loss / total_predict_count)

 utils.print_time("  eval %s: perplexity %.2f" % (name, perplexity),  start_time)

 return perplexity
```

인자:
    model: model for compute perplexity.
    sess: tensorflow session to use.
    name: name of the batch.

리턴:
    The perplexity of the eval outputs.

THANK YOU



EWHA EURON